UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Jessica Tais de Souza Reinaldo

Using Item Response Theory to evaluate feature relevance in missing data scenarios

Recife

2022

Jessica Tais de Souza Reinaldo

Using Item Response Theory to evaluate feature relevance in missing data scenarios

A M.Sc. Dissertation presented to the Center of Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Concentration Area: Artificial Intelligence

Advisor: Ricardo Bastos C. Prudêncio

Co-Advisor: Telmo de Menezes e Silva Filho

Recife

2022

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

R364u Reinaldo, Jessica Tais de Souza

Using item response theory to evaluate feature relevance in missing data scenarios / Jessica Tais de Souza Reinaldo. – 2022.
236 f.: il., fig., tab.

Orientador: Ricardo Bastos C. Prudêncio.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.

Inclui referências e apêndices.

1. Inteligência artificial. 2. Ranqueamento de variáveis. I. Prudêncio, Ricardo Bastos C. (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2022-129

Jessica Tais de Souza Reinaldo

“Using Item Response Theory to evaluate feature relevance in missing data scenarios”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 29 de março de 2022.

BANCA EXAMINADORA

_____

Prof. Dr. George Darmiton da Cunha Cavalcante
Centro de Informática/UFPE

_____

Prof. Dr. Rafael Gomes Mantovani
Universidade Tecnológica Federal do Paraná, Campus Apucarana

_____

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE
(**Orientador**)

I dedicate this dissertation to my family, friends, and all the colleagues that I met on this journey.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Ricardo Prudêncio, and my co-advisor, Dr. Telmo Silva Filho, for their guidance (and patience) throughout the development of this work. Your help, through enriching discussions and feedbacks, was fundamental to open my mind on several aspects of the work we build together.

I also thank all teachers and colleagues I had during my stay at CIn, from whom I learned a lot about several aspects of machine learning, academic life, and cultural differences. I want to specially thank my colleagues João Moraes and Manuel Ferreira Jr. for their valuable inputs on our weekly gatherings, virtual or not.

A big thanks is also due to all my coworkers and colleagues at CW, who always bring colorful discussions about all aspects of AI to the table, which are always amazing learning experiences.

Finally, in these unusual moments where the boundaries of work place and our homes became foggy, I also have to thank my family and friends, who were always with me, even at the most difficult times.

ABSTRACT

Item Response Theory (IRT) has been historically used to evaluate the latent abilities of human respondents to a set of items. Recently, efforts have been made to propose solutions that use IRT to solve classification problems, where the respondents are classifiers and the items are the instances of a dataset. Most of the initial works that tried to tackle this problem used a dichotomous IRT model, which is capable of modelling the classification problem only in terms of correct and wrong predictions. B3-IRT offers a powerful tool to analyze datasets and classifiers, as the response is continuous, so instead of representing the predictions just as right or wrong answers, the response can be represented by the probability of a correct prediction. Although the IRT formulation can provide rich information about the behavior of the models towards the instances of a dataset, no previous work has investigated the application of IRT to rank features in an instance-based approach, or even to evaluate how missing data can impact the IRT parameters for instances (difficulty and discrimination) and classifiers (ability). We propose a workflow that uses B3-IRT in missing data scenarios to evaluate the relevance of features both locally for each instance of a dataset, and globally for the whole dataset. In this workflow, data is missing at test time, and missing values are filled out with imputed values, in order to evaluate how much the missing data can affect the ability of classifiers and difficulty and discrimination of instances. This novel application represents an alternative to feature selection and feature ranking techniques that is capable to provide an overview of feature relevance both at global and instance level.

Keywords: item response theory; missing data; feature gathering; feature ranking.

RESUMO

A Teoria de Resposta ao Item (TRI) tem sido historicamente usada para avaliar as habilidades latentes de respondentes humanos quando estes respondem a um conjunto de questões, chamadas de itens do problema de TRI. Recentemente, a comunidade ciêntifica começou a propor soluções que utilizem a TRI para resolver problemas de classificação, onde os respondentes são classificadores e os itens são as instâncias de um conjunto de dados. A maioria dos trabalhos iniciais que tentaram resolver este problema utilizou um modelo dicotômico de TRI, que é capaz de modelar o problema de classificação apenas em termos de previsões corretas e incorretas. O B3-IRT oferece uma formulação mais poderosa para esta aplicação de TRI, já que a resposta deste modelo é contínua, portanto, em vez de representar as predições de um modelo classificador apenas como respostas certas ou erradas (dicotômico), a resposta pode ser representada pela probabilidade de uma predição correta. Embora a formulação da TRI possa conter muita informação sobre o comportamento dos modelos em relação às instâncias de um conjunto de dados, nenhum trabalho anterior investigou a aplicação da TRI para classificar a relevância ou importância das variáveis de um conjunto de dados em uma abordagem baseada nas próprias instâncias, ou mesmo avaliar como dados faltantes podem afetar os parâmetros da TRI para instâncias (dificuldade e discriminação) e classificadores (habilidade). Neste trabalho, nós propomos um *workflow* que usa B3-IRT em cenários de dados faltantes para avaliar a relevância dos variáveis tanto localmente para cada instância quanto globalmente para todo o conjunto de dados. Nesse *workflow*, os dados faltantes ocorrem apenas no momento do teste, e os valores faltantes são preenchidos com valores imputados, a fim de avaliar o quanto os dados faltantes podem afetar a habilidade dos classificadores e a dificuldade e discriminação das instâncias. Esta nova abordagem proposta neste trabalho representa uma alternativa às técnicas de seleção e ranqueamento de variáveis capaz de fornecer uma visão geral da relevância das variáveis de um conjunto de dados tanto em nível global quanto em nível de instância.

Palavras-chaves: teoria de resposta ao item; dados faltantes; coleta de variáveis; ranqueamento de variáveis.

LIST OF FIGURES

Figure 1 – ICC curve with difference in difficulty 28
Figure 2 – ICC curve with difference in discrimination 29
Figure 3 – ICC curve with difference in guessing 29
Figure 4 – ICC for the 1PL model 30
Figure 5 – IIC for the 1PL model 31
Figure 6 – ICC for the 2PL model 32
Figure 7 – IIC for the 2PL model 33
Figure 8 – ICC for the 3PL model 34
Figure 9 – IIC for the 3PL model 35
Figure 10 – Diagram for BIFI Workflow: Part I 51
Figure 11 – Diagram for BIFI Workflow: Part II 51
Figure 12 – Diagram for BIFI Workflow: Part III 52
Figure 13 – Diagram for Algorithm 2 56
Figure 14 – Diagram for Algorithm 3 58
Figure 15 – Diagram for Algorithm 4 59
Figure 16 – 2-component PCA per data set 63
Figure 17 – Heatmap of median feature ranking for Contraceptive with Simple imputation 75
Figure 18 – Heatmap of median feature ranking for Contraceptive with KNN imputation 76
Figure 19 – Heatmap of median feature ranking for Pima with Simple imputation . . . 76
Figure 20 – Heatmap of median feature ranking for Pima with Knn imputation 77
Figure 21 – Instance difficulty values for different scenarios: Pima dataset 80
Figure 22 – CDD of feature ranking: Contraceptive dataset with Simple imputation . . 88
Figure 23 – CDD of feature ranking: Pima dataset with Simple imputation 88
Figure 24 – OLS model fit for Contraceptive dataset - Simple Imputation 91
Figure 25 – OLS model fit for Pima dataset - Simple Imputation 92
Figure 26 – OLS model fit for Contraceptive dataset - KNN Imputation 93
Figure 27 – OLS model fit for Pima dataset - KNN Imputation 94
Figure 28 – Difficulty dispersion for the Contraceptive dataset - Simple Imputation . . . 114
Figure 29 – Difficulty dispersion for the Pima dataset - Simple Imputation 115
Figure 30 – Difficulty dispersion for the Wheat dataset - Simple Imputation 116

Figure 31 – Difficulty dispersion for the Haberman dataset - Simple Imputation 116
Figure 32 – Difficulty dispersion for the Cleveland dataset - Simple Imputation 117
Figure 32 – Difficulty dispersion for the Ionosphere dataset - Simple Imputation 119
Figure 31 – Difficulty dispersion for the Wine dataset - Simple Imputation 122
Figure 31 – Difficulty dispersion for the Yeast dataset - Simple Imputation 123
Figure 32 – Difficulty dispersion for the Contraceptive dataset - KNN Imputation 125
Figure 33 – Difficulty dispersion for the Pima dataset - KNN Imputation 126
Figure 34 – Difficulty dispersion for the Wheat dataset - KNN Imputation 127
Figure 35 – Difficulty dispersion for the Haberman dataset - KNN Imputation 127
Figure 36 – Difficulty dispersion for the Cleveland dataset - KNN Imputation 128
Figure 36 – Difficulty dispersion for the Ionosphere dataset - KNN Imputation 130
Figure 35 – Difficulty dispersion for the Wine dataset - KNN Imputation 133
Figure 35 – Difficulty dispersion for the Yeast dataset - KNN Imputation 134
Figure 36 – Difficulty dispersion for the Contraceptive dataset - Iterative Imputation . . 136
Figure 37 – Difficulty dispersion for the Pima dataset - Iterative Imputation 137
Figure 38 – Difficulty dispersion for the Wheat dataset - Iterative Imputation 138
Figure 39 – Difficulty dispersion for the Haberman dataset - Iterative Imputation 138
Figure 40 – Difficulty dispersion for the Cleveland dataset - Iterative Imputation 139
Figure 40 – Difficulty dispersion for the Ionosphere dataset - Iterative Imputation 141
Figure 39 – Difficulty dispersion for the Wine dataset - Iterative Imputation 144
Figure 39 – Difficulty dispersion for the Yeast dataset - Iterative Imputation 145
Figure 40 – Discrimination dispersion for the Contraceptive dataset - Simple Imputation 147
Figure 41 – Discrimination dispersion for the Pima dataset - Simple Imputation 148
Figure 42 – Discrimination dispersion for the Wheat dataset - Simple Imputation 149
Figure 43 – Discrimination dispersion for the Haberman dataset - Simple Imputation . . 149
Figure 44 – Discrimination dispersion for the Cleveland dataset - Simple Imputation . . 150
Figure 44 – Discrimination dispersion for the Ionosphere dataset - Simple Imputation . 152
Figure 43 – Discrimination dispersion for the Wine dataset - Simple Imputation 155
Figure 43 – Discrimination dispersion for the Yeast dataset - Simple Imputation 156
Figure 44 – Discrimination dispersion for the Contraceptive dataset - KNN Imputation . 158
Figure 45 – Discrimination dispersion for the Pima dataset - KNN Imputation 159
Figure 46 – Discrimination dispersion for the Wheat dataset - KNN Imputation 160
Figure 47 – Discrimination dispersion for the Haberman dataset - KNN Imputation . . . 160

Figure 48 – Discrimination dispersion for the Cleveland dataset - KNN Imputation . . . 161
Figure 48 – Discrimination dispersion for the Ionosphere dataset - KNN Imputation . . 163
Figure 47 – Discrimination dispersion for the Wine dataset - KNN Imputation 166
Figure 47 – Discrimination dispersion for the Yeast dataset - KNN Imputation 167
Figure 48 – Discrimination dispersion for the Contraceptive dataset - Iterative Imputation169
Figure 49 – Discrimination dispersion for the Pima dataset - Iterative Imputation 170
Figure 50 – Discrimination dispersion for the Wheat dataset - Iterative Imputation . . . 171
Figure 51 – Discrimination dispersion for the Haberman dataset - Iterative Imputation . 171
Figure 52 – Discrimination dispersion for the Cleveland dataset - Iterative Imputation . 172
Figure 52 – Discrimination dispersion for the Ionosphere dataset - Iterative Imputation . 174
Figure 51 – Discrimination dispersion for the Wine dataset - Iterative Imputation 177
Figure 51 – Discrimination dispersion for the Yeast dataset - Iterative Imputation . . . 178
Figure 52 – Heatmap of median feature ranking for Contraceptive with Simple imputation192
Figure 53 – Heatmap of median feature ranking for Pima with Simple imputation . . . 192
Figure 54 – Heatmap of median feature ranking for Wheat with Simple imputation . . . 193
Figure 55 – Heatmap of median feature ranking for Haberman with Simple imputation . 193
Figure 56 – Heatmap of median feature ranking for Cleveland with Simple imputation . 193
Figure 57 – Heatmap of median feature ranking for Ionosphere with Simple imputation 194
Figure 58 – Heatmap of median feature ranking for Wine with Simple imputation . . . 194
Figure 59 – Heatmap of median feature ranking for Yeast with Simple imputation . . . 195
Figure 60 – Heatmap of median feature ranking for Contraceptive with Knn imputation 195
Figure 61 – Heatmap of median feature ranking for Pima with Knn imputation 196
Figure 62 – Heatmap of median feature ranking for Wheat with Knn imputation 196
Figure 63 – Heatmap of median feature ranking for Haberman with Knn imputation . . 196
Figure 64 – Heatmap of median feature ranking for Cleveland with Knn imputation . . 197
Figure 65 – Heatmap of median feature ranking for Ionosphere with Knn imputation . . 197
Figure 66 – Heatmap of median feature ranking for Wine with Knn imputation 198
Figure 67 – Heatmap of median feature ranking for Yeast with Knn imputation 198
Figure 68 – Heatmap of median feature ranking for Contraceptive with Iterative imputation 199
Figure 69 – Heatmap of median feature ranking for Pima with Iterative imputation . . . 199
Figure 70 – Heatmap of median feature ranking for Wheat with Iterative imputation . . 199
Figure 71 – Heatmap of median feature ranking for Haberman with Iterative imputation 200

Figure 72 – Heatmap of median feature ranking for Cleveland with Iterative imputation 200
Figure 73 – Heatmap of median feature ranking for Ionosphere with Iterative imputation 200
Figure 74 – Heatmap of median feature ranking for Wine with Iterative imputation . . . 201
Figure 75 – Heatmap of median feature ranking for Yeast with Iterative imputation . . 201
Figure 76 – CDD for Ability per Scenario: Contraceptive with Simple Imputation 202
Figure 77 – CDD for Ability per Scenario: Pima with Simple Imputation 202
Figure 78 – CDD for Ability per Scenario: Wheat with Simple Imputation 202
Figure 79 – CDD for Ability per Scenario: Haberman with Simple Imputation 203
Figure 80 – CDD for Ability per Scenario: Cleveland with Simple Imputation 203
Figure 81 – CDD for Ability per Scenario: Wine with Simple Imputation 203
Figure 82 – CDD for Ability per Scenario: Yeast with Simple Imputation 203
Figure 83 – CDD for Ability per Scenario: Contraceptive with KNN Imputation 204
Figure 84 – CDD for Ability per Scenario: Pima with KNN Imputation 204
Figure 85 – CDD for Ability per Scenario: Wheat with KNN Imputation 204
Figure 86 – CDD for Ability per Scenario: Haberman with KNN Imputation 204
Figure 87 – CDD for Ability per Scenario: Cleveland with KNN Imputation 205
Figure 88 – CDD for Ability per Scenario: Wine with KNN Imputation 205
Figure 89 – CDD for Ability per Scenario: Yeast with KNN Imputation 205
Figure 90 – CDD for Ability per Scenario: Contraceptive with Iterative Imputation . . . 206
Figure 91 – CDD for Ability per Scenario: Pima with Iterative Imputation 206
Figure 92 – CDD for Ability per Scenario: Wheat with Iterative Imputation 206
Figure 93 – CDD for Ability per Scenario: Haberman with Iterative Imputation 206
Figure 94 – CDD for Ability per Scenario: Cleveland with Iterative Imputation 206
Figure 95 – CDD for Ability per Scenario: Wine with Iterative Imputation 207
Figure 96 – CDD for Ability per Scenario: Yeast with Iterative Imputation 207
Figure 97 – OLS model fit for Contraceptive dataset - Simple Imputation 208
Figure 98 – OLS model fit for Pima dataset - Simple Imputation 209
Figure 99 – OLS model fit for Wheat dataset - Simple Imputation 210
Figure 100 – OLS model fit for Haberman dataset - Simple Imputation 210
Figure 101 – OLS model fit for Cleveland dataset - Simple Imputation 211
Figure 101 – OLS model fit for Ionosphere dataset - Simple Imputation 213
Figure 100 – OLS model fit for Wine dataset - Simple Imputation 216
Figure 100 – OLS model fit for Yeast dataset - Simple Imputation 217

Figure 101 – OLS model fit for Contraceptive dataset - KNN Imputation 218
Figure 102 – OLS model fit for Pima dataset - KNN Imputation 219
Figure 103 – OLS model fit for Wheat dataset - KNN Imputation 220
Figure 104 – OLS model fit for Haberman dataset - KNN Imputation 220
Figure 105 – OLS model fit for Cleveland dataset - KNN Imputation 221
Figure 105 – OLS model fit for Ionosphere dataset - KNN Imputation 223
Figure 104 – OLS model fit for Wine dataset - KNN Imputation 226
Figure 104 – OLS model fit for Yeast dataset - KNN Imputation 227
Figure 105 – OLS model fit for Contraceptive dataset - Iterative Imputation 228
Figure 106 – OLS model fit for Pima dataset - Iterative Imputation 229
Figure 107 – OLS model fit for Wheat dataset - Iterative Imputation 230
Figure 108 – OLS model fit for Haberman dataset - Iterative Imputation 230
Figure 109 – OLS model fit for Cleveland dataset - Iterative Imputation 231
Figure 109 – OLS model fit for Ionosphere dataset - Iterative Imputation 232
Figure 108 – OLS model fit for Wine dataset - Iterative Imputation 235
Figure 108 – OLS model fit for Yeast dataset - Iterative Imputation 236

LIST OF TABLES

Table 1 – Item parameters for the 1PL model 32
Table 2 – Item parameters for the 2PL model 33
Table 3 – Item parameters for the 3PL model 36
Table 4 – Description of the OLS input data 55
Table 5 – Datasets 62
Table 6 – Classifier Types 66
Table 7 – Feature selection methods 67
Table 8 – Parameters used for BIRT 69
Table 9 – BIRT configuration in classification tasks 70
Table 10 – Median feature ranking for Contraceptive with Simple imputation 75
Table 11 – Median feature ranking for Contraceptive with KNN imputation 76
Table 12 – Median feature ranking for Pima with Simple Imputation 77
Table 13 – Median feature ranking for Pima with KNN Imputation 77
Table 14 – Instance difficulty values for different scenarios: Contraceptive dataset . . . 79
Table 15 – Imputation for an instance: Contraceptive 81
Table 16 – Imputation for an instance: Pima Diabetes 81
Table 17 – Mean difficulty per scenario: Contraceptive with Simple Imputation 83
Table 18 – Mean difficulty per scenario: Pima with Simple Imputation 83
Table 19 – Mean discrimination per scenario: Contraceptive with Simple Imputation . . 84
Table 20 – Mean discrimination per scenario: Pima with Simple Imputation 84
Table 21 – Ability per model and scenario: Contraceptive with Simple Imputation . . . 85
Table 22 – Ability per model and scenario: Pima with Simple Imputation 85
Table 23 – Ability change (%) per scenario: Contraceptive with Simple Imputation . . . 86
Table 24 – Ability change (%) per scenario: Contraceptive with KNN Imputation 87
Table 25 – Ability change (%) per scenario: Pima with KNN Imputation 87
Table 26 – Ability change (%) per scenario: Pima with Simple Imputation 88
Table 27 – Euclidean distance per imputation for Contraceptive dataset 90
Table 28 – Euclidean distance per imputation for Pima dataset 90
Table 29 – OLS results for Contraceptive with Simple Imputation 94
Table 30 – OLS results for Pima with Simple Imputation 95

Table 31 – Mean feature ranking for Contraceptive with Simple Imputation 179
Table 32 – Mean feature ranking for Pima with Simple Imputation 179
Table 33 – Mean feature ranking for wheat with Simple Imputation 180
Table 34 – Mean feature ranking for Haberman with Simple Imputation 180
Table 35 – Mean feature ranking for Cleveland with Simple Imputation 181
Table 36 – Mean feature ranking for Ionosphere with Simple Imputation 182
Table 37 – Mean feature ranking for Wine with Simple Imputation 183
Table 38 – Mean feature ranking for Yeast with Simple Imputation 183
Table 39 – Mean feature ranking for Contraceptive with KNN Imputation 184
Table 40 – Mean feature ranking for Pima with KNN Imputation 184
Table 41 – Mean feature ranking for wheat with KNN Imputation 184
Table 42 – Mean feature ranking for Haberman with KNN Imputation 185
Table 43 – Mean feature ranking for Cleveland with KNN Imputation 185
Table 44 – Mean feature ranking for Ionosphere with KNN Imputation 186
Table 45 – Mean feature ranking for Wine with KNN Imputation 187
Table 46 – Mean feature ranking for Yeast with KNN Imputation 187
Table 47 – Mean feature ranking for Contraceptive with Iterative Imputation 188
Table 48 – Mean feature ranking for Pima with Iterative Imputation 188
Table 49 – Mean feature ranking for wheat with Iterative Imputation 188
Table 50 – Mean feature ranking for Haberman with Iterative Imputation 189
Table 51 – Mean feature ranking for Cleveland with Iterative Imputation 189
Table 52 – Mean feature ranking for Ionosphere with Iterative Imputation 190
Table 53 – Mean feature ranking for Wine with Iterative Imputation 191
Table 54 – Mean feature ranking for Yeast with Iterative Imputation 191

LIST OF ABBREVIATIONS AND ACRONYMS

ABC	AdaBoost Classifier
AI	Artificial Intelligence
AIRT	Algorithmic Item-Response Theory
ALE	Arcade Learning Environment
BIFI	β^3-IRT for Feature Importance
CBC	CatBoost Classifier
CDD	Critical Difference Diagram
CND	Cumulative Normal Distribution
DL	Deep Learning
DT	Decision Tree
EM	Expectation-Maximization
ETC	Extra Trees Classifier
FCS	Fully Conditional Specification
FFNN	Feed-Forward Neural Network
FPR	False Positive Rate
GBC	Gradient Boosting Classifier
GBM	Gradient Boosting Machine
GD	Gradient Descent
GNB	Gaussian Naïve Bayes
GPC	Gaussian Process Classifier
GPU	Graphics Processing Unit
GRM	Graded Response Model
ICC	Item Characteristic Curve
IIC	Item Information Curve
IRT	Item-Response Theory

KNN	K-Nearest Neighbors
LDA	Linear Discriminant Analysis
LGBM	Light Gradient Boosting Models
MAR	Missing At Random
MCAR	Missing Completely At Random
MI	Multiple Imputation
MI	Mutual Information
MICE	Multivariate Imputation by Chained Equations
ML	Machine Learning
MLP	Multi-Layer Perceptron
MNAR	Missing Not At Random
MRMR	Minimum Redundancy Maximum Relevance
NLP	Natural Language Processing
OLS	Ordinary Least Squares
PCA	Principal Component Analysis
PL	Parameter Logistic
PSO	Particle Swarm Optimization
QDA	Quadratic Discriminant Analysis
RBF	Radial Basis Function
RF	Random Forest Classifier
RFE	Recursive Feature Elimination
RL	Reinforcement Learning
ROC-AUC	Area Under the Receiver Operating Curve
RTE	Recognizing Textual Entailment
SVC	Support Vector Machine Classifier
SVM	Support Vector Machine
TDIDT	Top-Down Induction of Decision Trees

TPR	True Positive Rate
XGB	XGBoost Classifier

LIST OF SYMBOLS

$\mathcal{B}$	Beta distribution
δ	difficulty parameter
a	discrimination parameter
θ	respondent ability parameter
Δ	difference symbol

CONTENTS

1 INTRODUCTION 21
1.1 PROBLEM AND MOTIVATION 21
1.2 DISSERTATION PROPOSAL 23
1.3 STRUCTURE 23
2 RELATED WORK 25
2.1 ITEM RESPONSE THEORY 25
2.1.1 IRT in Psychometrics 26
2.1.1.1 Dichotomous response models 26
2.1.1.1.1 Example: differences between 1PL, 2PL, and 3PL models 27
2.1.1.1.2 Example: toy problem 30
2.1.1.2 Polytomous response models 32
2.1.1.3 Graded response models 34
2.1.1.4 Continuous response models 35
2.1.2 IRT in Artificial Intelligence 36
2.2 MISSING DATA 41
2.3 MISSING DATA IMPUTATION 42
2.3.1 Single Imputation 43
2.3.2 Multiple Imputation 43
2.4 FEATURE GATHERING 44
2.5 FEATURE SELECTION AND FEATURE RANKING 44
2.6 FINAL CONSIDERATIONS 47
3 β^3-IRT FOR FEATURE IMPORTANCE ANALYSIS 49
3.1 PROPOSAL 49
3.1.1 Data cleaning 52
3.1.2 Classifiers' Evaluation 53
3.1.3 IRT Results 53
3.1.4 Missing feature loop 54
3.2 REGRESSION ANALYSIS 55
3.2.1 OLS Regression 55
3.2.2 Data Binarization 57

3.2.3 Classification analysis . . . 57
4 EXPERIMENTAL SETUP . . . 60
4.1 DATASETS . . . 61
4.2 CLASSIFIER MODELS . . . 63
4.2.1 Model Parameters . . . 66
4.3 FEATURE SELECTION METHODS . . . 67
4.4 IMPUTATION TECHNIQUES . . . 68
4.5 IRT MODEL . . . 69
4.6 OLS REGRESSION ANALYSIS . . . 70
4.6.1 OLS Regression . . . 70
4.6.2 Data Binarization . . . 71
4.6.3 Classification Analysis . . . 72
5 EXPERIMENTAL RESULTS . . . 74
5.1 COMPARISON TO FEATURE SELECTION TECHNIQUES . . . 74
5.2 INSTANCE-LEVEL ANALYSIS . . . 78
5.3 CASE STUDY . . . 80
5.3.1 Instance analysis . . . 80
5.3.2 Difficulty, discrimination, and ability change . . . 81
5.3.3 Imputation analysis . . . 89
5.4 REGRESSION ANALYSIS . . . 90
5.5 DISCUSSION . . . 96
6 CONCLUSION AND FUTURE WORK . . . 97
6.1 IDEAS FOR FUTURE WORK . . . 98
REFERENCES . . . 100
APPENDIX A – EXECUTION ENVIRONMENT . . . 104
APPENDIX B – DIFFICULTY DISPERSION PER SCENARIO . . 113
APPENDIX C – DISCRIMINATION DISPERSION PER SCENARIO146
APPENDIX D – MEDIAN RANKING . . . 179
APPENDIX E – MEDIAN FEATURE RANKING: HEATMAPS . . 192
APPENDIX F – CRITICAL DIFFERENCE DIAGRAMS . . . 202
APPENDIX G – OLS REGRESSION ANALYSIS . . . 208

1 INTRODUCTION

This dissertation addresses the problem of evaluating feature relevance in the context of classification by adopting Item-Response Theory (IRT). In Section 1.1, the addressed problem is motivated and introduced, followed by Section 1.2, in which the proposal of this dissertation is introduced. Finally, Section 1.3 presents the dissertation structure.

1.1 PROBLEM AND MOTIVATION

IRT models were first developed to tackle problems in the scope of Psychometrics, which is a field of study whose goal is to measure cognitive traits, which can be intelligence, personality, among others (LORD; NOVICK; BIRNBAUM, 1968; BIRNBAUM, 1968). The main advantage of IRT when compared to the existing tests at the time is the invariance of item and ability, which means that the parameters that characterize an item are not dependent on the parameters that characterize a respondent and that the parameters that characterize a respondent are not dependent on the set of items. In general, it is common to define a IRT problem in terms of its parameters (Parameter Logistic (PL)); 1PL problem has one single parameter (difficulty), a 2PL problem has two parameters (difficulty and discrimination), 3PL problem has three parameters (usually difficulty, discrimination, and guessing), and so on.

In the past few years, researchers started investigating the use of IRT models in Artificial Intelligence (AI) and Machine Learning (ML). The first experiments in this context map the instances of a dataset as the items and the prediction models as the respondents in the IRT setup, which gives difficulty and discrimination parameters associated with the instances of the dataset while assigning abilities to the prediction models (MARTÍNEZ-PLUMED et al., 2016). These authors investigate the concept of instance hardness using the difficulty parameter in IRT, and how the discrimination parameter can be interpreted in this context. Other authors proposed the use of IRT in Natural Language Processing (NLP) (LALOR; WU; YU, 2016), in weighted tree ensembles (CHEN; AHN, 2020), in feature selection (KLINE et al., 2020), among others. Most of the IRT models used in the aforementioned works are dichotomous. In turn, (CHEN et al., 2019) proposed a 2PL model where abilities and difficulties are drawn from the beta distribution, and restricted to the interval $[0, 1]$, which can be much richer than the dichotomous model when evaluating prediction models. In this work, the proposed model was

adopted to evaluate ML models that estimate class probabilities.

This dissertation addresses a different context in which it is assumed that not all feature values can be available in a classification dataset. In fact, missing data is a prevalent problem in real world datasets. The reason for the missingness is diverse and explained in more detail in the next chapter. It is difficult to understand and evaluate the impact of missing data in a dataset for a classification problem. Usually, the efforts can go in the direction of explaining which features are the most important, or which is the best imputation technique to deal with the missing values. However, understanding, for each instance, how much missing a feature value can impact on its correct classification, is a complicated and less studied task.

There are several ways of dealing with missing data: deleting the data that is missing, filling in with values that represent missingness or imputing a value to fill in the gaps and use the data as usual. Deleting the data is common although not recommended because otherwise perfectly good data can be thrown away because of a single missing value. Filling the gaps with a constant value to represent missingness can change the distribution of the data and the inter-variable relationships.

Regarding imputation, there are several techniques proposed in literature, some of them considered as effective even in the cases that are more difficult to deal with. However, even as it might be helpful in some cases, sometimes a variable can be so important in a given dataset that imputing its value could not be worth it. In such cases, if possible, it would be better to gather the missing information instead of imputing a value.

We define *feature gathering* as the task of gathering feature f that is missing for an instance i. Very often, to gather the value of a given feature can be much more a human choice. Take as an example a dataset that is built to detect whether a patient has breast cancer. The features of this dataset can be images of the breast in different angles, results of breast examinations, and so on. A missing feature in this context can mean that a patient has not taken a given exam. So it is much more a matter of cost and time than a matter of not being able to collect that information. It would be very relevant to understand, for that given patient, whether it would be worth it to take the missing exam or not, in order to have the best possible chance of correctly predicting the output, that is, whether that patient has breast cancer or not. As this is an individual analysis unique for each instance, the evaluation of how important it is to take this exam can vary deeply from one patient to another.

1.2 DISSERTATION PROPOSAL

We propose a novel solution to evaluate the relevance of gathering a feature on an instance basis. IRT has been adopted in previous work to evaluate difficulties of instances and abilities of classifiers. In our solution, we apply IRT in two different setups: with all features and with missing features. Then we evaluate ability change (for the classification models) and difficulty change (for the instances of the dataset) to evaluate the importance of each feature of the dataset, analyzing the overall performance and the individual instances and how they are affected by the missingness. In the experiments, we apply an IRT model with bounded responses in a classification task (CHEN et al., 2019).

Additionally, we create a prediction model to indicate which instances are more affected by the missingness of each individual feature. For this we discretize the difficulty change returned by the IRT models and adopted it as a target attribute to be predicted by a classifier. Thus it allow us to predict whether a given instance will be impacted by the lack of information on any particular feature of the dataset being analyzed.

Even though the works involving IRT in the field of ML have been increasing over the past few years, with a lot of attention being given to this research field, the effect of missing data in IRT has not been investigated in the literature as of yet. We propose a workflow with two main goals: understanding which is the overall most important feature in a dataset, by ranking the features using results of the IRT evaluation; understanding how the missingness each individual instance, and evaluate the importance of each feature for each instance.

The main contributions of the proposed work can be summarized as follows:

- Evaluating IRT in a missing feature scenario;
- Proposing an IRT solution to create a feature ranking for gathering;
- Developing prediction models to understand the impact of the missingness for each instance and provide a solution to evaluate the impact of missingness in unseen instances.

1.3 STRUCTURE

This dissertation is organized in six chapters, as follows. Chapter 1 introduces the problem and the context in which the present work is inserted, as well as the proposed solution to tackle

the problem. Chapter 2 presents the literature review covering IRT models and the application in Psychometrics and, more recently, in AI, an overview on feature selection techniques, missing data and imputation techniques, and feature gathering and the associated costs. Chapter 3 goes in depth in describing the proposal of the current dissertation. Chapter 4 presents detailed information about the setup used to evaluate our proposed solution. Chapter 5 shows the experimental results. Chapter 6 concludes the present work and introduces some ideas for future work.

2 RELATED WORK

In this chapter, we dive deeper into the main topics related to this dissertation. In Section 2.1 we discuss IRT models, applications, and the use of IRT in artificial intelligence applications. In Section 2.2 we discuss the different types of missing data and the impact that missing data can have on a dataset. In Section 2.5 we present the main types of feature selection techniques and how we can use the insights these methods give us to create a feature ranking for the features of a dataset. In Section 2.4 we discuss the topic of feature gathering in the context of missing data, when the collection (or not) of a given feature can be a choice associated with a cost. Finally, in Section 2.6 we discuss drawbacks of the topics previously discussed and introduce our proposal.

2.1 ITEM RESPONSE THEORY

The seminal work on IRT models is associated with (LORD; NOVICK; BIRNBAUM, 1968). In this same book an important paper on latent traits of evaluation models was also presented (BIRNBAUM, 1968). The most basic IRT model, the 1PL dichotomous response model, was presented there. Later on, there were experiments with polytomous models, graded response models, and finally, continuous response models. An IRT problem has at least one latent trait for its items (difficulty) (depending on the number of parameters), and one for the respondent (ability).

(HAMBLETON; SWAMINATHAN; ROGERS, 1991) is one of the most important books on IRT models. They define two basic postulates for an IRT problem:

- The performance of the respondent on a test item can be predicted by a set of factors (traits, abilities).
- The relationship between the performance of respondents to an item and the set of traits underlying item performance is called an item characteristic curve, and is monotonically increasing.

For several decades, the main applications of IRT models were in the field of Psychometrics. This has started to change in the past few years, where experiments with IRT applied in the field of ML started to pop up.

We delve into the traditional IRT in Psychometrics, and how the models evolved from the initial dichotomous to continuous in section 2.1.1. The applications of IRT in AI are discussed in section 2.1.2.

2.1.1 IRT in Psychometrics

The first applications of the IRT model were in the field of Psychometrics. The goal of IRT was to be a reliable evaluation tool that was capable to predict the *ability* of human respondents when answering a set of questions. This evaluation method went way beyond just counting the number of correct responses, or even assigning weights to different questions depending on their perceived *difficulty* level. Besides the item difficulty, different formulations of the IRT model can allow other parameters: the item *discrimination* and *guessing*.

The first proposed IRT models were able to map only binary responses: correct or incorrect. Soon it became clear that this could not be sufficient information for multiple response cases, so the polytomous models were proposed to address this issue. If the response categories had an intrinsic ordering, however, the polytomous models were not sufficient; then the graded response models were proposed to better characterize this ordering. Later on came the continuous response model, which was able to map continuous responses and not only discrete ones, being the best choice to model the scores of open-ended questions.

In the next subsections, we go into further detail on each of these models.

2.1.1.1 Dichotomous response models

The three most popular IRT models are dichotomous 1, 2 and 3PL models, depending on the number of item parameters that are incorporated to the model.

In the 1PL model (also known as the Rasch model, in homage to the person who proposed the first 1PL model), it is assumed that the only item parameter that influences the respondent ability is the difficulty, which is the location parameter in an Item Characteristic Curve (ICC). The greater the difficulty value of an item, the greater the ability of the respondent to predict its correct response with a chance higher than 50%. According to (LORD; NOVICK; BIRNBAUM, 1968), in some cases, in order to facilitate the interpretation of results, the ability values - which can range from $(-\infty, +\infty)$ - can be scaled to the $[0, 1]$ interval. In this case, the difficulty parameter for an item will vary inside the interval $[-2, +2]$, with higher values meaning more

difficult items.

The discrimination of the items in this type of formulation is fixed, which means that all items are considered to have the same discrimination power on separating the respondents based on their abilities. This is not a realistic take in some cases, and this is what 2PL models try to solve.

The 2PL model was first developed by Lord, Novick and Birnbaum (1968), based on the Cumulative Normal Distribution (CND) model (normal ogive), by substituting the 2PL for a 2-parameter normal ogive function. The 2PL model is very similar to the 1PL model, with the difference that it has two extra elements, one is a scaling factor to make the logistic similar to the ogive function, and the other is the item discrimination parameter. Items with steeper slopes are considered to be more discriminant in being able to differentiate the good from the bad respondents, that is, separating the respondents according to their ability levels. A negative discrimination value means that the item might have some problems, such as noise or misclassification. This is because when an item has negative discrimination, the probability of a respondent predicting the item correctly decreases with higher ability values.

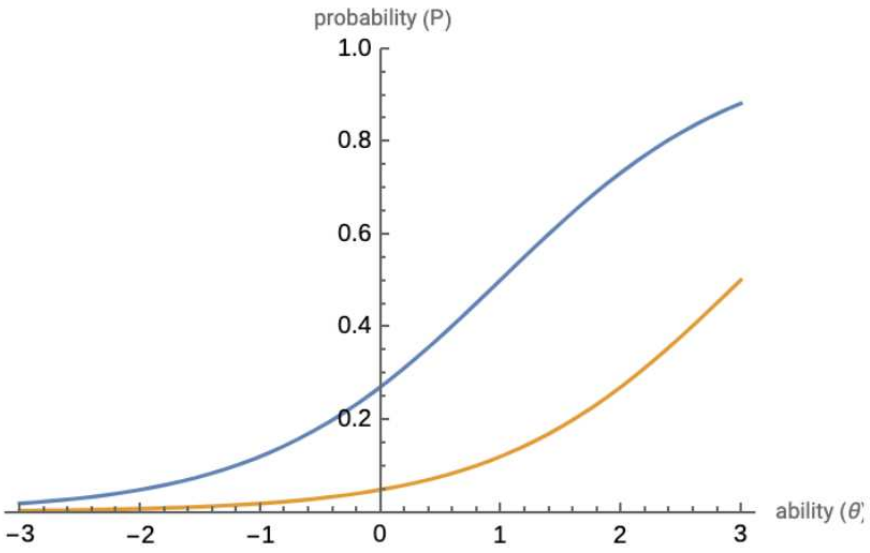
This model makes no allowance for guessing behavior, that is, it assumes that guessing is not possible. If the guessing parameter is crucial in an IRT formulation, one needs to use a 3PL model instead.

3PL models are similar to 2PL, with an additional item parameter, called *pseudo change level* parameter. It represents the probability of a respondent with low ability answering the item correctly. In the literature, this is sometimes also called the *guessing* parameter, but Lord, Novick and Birnbaum (1968) argues that this parameter usually assumes values that are smaller than the value that would result if respondents guessed the item at random, so this parameter is instead associated with a tactic of the test makers to create response options that might look correct when they are in fact wrong to fool low ability respondents.

2.1.1.1.1 Example: differences between 1PL, 2PL, and 3PL models

In order to show how the IRT parameters impact the ICC curves of the items, we will use a set of three figures in which one parameter changes at a time. The figures were generated using the Wolfram Alpha demonstration tool for IRT, which is available at <https://demonstrations.wolfram.com/ItemCharacteristicCurves/>. The curves show the probability (y-axis) of a respondent with a given ability (x-axis) giving the correct response to the item

Figure 1 – ICC curve with difference in difficulty



Source: Author's own elaboration

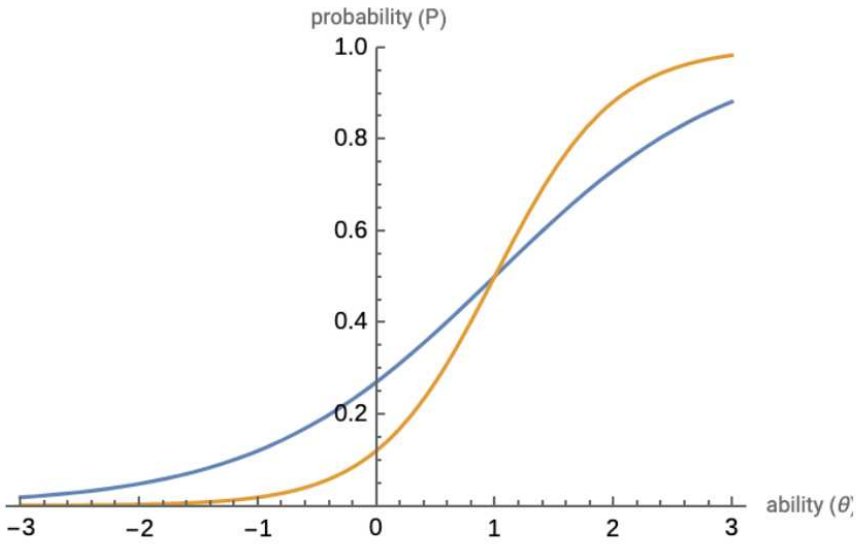
represented by the curve. The default values for each parameter are the following: 1.0 for difficulty, 1.0 for discrimination, and 0.0 for the guessing parameter.

Figure 1 shows the IRT curves for difficulty values of 1.0 (blue line) and 3.0 (red line). The difficulty parameter is related with the ability needed for a respondent to provide the correct response for an item. A higher difficulty means that the probability of giving a correct response increases slowly, and can even be low; in the example depicted in the figure, even a respondent with very high ability has about 40% chance to provide the correct response for the item whose ICC is defined by the red line, while it is about 90% for the blue line, where the item has a lower difficulty value.

Figure 2 shows the IRT curves for discrimination values of 1.0 (blue line) and 3.0 (red line). A higher discrimination value for an item generates steeper ICC curves, as it can better discriminate between a lower ability respondent and a higher ability respondent. The ICC curve depicted by the red line shows that the probability of correct response increases the most for the ability interval between 0.5 (probability around 20%) and 1.5 (probability around 80%).

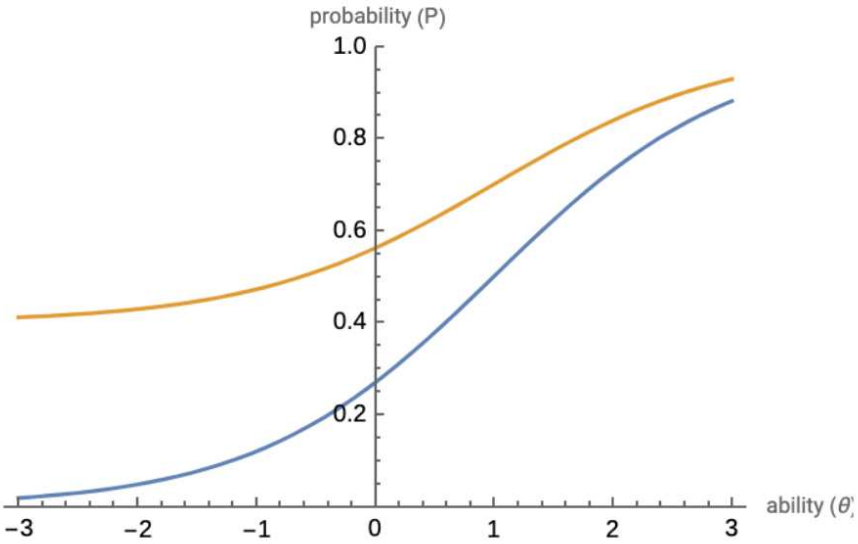
Figure 3 shows the IRT curves for guessing values of 0.0 (blue line) and 0.4 (red line). A guessing parameter $g > 0$ represents the probability of a respondent with the lowest ability of giving the correct response for the item. Higher guessing parameters also account for less steep curves.

Figure 2 – ICC curve with difference in discrimination



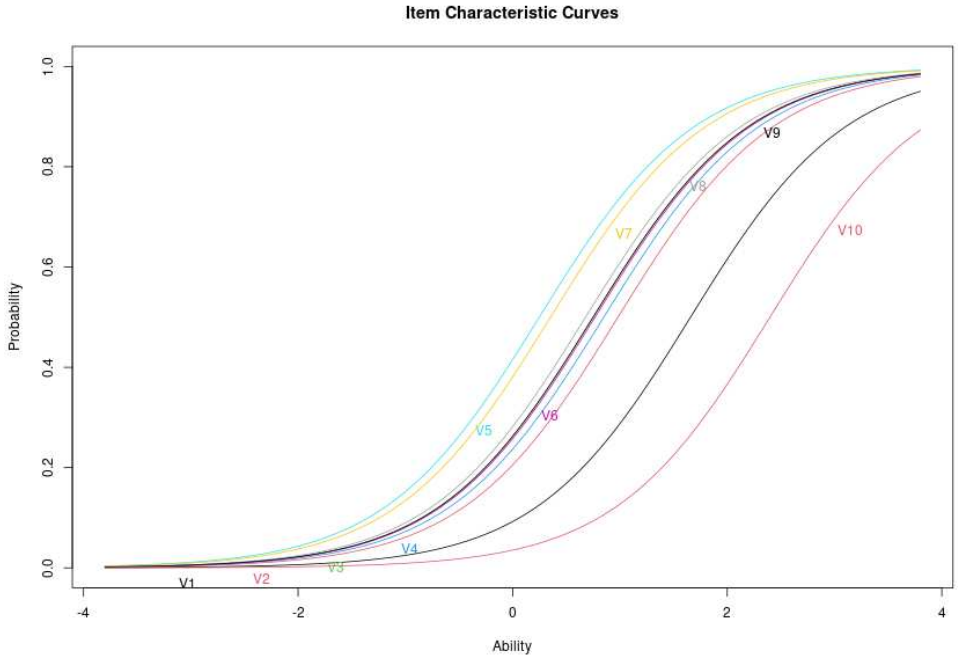
Source: Author's own elaboration

Figure 3 – ICC curve with difference in guessing



Source: Author's own elaboration

Figure 4 – ICC for the 1PL model



Source: Author's own elaboration (based on QuantDev tutorial)

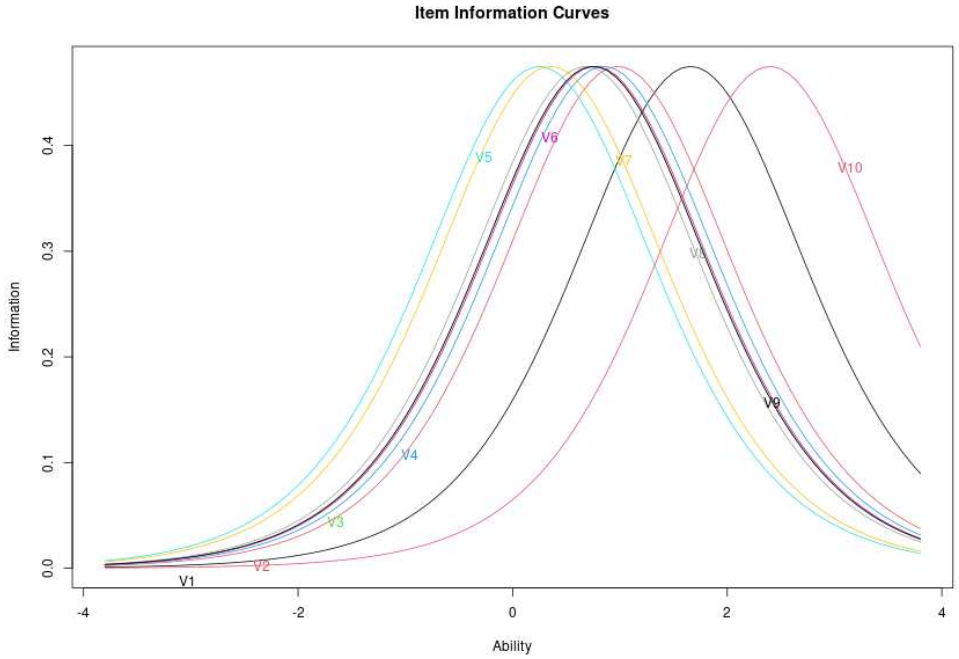
2.1.1.1.2 Example: toy problem

To show differences about each IRT model (1PL, 2PL, 3PL), we reproduce the IRT experiments with the *ltm* R package (RIZOPOULOS, 2006) from the QuantDev website of the Pennsylvania State University, which can be found at <https://quantdev.ssri.psu.edu/tutorials/introduction-irt-modeling>. The dataset has 10 different respondents to 500 items, and all responses are binary (0 or 1). We will show the results in terms of the ICC and Item Information Curve (IIC) plots, which show the relationship of the item parameters to the abilities of the respondents. The ICC models the probability of a respondent of a given ability providing a correct response for each item. The IIC, on the other hand, models the relationship between the information provided by each item for each ability level. There are items that might be more informative for lower or for higher levels of abilities.

Figure 4 shows the ICC for the items of the 1PL model. It can be seen that the steepness is the same for all the item curves, which happens because in the 1PL model all items have the same discrimination value. Figure 5 shows the IIC for the items of the 1PL model. All items have the same information value, which represents the discrimination parameter of the IRT model.

Figure 6 shows the ICC for the items of the 2PL model. In this case, the steepness of the

Figure 5 – IIC for the 1PL model



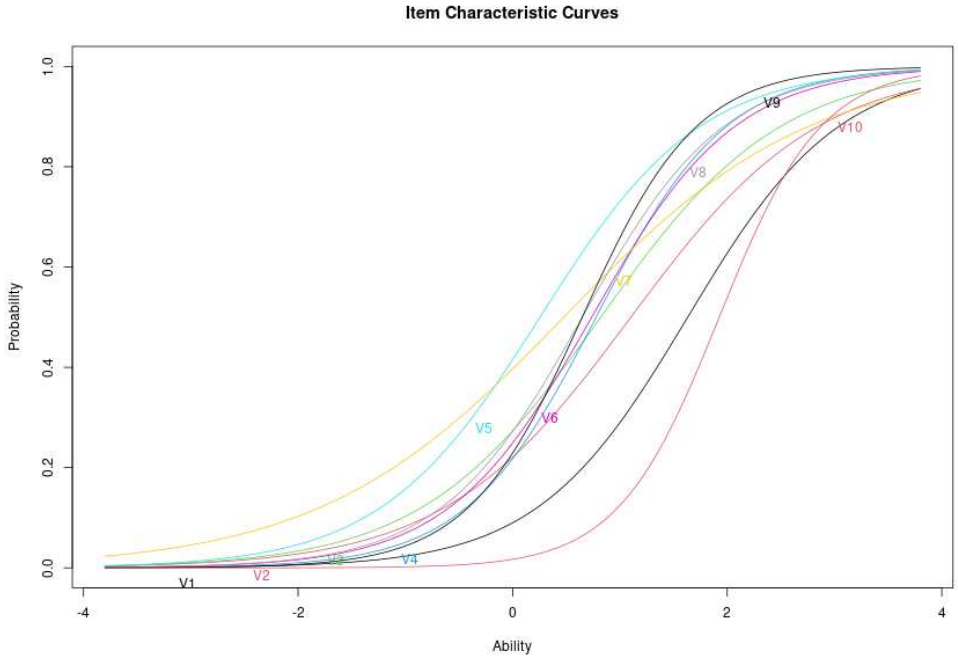
Source: Author's own elaboration (based on QuantDev tutorial)

slopes of the item curves are different, as the discrimination values are different for each item. Figure 7 shows the IIC for the items of the 2PL model. This curve shows more clearly the relationship between the discrimination of each item and the ability of the respondents. The figure shows that the information is higher for moderate abilities; extreme abilities, both on the lower and higher end, do not have items with high discrimination capabilities.

Figure 8 shows the ICC for the items of the 3PL model. Figure 9 shows the IIC for the items of the 3PL model. The curves here incorporate the guessing parameter, and the most evident difference when compared to the same curve in the 2PL model is for item V2, that is the one with the highest guessing score. The discrimination of this item increased significantly with the addition of the guessing parameter, specially for moderate abilities. This might be a relevant item to discriminate the good respondents from the worst ones.

Table 1 shows the table with the IRT item parameters for the 1PL model, in this case, only the difficulty; the discrimination is fixed at 1.38, and the guessing is fixed at 1.0. Table 2 shows the table with the IRT item parameters for the 2PL model, the difficulty and the discrimination. The guessing parameter is fixed at 1.0. Table 3 shows the table with the IRT item parameters for the 3PL model, the difficulty, the discrimination, and the guessing. The guessing parameter value is very low (near zero) for most of the items, but slightly higher for items *V2*, *V4*, and *V9*.

Figure 6 – ICC for the 2PL model



Source: Author's own elaboration (based on QuantDev tutorial)

Table 1 – Item parameters for the 1PL model

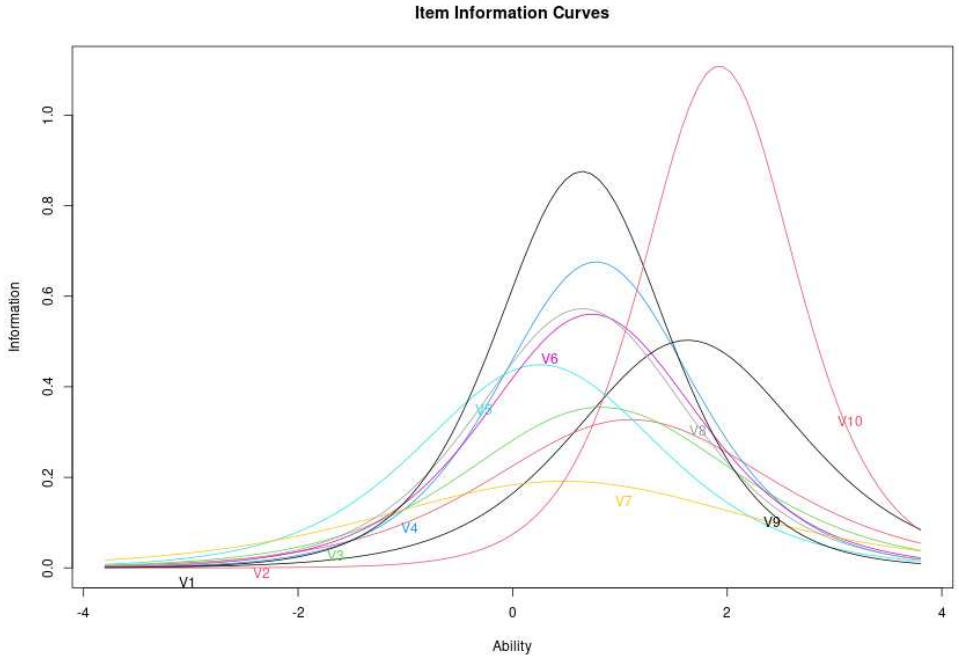
Item	Difficulty	Discrimination
V1	1.658	1.378
V2	0.982	1.378
V3	0.750	1.378
V4	0.850	1.378
V5	0.248	1.378
V6	0.768	1.378
V7	0.353	1.378
V8	0.679	1.378
V9	0.750	1.378
V10	2.398	1.378

Source: Author's own calculation

2.1.1.2 Polytomous response models

A polytomous response model (also called *nominal response model*) expands the dichotomous IRT case to allow more categories as a response choice in the IRT problem setup. The polytomous model, just like the dichotomous model, can also have 1, 2, or 3 parameters associated with the items.

Figure 7 – IIC for the 2PL model



Source: Author's own elaboration (based on QuantDev tutorial)

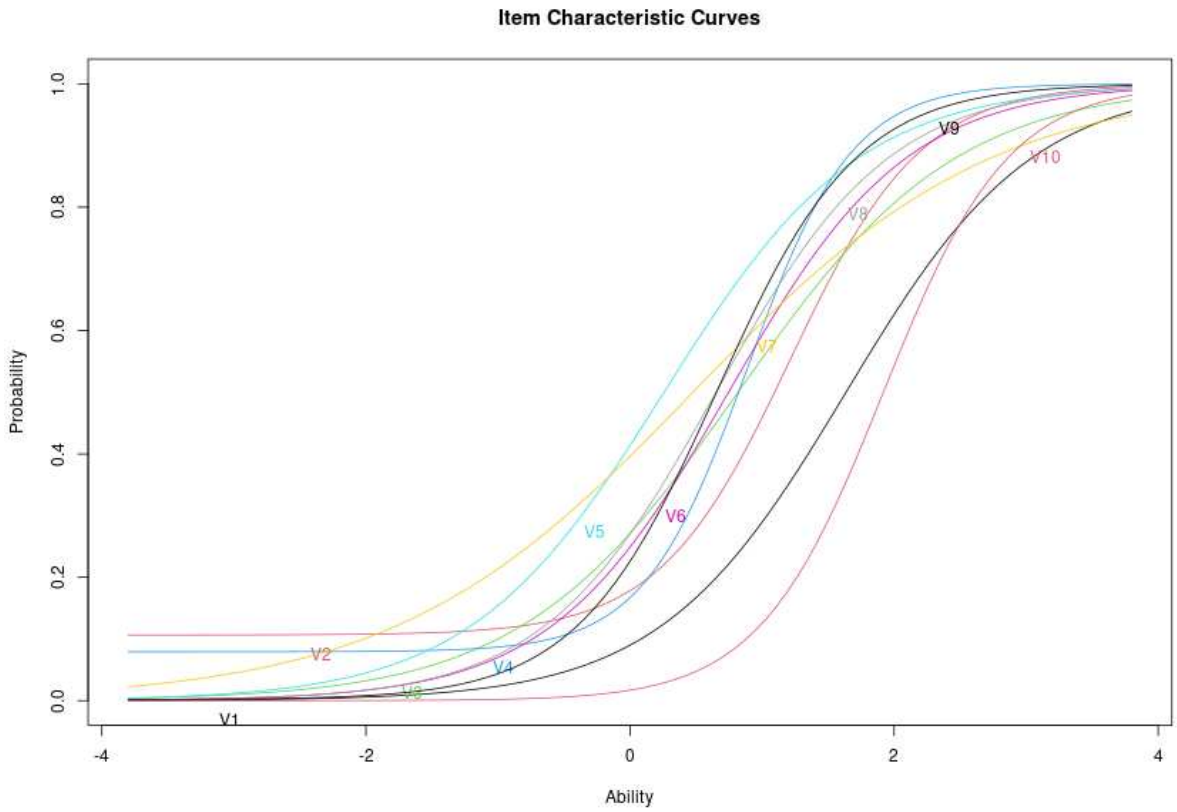
Table 2 – Item parameters for the 2PL model

Item	Difficulty	Discrimination
V1	1.632	1.418
V2	1.102	1.145
V3	0.821	1.191
V4	0.778	1.644
V5	0.255	1.340
V6	0.737	1.497
V7	0.477	0.875
V8	0.649	1.513
V9	0.649	1.872
V10	1.926	2.106

Source: Author's own calculation

Masters (1982) proposed the Partial Credit Model, Muraki (1992) proposed the Generalized Partial Credit Model and Samejima (1969), Samejima (1972) proposed the Graded Response Model, which is a polytomous response model where the categories have an intrinsic order. The polytomous response model can be thought of as a generalization of the dichotomous response model for an arbitrary number of categories; put it in a different way, the dichotomous response model can be thought of as a special case of the polytomous response model where

Figure 8 – ICC for the 3PL model



Source: Author's own elaboration (based on QuantDev tutorial)

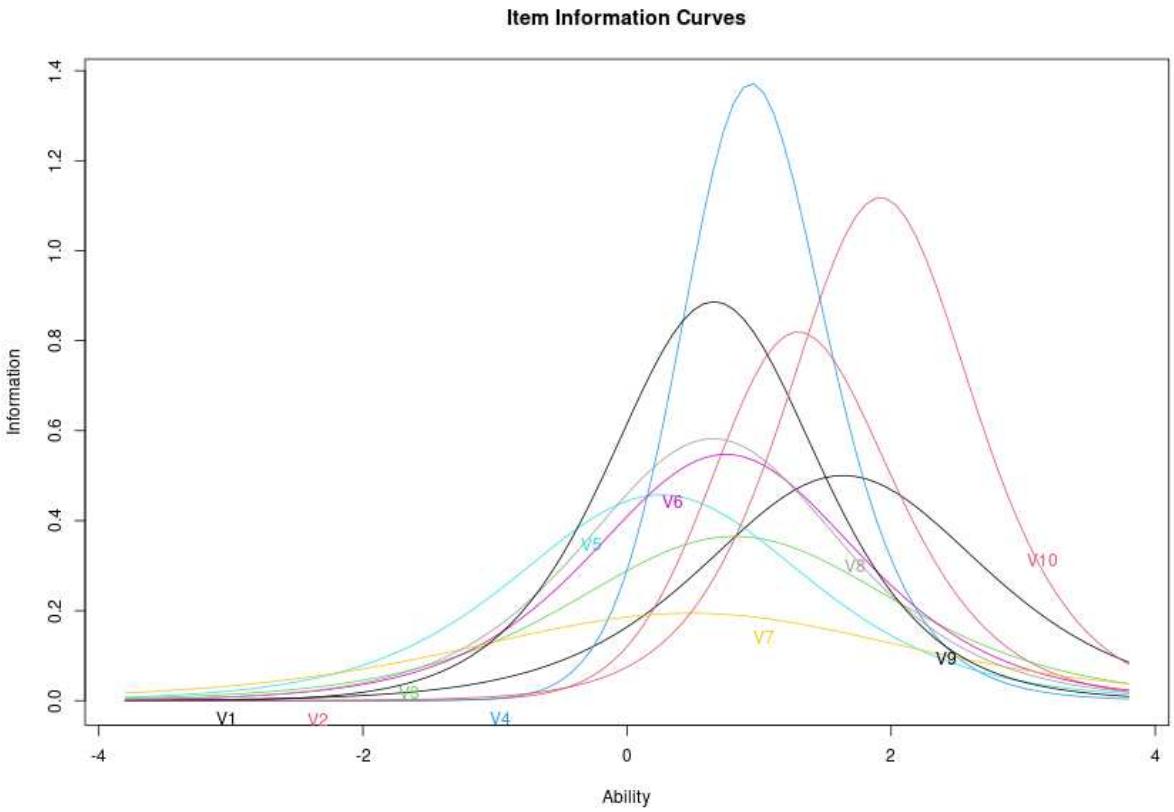
the number of categories is equal to two.

2.1.1.3 Graded response models

Samejima (1969), Samejima (1972) expanded the polytomous response model to make room for categories that have an intrinsic ordering among them, which was not possible to represent with the polytomous model. The Graded Response Model (GRM) is commonly used to model the agreement level of a respondent to a question; a common formulation is, for instance: strongly disagrees, disagrees, neutral, agrees, strongly agrees. The traditional polytomous model was not capable of mapping this type of relationship between the response categories.

Samejima (1972) also theorized about the cases where the graded response model would derive to a discrete or a continuous case, as the number of categories increased to infinity.

Figure 9 – IIC for the 3PL model



Source: Author's own elaboration (based on QuantDev tutorial)

2.1.1.4 Continuous response models

Although the polytomous and graded response models have much more power in the expressiveness of the possible responses that they can map, they are still not suitable to model continuous responses.

In (SAMEJIMA, 1972) the basic formulation for the continuous response model is presented to tackle the problem of dealing with free-response data in psychological evaluations. Samejima (1974) proposed a continuous response model derived from a limiting case from his graded response model, where the number of categories tend to infinite.

Noel and Dauvier (2007) proposed an IRT model that is parametrized using a Beta distribution, to model responses that are continuous but bounded in the interval $[0, 1]$. Chen et al. (2019) expanded this concept by creating a 2PL model where the ability and difficulty values are drawn from Beta distributions, and the ability and difficulty values lay in the interval $[0, 1]$ which makes it easier to interpret and compare the abilities of the respondents of an evaluated scenario. This model is capable of creating richer ICCs that are not logistic. They

Table 3 – Item parameters for the 3PL model

item	Difficulty	Discrimination	Guessing
V1	1.636	1.415	0.000
V2	1.214	2.007	0.106
V3	0.818	1.208	0.000
V4	0.890	2.530	0.079
V5	0.258	1.353	0.000
V6	0.747	1.480	0.000
V7	0.477	0.882	0.000
V8	0.652	1.526	0.000
V9	0.657	1.887	0.002
V10	1.919	2.115	0.000

Source: Author's own calculation

have also shown that this model is suitable to both use it in the traditional IRT application of human respondent evaluation as well as in the evaluation of classifier model in machine learning applications.

The β^3-IRT model is defined in Equation (2.1), where M is the number of respondents and N is the number of items, p_{ij} is the observed response of respondent i to item j, which is drawn from a Beta distribution. θ_i represents the ability of respondent i, δ_j is the difficulty of item j, and a_j is the discrimination of item j.

$$\begin{aligned} \mathsf{p}_{ij} &\sim \mathcal{B}(\alpha_{ij}, \beta_{ij}), \\ \alpha_{ij} &= \mathcal{F}_\alpha(\theta_i, \delta_j, a_j) = \left(\tfrac{\theta_i}{\delta_j}\right)^{a_j}, \\ \beta_{ij} &= \mathcal{F}_\beta(\theta_i, \delta_j, a_j) = (\tfrac{1-\theta_i}{1-\delta_j})^{a_j}, \\ \theta_i \sim \mathcal{B}(1,1), \delta_j &\sim \mathcal{B}(1,1), a_j \sim \mathcal{N}(1, \delta_0^2) \end{aligned} \tag{2.1}$$

The ICC is defined in Equation (2.2).

$$E[p_{ij}|\theta_i, \delta_j, a_j] = \frac{\alpha_{ij}}{\alpha_{ij} + \beta_{ij}} = \frac{1}{1 + (\frac{\delta_j}{1-\delta_j})^{a_j} + (\frac{\theta_i}{1-\theta_i})^{-a_j}} \tag{2.2}$$

2.1.2 IRT in Artificial Intelligence

In the past few years, IRT has also been used in the context of ML. It has been used to determine instance hardness, differentiate which instances are the most capable of discriminating between good and bad models, to evaluate NLP models, to evaluate classifier models

using continuous responses, to evaluate regression models, to evaluate speech synthesizers, to perform feature selection, to create an algorithm portfolio, among other applications.

One of the first attempts on understanding how IRT can be applied in ML problems was proposed by Martínez-Plumed et al. (2016). The IRT formulation maps classifier models as the respondents and instances of a dataset as the items. IRT can be a more powerful tool to evaluate models because it can differentiate different items in terms of difficulty and discrimination; a classifier that correctly predicts all easy instances, but none of the difficult ones, might be worse than one that correctly predicts most of the easier instances (but not all of them) and some of the difficult ones. A 2PL dichotomous IRT model is used in the experiments (guessing parameter set to 0). 128 classifiers from different families are used, in order to introduce variety in the respondents. Besides these "real" classifiers, artificial models are also used to serve as baselines and facilitate the interpretation of the results; the artificial models used include: random classifiers, majority and minority classifiers (always predict majority and minority class, respectively), and optimal and pessimal classifiers (always predict the correct and the wrong response, respectively). With this evaluation with a broad range of classifiers across several families, the results have shown that IRT can be used as a tool to evaluate the performance of classifiers on a dataset. The baseline artificial models used in the evaluation also present interesting findings, such as that the optimal classifier does not necessarily have the highest ability. This happens because the discrimination parameter, which is a trait of the instances of the dataset, is crucial to define the ability of a model. The difficulty and discrimination traits of the instances can be used to improve the classifiers. One of the main upsides of using IRT is understanding the item discrimination and its relationship with the abilities.

In Lalor, Wu and Yu (2016), IRT models are used to evaluate NLP systems and generate better test-sets for these evaluations. The proposal consists of applying IRT to the task of Recognizing Textual Entailment (RTE), where NLP systems are evaluated for their performance in correctly labelling sentences. In this case, the items are pairs of sentences to which humans or NLP systems assign a label, and the NLP systems are the respondents. The labels in the context of a RTE task can be described as follows. Consider a sentence pair consisting of (T, H), where T is a text and H is a hypothesis. T entails H if a human that has read T would infer that H is true (label=*entailment*); if a human would infer H is false, then it contradicts (label=*contradiction*); if the sentences are unrelated, then it is neutral (label=*neutral*). They have shown that IRT can be used to test NLP systems in the task of RTE with some good

insights, such as that high accuracy values do not always translate to high ability values. The IRT modelling also helps to select the items that are the most discriminant of the best NLP systems, by looking at the values of discrimination of the sentence pairs.

In Chen and Ahn (2020), a probabilistic framework is proposed to improve the accuracy of weighted majority voting classification algorithms using IRT to calculate the weights of the classifiers. The weights are assigned to classifiers based on their ability estimated by an IRT model. The IRT model is dichotomous, with the items being assigned to dataset instances, the respondents as the classifiers, and the response is binary and equal to 1 in case the classifier correctly predicts an instance and 0 otherwise. They use a 3PL dichotomous model that is parametrized in terms of difficulty, discrimination, and guessing parameters which are associated with items. The weight of a classifier is assigned as the normalized exponential value of ability for a given model divided by the sum of the exponential abilities of all models. They use the exponential value in order to have all values as nonnegative. To evaluate this proposal, they tested scenarios where the number of classifiers in the weighted ensemble varies, using datasets from the , which can have several ratios of number of features to number of instances. The IRT weighted model is compared to more traditional models such as random forest, Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), among others. The results have shown that the model that used IRT based weights has the best performance, and that the performance is better as the ensemble size increases.

Martínez-Plumed and Hernández-Orallo (2017) attempted to analyze the behavior of 40 learning techniques for Arcade Learning Environment (ALE), based on Atari 2600 games, which is a very popular general purpose AI benchmark. Only the techniques that use learning approaches (Reinforcement Learning (RL) or Deep Learning (DL)) are selected. A 2PL dichotomous IRT model is used to model this scenario, where the techniques are the respondents and the games are the items. First, the scores are normalized so that 0 means random behavior and 100 is equivalent to a human playing the game. Then, these scores are dichotomized, assigning 1 when the normalized score is above 100 and 0 otherwise, that is, the response is equal to 1 if the behavior is "human-like" and 0 if it looks random. The results have shown that some games might not be very useful distinguishing the best techniques from the rest. Interestingly, these were the games that had negative or low discrimination values. The ones with higher discrimination were the most relevant to distinguish the better techniques.

The application of IRT to feature selection was proposed by Kline et al. (2020). A 2PL dichotomous model is used to perform the task of feature selection in a binary classification

dataset from an intensive care unit with 2520 cases and 18 features. As the dataset is imbalanced, they undersample the majority class (non-death) so the number of instances matches the minority class (deaths). The IRT model parametrization assigns the features of the dataset as the items of the IRT problem; the ability values are not given much importance in their formulation. As the IRT model used in this task is dichotomous, the responses for each feature were dichotomized by their "unhealthiness level", where higher levels of unhealthiness are associated with results that are more divergent (class=*1*) from the normal laboratory values (class=*0*). To perform the feature selection with this IRT modelling, the features are evaluated in terms of their discrimination and difficulty. More discriminative features are considered to be the most relevant. This solution is then compared to traditional feature selection methods such as ridge and Recursive Feature Elimination (RFE), using a neural network classifier and the classifier accuracy to select the features. The results obtained with the IRT feature selection technique were similar to the ones obtained with the traditional models. The upside of using IRT for feature selection is that is does not depend on the outcome of a performance metric, so it is not directly linked to the accuracy of the model for the particular dataset.

Chen et al. (2019) propose a model that is an extension of the model proposed in Noel and Dauvier (2007). The original model did not have a discrimination parameter, that is, the ICC curves had a fixed slope and all items were assumed to have the same discrimination. This is an oversimplified assumption in most cases, and not necessarily a realistic assumption when dealing with ML models and datasets. Besides, the ability values are in the interval $[0, 1]$ which makes it easier to interpret and allows for non sigmoidal ICCs and have better predictive power than traditional logistic IRT models. When applied to classification models, the response is mapped to the probability of classifier *C* predicting the correct class *k* of instance *i*. This is a much richer information mapping than the traditional approaches to apply IRT to classification problems, which use dichotomous models. the probability gives a sense of how sure the classifier is about its prediction of an instance to each possible class. To show that this formulation can also be applied to the more traditional IRT applications, which aim to predict the abilities of human respondents, a case study with students responses is also provided.

The model was previously defined in Equation (2.1), and the ICCs in Equation (2.2)

For a classifier, the p_{ij} is defined in Equation (2.3), where I is the indicator function.

$$p_{ij} = \sum_{k} I(y_j = k) p_{ijk} \tag{2.3}$$

The investigation proposed by Martínez-Plumed et al. (2019) aims to understand how the IRT model can be used for classification problems. A set of classifiers is evaluated on 12 different datasets. Artificial classifiers are also added to the evaluation scenario to act as baselines. Several configurations of IRT models are also evaluated (dichotomous imbalanced 2PL, imbalanced 3PL, and balanced 3PL). The classifiers are the respondents and the instances of a dataset are the items of the IRT problem. The experiments show that a classifier's ability is not measured solely based on the number of correct answers, but rather on its responses for discriminatory items, with different levels of difficulty. Also, the items with negative discrimination can affect the abilities of the models. The choice of using items with negative discrimination is up to the practitioner, depending on what is the goal of an evaluation, as removing these items with negative discrimination can make for more robust ability values. On the other hand, items with higher values of discrimination (steeper slopes) seem to be the ones that are more capable of differentiating the best models from the rest. The experiments show that IRT can be used to improve models, construct algorithm portfolios (based on their performance with difficult and discriminant instances), classifier selection in deployment, better understanding of datasets, and adaptive testing by selecting only the most discriminant items to minimize the number of instances required to test the performance of a given model.

In order to deal with nonnegative responses, Moraes et al. (2020) proposed the Γ-IRT, which is a modification of the β^3-IRT, to model nonnegative unbounded responses. These can be student grades on open-ended questions or the absolute error of regression models. The use of nonnegative responses means that this model is not suitable to map the regression prediction directly, because this prediction can also have a negative value, depending on the context and the dataset being evaluated; instead, it needs to map the response to an absolute value of error for the predicted regression value. The errors are modeled as random variables following the gamma distribution, with support in $[0, \infty]$. This IRT model is parametrized in terms of item difficulty and discrimination and model ability, similarly to most works that use IRT in the context of algorithm evaluation in ML. The authors have also demonstrated that their proposed IRT modelling can be applied with good results both to evaluating regression problems in ML and to the traditional IRT application in Psychometrics.

Kandanaarachchi and Smith-Miles (2020) proposed a framework in which the traditional mapping of IRT to ML problems is inverted. Usually, the prediction model is mapped as the problem's respondent and the instances of a dataset are mapped as the items of the problem. In this inverted setting, the datasets are the respondents and the prediction algorithms are

the items of the problem. Their proposed solution is called Algorithmic Item-Response Theory (AIRT). The IRT latent traits in this scenario will result in difficulty and discrimination traits for the algorithms and ability traits for the datasets. This might seem counter intuitive, but their explanation to this mapping is that the traits of the algorithms could be interpreted as the easiness threshold (difficulty) and anomalousness and stability (discrimination), and the easiness (ability) trait for datasets. They have shown that IRT can also be used in this inverted mapping to create useful algorithm portfolios of models and datasets.

Oliveira, Tenório and Prudêncio (2020) proposed the use of IRT to evaluate speech synthesizers. The β^3-IRT formulation proposed in Chen et al. (2019) is used to evaluate 62 speakers and 12 sentences. The speakers are the respondents, and the sentences to be synthesized are the items of the IRT problem. The response is the transcription accuracy for a given speaker and sentence, which lies in the interval $[0, 1]$, where 1 means the best response. The goal to use IRT is to predict the difficulty of each sentence and how discriminant they are in separating the good speakers from the bad ones. Speakers with higher abilities are expected to have had higher accuracy values for the most difficult sentences. Indeed, the results show that there is a correlation of 0.9 between the abilities and the accuracy values. The results have demonstrated that IRT can be used to both evaluate the best speech synthesizer techniques as well as which are the best sentences to distinguish the best synthesizers from the bad ones. This can help to create better speech test sets, by keeping the sentences with higher discrimination power.

2.2 MISSING DATA

Rubin (1976) classifies missing data focusing on the mechanisms of missing data, that is, the mechanisms that made the data to be missing, based on relationships between the missing data and observable variables. These missing data mechanisms can depend on: the variable with the missing data itself, associated variables, and the mechanism underlying the missing data.

In McKnight et al. (2007), a very simple explanation of the types of missing data is provided:

- Missing Completely At Random (MCAR): the mechanism through which the data is missing is completely random. When the data is MCAR, there is no relationship between the mechanism and either the missing data or the observable data.

- Missing At Random (MAR): the mechanism through which the data is missing might depend on another variable, but not on the variable with missing data itself. When the data is MAR, there is no relationship between the mechanism and the missing data, but there is a relationship between the mechanism and the observable data.
- Missing Not At Random (MNAR): the mechanism through which the data is missing depends on the missing variable itself. For the data to be MNAR, there has to be a relationship through which the data is missing and the missing data itself.

Given these classification types of missing data, one important aspect to be mentioned is that in real life it might not be possible to differentiate MAR from MNAR data. The reason for this is that, because the data is missing, it is not possible to analyze the mechanism through which the data is missing in case the data is MNAR, so it could appear that the data is just MAR.

In McKnight et al. (2007) a cookbook that goes in detail about several aspects of missing data is provided. One of the most important aspects is how to deal with missing data; the possibilities can range from data deletion to data imputation. Focusing on data imputation, we will go into further detail about this topic in the next section.

2.3 MISSING DATA IMPUTATION

According to Buuren and Groothuis-Oudshoorn (2011), ideally, a good imputation model should:

- Account for the process that created the missing data.
- Preserve data relations intra and inter-variables.
- Preserve the uncertainty about the data relations.

McKnight et al. (2007) define the different approaches when dealing with missing data imputation: single imputation and multiple imputation.

2.3.1 Single Imputation

For the single imputation, it can be defined in terms of three different categories: constant, randomly selected value, nonrandomly derived value.

- Constant

  Examples of constant imputation can be zero imputation, and mean or median imputation. Zero imputation is usually used when the data is missing by choice or design. Imputing missing values by their mean is usually a poor choice of imputation technique, albeit being the simplest one, it can significantly change the variance of the data and essentially erase the extreme values from the data.

- Randomly selected value

  The techniques of hot deck and cold deck set the imputed value to a value that is present in the training data and a value that is present in similar data, respectively. The selected value is chosen at random. This is also not considered a good approach when imputing missing data as it does not take any relationships among the variables into account, just imputes a random value that is found in a different instance of the dataset.

- Nonrandomly derived value

  One example of single imputation nonrandomly selected is using a regression technique to impute missing data. While this is probably the best option when using single imputation, because it imputes the value a single time, it is less effective than approaches that use multiple imputation.

The problem with single imputation is that it introduces substantial bias into the data, while also messing up with the original distribution of the data and inter-variable relationships. Single imputation, especially if there are large amounts of missing data, might raise more problems than it solves.

2.3.2 Multiple Imputation

Multiple Imputation (MI) techniques are the best option when missing data needs to be imputed. MI do not input a single value for the missing datum, but rather, a set of K different

estimated values (usually K is a value between 3 and 10) to fill the missing data. This is an iterative process that uses the observable data to fill in the gaps to produce new "complete datasets". The set of estimated values are combined to find the best possible missing data filler for each datum that is missing.

One of the most well known examples of multiple imputation is Multivariate Imputation by Chained Equations (MICE) which was proposed by Buuren et al. (2006), based on the ideas initially discussed by Rubin (1976) to impute missing data iteratively. This model can handle both MAR and MNAR data, which are the trickiest types of missing data.

MICE (BUUREN; GROOTHUIS-OUDSHOORN, 2011) is an implementation of the multiple imputation that was proposed in (BUUREN et al., 2006), using the Fully Conditional Specification (FCS), also known as MICE. This type of imputation is much more powerful when dealing with missing data, and has lower standard errors when compared to single imputation. The model can deal with multivariate imputation (missing data imputed for several variables at the same time), and also with univariate imputation (missing data belong to a single missing feature).

Although the multiple imputation techniques give a better estimation of the standard errors when compared to the single imputation techniques, they are not without fault. Of course, if the data is missing because of a human choice or intervention, it might be worth it to collect the missing data to fill the gaps. The analysis of whether this is worth it or not could include a cost sensitive analysis.

2.4 FEATURE GATHERING

It is not unusual, in datasets, that missing data is not actually missing, but rather it is information that was not collected by option of a human designer. We call *feature gathering* the action of collecting information that is missing, if that information is possible to be obtained. The gathering of information has an associated cost; this cost might depend on each instance, or have a fixed value for any instance.

2.5 FEATURE SELECTION AND FEATURE RANKING

Miao and Niu (2016) classify feature selection techniques in terms of *label information* and in terms of *search strategy*. Considering label information, a technique can be supervised,

semi-supervised, or unsupervised. The most common are the supervised techniques, where the data is labeled for all instances of the dataset. The semi-supervised technique can use the labeled data to explain the unlabeled data. The unsupervised problem is much harder, and it usually goes in the sense of choosing features that better preserve the manifold structure of the original data. In terms of search strategy, there are three main categories: filter models, wrapper models, and embedded models.

Chandrashekar and Sahin (2014) conducted a survey on the main proposed solutions for the task of feature selection in terms of search strategy. These techniques are briefly described below. Ideally, for a feature to be considered relevant it needs to contain information about different classes in the data and be able to discriminate the different classes. That is, a feature is not relevant if it is conditionally independent of the target classes.

- Filter models

  Filter-based models are the simpler methods for feature selection. In this type of model, features are selected using variable ranking techniques. One example of filter-based methods are the correlation of the variables to the output labels, which can detect linear dependencies between features and targets. To produce a feature ranking, the features can be ordered in descending order based on their correlation value to the targets; to select a set of features, a threshold value t can be used to cut out the features whose correlation to the targets is lower than t. Another example is the Mutual Information (MI), where the MI values can be calculated for each feature related to the target variables; however, this does not take any relationships inter-features into account. The main advantages of filter-based methods is that they are light and do not rely on learning algorithms, which can introduce bias. The drawbacks are that these methods do not consider the relationships between the variables, and for this reason, can select a redundant subset of features. Besides, as the split between the selected features and the non-selected features is a threshold, there is no clear way on how to choose the best number of features to select.

- Wrapper models

  Wrapper methods rely on learning models to select the features. they use the performance measure of the predictor as the objective function to be optimized. Subsets of features are found heuristically. exhaustive search methods are too computationally expensive,

so search heuristics like genetic algorithms or Particle Swarm Optimization (PSO) can be used to reduce the search space. The sequential search starts with an empty set and adds features; the heuristic search evaluates different sets to try to optimize the objective function. The main drawback of wrapper methods is that they are usually computationally expensive because they need to perform several evaluations on several feature subsets in order to find the one that optimizes the objective function of the problem.

- Embedded models

  Embedded models try to reduce the computational time of wrapper models while providing less redundant subsets than the filter models. To do this, they incorporate the feature selection as a part of the training. MI can be used to maximize the MI between a feature to be added to the subset and minimize the MI between the features that are in the subset. Minimum Redundancy Maximum Relevance (MRMR) is an example of model that applies this technique. As the model's name itself says, it tries to keep minimum redundancy and maximum feature relevance at the same time, in order to select a set of features that is relevant but not redundant. RFE is another example of an embedded model. Instead of adding features recursively, it starts with the complete set of features and recursively removes them.

- Other techniques

  Besides the models presented, there are also other methods to perform feature selection, such as using a classifier to understand feature relevance (such as Radial Basis Function (RBF) or tree-based models which output a feature importance value for each feature); Principal Component Analysis (PCA) has also been shown to perform feature selection in the sense that it can keep a percentage of data variance for K principal components that can be directly mapped to the dataset features.

  PCA is originally a dimensionality reduction algorithm, but it can also be used to select the most relevant features of a data set (SONG; GUO; MEI, 2010). One of the drawbacks of PCA is that it can only capture linear relationships between the variables and the results can be opaque to analyze. Although many researchers have been using PCA for this task, there is controversy about whether PCA should be used to perform these tasks.

Boruta (KURSA; RUDNICKI, 2010), which is available as R and Python packages, is a tool that claims to understand the importance of each feature by removing each feature — one at a time — and analyzing whether or not the metric being evaluated changed (by a certain amount) in the absence of that feature. Since most models do not allow training and testing in different sets of features, Boruta transforms the feature being evaluated into noise at test time - so it should not add any relevant information to the dataset. One of the main constraints of Boruta is that it depends on a tree-based model to perform the evaluation.

MRMR (DING; PENG, 2003) was initially proposed as a method for feature selection applied to gene expression data, which is a high dimensionality domain that can leverage a lower dimensionality domain that offers a simpler and more explainable problem. Nowadays, this algorithm is widely used in the industry, specially because it is much faster than some of its counterparts.

None of the solutions presented in this section are model-agnostic; all of them depend on a single base model that is used to evaluate the results of changing one or a small set of the available features.

Because they are based on a single model, these techniques will lead to solutions that are biased on the model used to perform the evaluations. Tree-based models, linear models, neural networks — all these algorithms might present different evaluations for a single dataset, and using only one of them to select which are the best features of a dataset is a biased analysis.

One possible solution to this is to use a set of different models to evaluate the relevance of a feature. But how can the models decide a group solution? IRT presents a framework that can calculate how important each model is, for each feature, and each instance of the dataset. In the next chapter we will explain the solution with further detail.

2.6 FINAL CONSIDERATIONS

Feature selection is an important process to remove unnecessary or noisy features from a dataset. Sometimes features can introduce noise instead of contributing to describe the data.

Dealing with missing information is always a complex task. Understanding whether it is worth it to expend resources to gather the missing information is a tough decision to make.

In the past few years, several efforts have been made in applying IRT in the context of artificial intelligence and machine learning, in several different fields. The previous works have shown that IRT models can help providing a solution that evaluates a set of classifiers in a given dataset. However, one problem that has not yet been tackled is the use of IRT to perform feature gathering on an instance based level. The solution proposed in this dissertation aims to help solving this problem.

3 β^3-IRT FOR FEATURE IMPORTANCE ANALYSIS

To the best of our knowledge, using IRT in the context of feature gathering has not yet been tackled. In this chapter we propose β^3-IRT for Feature Importance (BIFI), a workflow to estimate the feature importance of a dataset at test time.

This chapter is organized as follows. Section 3.1 describes the IRT workflow used to analyze the feature relevance on a global and local scopes. In Section 3.2, a methodology to analyze these results is proposed.

3.1 PROPOSAL

In the previous chapter, we discussed that when a classifier is deployed to predict the class label of an instance, some input features may be missing. If the cost of acquiring the true feature value is high, it may be better to simply adopt imputed values. Deciding whether to acquire the feature value or not also depends on the relevance of the feature to provide an accurate prediction for that instance. Measuring feature relevance is commonplace in the ML literature but few previous works have investigated this problem in the context of missing information.

In this work, an original solution is proposed to measure feature relevance in the context of missing information. In BIFI, IRT is employed to measure the impact of a missing feature in classifier performance and instance difficulty. IRT has been adopted as a robust procedure to evaluate classifiers by considering that some instances may be more difficult to be classified. In this work, we extended the use of IRT to measure feature relevance by considering that important features are more relevant to provide better predictions for more difficult instances. Additionally, we highlight that IRT adopts a set of classifiers instead of a single learning model, which makes the method more robust to weaknesses directly related to a particular type of model. The difficulty of each instance is related to the overall performance of all classifiers, not just a single one.

Given a pool of classification algorithms and a dataset of interest, the BIFI workflow consists of three main steps, as shown in Algorithm 1. Initially, in Part I, the classifiers are trained and tested using all features (i.e., without any missing features). The prediction of each classifier on each instance is recorded in order to produce a response matrix. Next, in Part II, the β^3-IRT

algorithm is run by using as input the response matrix built in Part I. This step results in two main outputs: the ability of each classifier and the difficulty of each instance.

Finally in Part III, the classifiers trained in Part I (i.e., trained without missing features) are tested in a scenario with missing features. For each feature f, we have a different missing feature scenario. The objective is to measure the impact of the feature on the instances' difficulties and classifiers' abilities if that feature is eventually not available at deployment time. So if a dataset has a total of N features, the number of evaluation scenarios is $N + 1$, which corresponds to a scenario without any missing features (which we call *original scenario*), plus one scenario per feature f, with values from feature f being imputed in the test set.

For each feature, a new version of the classification dataset is produced by excluding the original feature values and replacing them by imputed values. The classifiers are evaluated again and a new response matrix is produced for the IRT model. As a result, new values of difficulties and abilities are estimated. It is expected that classifiers' abilities are harmed when a relevant feature is missing. Additionally, instances' difficulties tend to increase. Feature relevance in BIFI is then measured taking into account the change in difficulties and abilities when the classifiers are evaluated with and without the feature.

Algorithm 1: BIFI Workflow

Input: Classification dataset (with N instances and P features) and a pool of M classification algorithms

Part I: Classifiers' Evaluation

1. Evaluate classifiers on the classification dataset, using all available features
2. Create a $M \times N$ response matrix R for all classifiers and instances

Part II: IRT results

1. Apply IRT to estimate each instance's difficulty δ_i and each classifier's ability θ_m

Part III: Missing feature loop

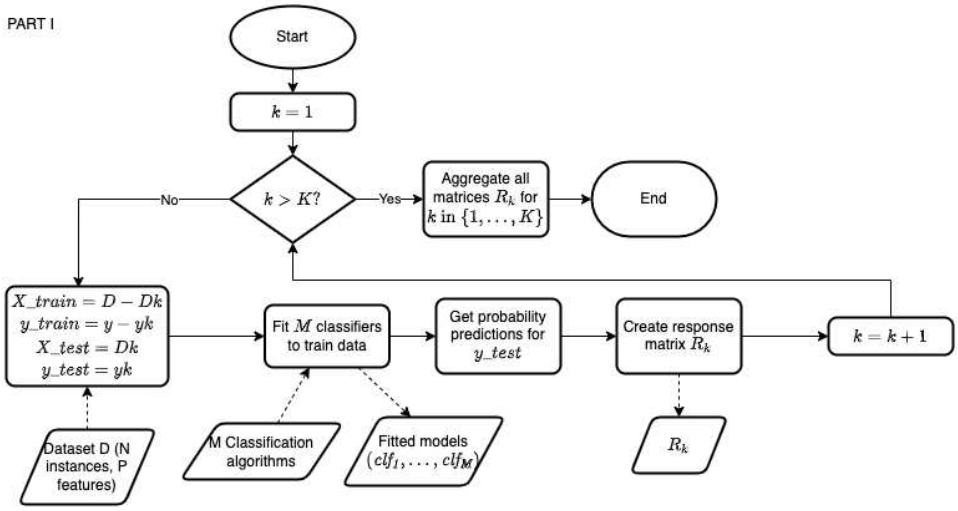
for *each feature* $f \in \{1 \dots P\}$ **do**

1. Evaluate the classifiers on the classification dataset, but excluding the original feature values and replacing them by imputed ones
2. Create a response matrix R^f, with the new classifiers' responses
3. Apply IRT (as in Part II) to estimate each instance's difficulty δ_i^f and each classifier's ability θ_m^f
4. Compute the impact measures of feature f by comparing instances' difficulties and classifiers' abilities in both scenarios: with and without feature f

end

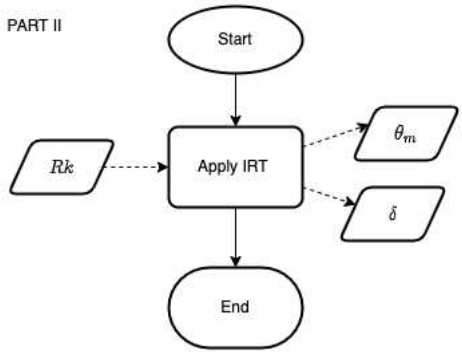
Output: Impact measures for each feature based on the corresponding changes on instances' difficulties and classifiers' abilities

Figure 10 – Diagram for BIFI Workflow: Part I



Source: Author's own elaboration

Figure 11 – Diagram for BIFI Workflow: Part II

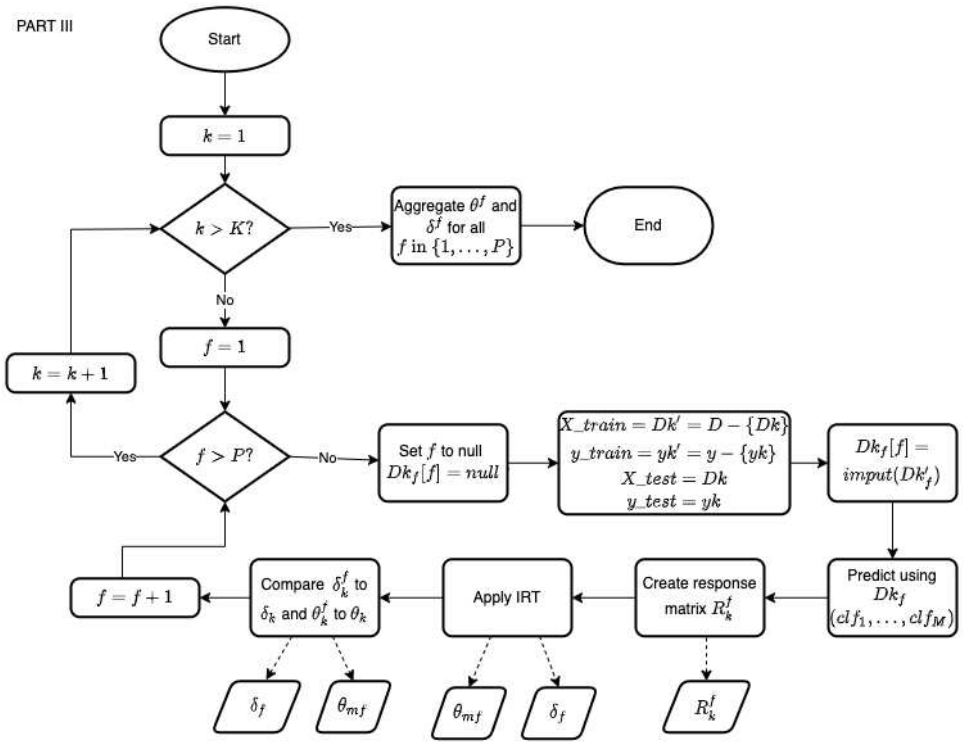


Source: Author's own elaboration

In order to provide a better visualization of the workflow described in Algorithm 1, we provide three diagrams to visually describe the three parts that compose the workflow. Figure 10 shows the diagram for the Part I of Algorithm 1, where the data is split into train and test sets, the classification models are trained, and the response matrices R_k for the scenario with the complete dataset are generated. Figure 11 shows the diagram for Part II, where the IRT models are trained using the response matrices outputted by Part I, which generate ability, discrimination, and difficulty values for the scenario with complete data (*original scenario*). Finally, Figure 12 presents the diagram for Part III, where the data of the test set is imputed, one feature f at a time, then evaluated by the classification models trained in Part I (clf_m), which will provide the corresponding response matrices R_k^f that will be then used as input of the IRT model to calculate the ability, discrimination, and difficulty values for each missing feature scenario.

None of the previous studies considered using an instance-level approach to determine the

Figure 12 – Diagram for BIFI Workflow: Part III



Source: Author's own elaboration

relevance of each feature, thus constructing a feature ranking for a given dataset according to the change in the abilities of the classifiers. Additionally, BIFI can be used for feature gathering at deployment time, i.e., one can decide if gathering a missing feature for an input instance is really necessary, based on the impact on difficulty in similar instances in the training set. Finally, the proposed solution can be applied for algorithm selection to select the more suitable algorithms in case it is not possible to gather a missing feature value or simply the feature is not relevant.

3.1.1 Data cleaning

The first step of the algorithm, before any model execution, is to clean and prepare the data for execution.

The main steps of data cleaning are the following:

1. Remove any null entries from the data
2. Infer feature type, based on the type of the variable and the number of different values for that feature in the whole dataset

3. Transform categorical data into one-hot-encoding categories

The inference of the feature type is used to select the type of imputation, when using Simple Imputer. If the feature is inferred to be numerical, the mean is used for imputation; if the feature is inferred to be integer, the median is used for imputation; finally, if the feature is inferred to be categorical, the mode is used for imputation.

3.1.2 Classifiers' Evaluation

In the proposed solution, initially M different classification algorithms are evaluated on the classification dataset given as input. As mentioned in the previous section, in this step all features in the dataset are used during classifier training and evaluation.

Each classification algorithm is evaluated on the classification dataset by following a k-fold cross-validation experiment. In each cross-validation step, the dataset is split into training and test set. Each classifier m is trained by using the training set and then applied to calculate the response $R(m, i)$ of each instance i in the test set.

$$R(m, i) = \sum_{k} \mathbb{I}(y_i = k) p_{mik} \tag{3.1}$$

where $\mathbb{I}$ is the indicator function.

In order to obtain classifier models with better parameters according to each dataset evaluated, a grid-search of parameters is executed for each classifier, using the training set. Hence, we aim to minimize the effect of poorly built classifiers in the estimation of difficulties for the instances.

3.1.3 IRT Results

The β^3-IRT model receives the response matrix R as input and then estimates the ability of each classifier m (respondent) and the values of difficulty and discrimination for each instance i (item). The classifiers are evaluated based on their ability values, which are in the interval $(0, 1)$, with higher abilities representing better classifiers. Although instance discrimination is an output of the IRT process, we chose to focus on using only instance difficulty to evaluate the instances. Difficulty values are also in the interval $(0, 1)$, which is intuitive to interpret, as higher difficulty values mean instances are considered more difficult.

3.1.4 Missing feature loop

The goal of this step of the algorithm is to obtain the IRT data needed to compare the results of each missing feature scenario to results of the original dataset. We want to asses how much impact the missing data in the test set can have on the overall performance of the models (which in IRT is given by classifier ability), as well as the changes in difficulty for each instance.

The main tasks performed in this step of the algorithm, for each feature of the dataset, are:

- Evaluate the trained classifiers in a simulated missing data scenario. This is similar to Part I of Algorithm 1, but the feature values are excluded in the validation set and replaced by imputed ones;
- Produce the response matrix R^f, with the responses provided by the classifiers to the instances modified in the validation set;
- Apply the β^3-IRT model given the matrix R^f as input. If the feature is relevant we expect difficulties to increase compared to the original scenario, since imputed values may be very different from the original ones. Classifiers' abilities tend be lower on the other hand;
- Compare the IRT results with imputed data and with the original, complete data. This is done by computing the impact measures described below.

The IRT model results in two main outputs: instance difficulties and classifier ability values. In order to assess the impact of missing features, two corresponding metrics are proposed: difficulty change and ability change.

Given a feature f, difficulty change is defined as the average difference between difficulties with and without the feature over all instances, as defined in Equation (3.2).

$$\Delta\delta^f = diff\_change(f) = \frac{1}{N}\sum_{i=1}^{N}(\delta_i - \delta_i^f), \tag{3.2}$$

in which N is the number of instances.

Similarly, ability change can be aggregated for all models in a missing feature scenario. Given a feature f, ability differences are averaged over all models. as defined in Equation (3.3).

$$\Delta\theta^f = ability\_change(f) = \frac{1}{M}\sum_{m=1}^{M}(\theta_m^f - \theta_m), \tag{3.3}$$

in which M is the number of classifiers.

3.2 REGRESSION ANALYSIS

In order to understand how the difficulty changes in different scenarios, we run a regression task that takes as input the difficulty values of the original data and the corresponding difficulties of each missing feature scenario.

To achieve this, we perform a regression task using Ordinary Least Squares (OLS), which provides a prediction interval and a confidence interval, which are used to understand the behavior of the evaluated metric and the outliers. We use the *statsmodels* (SEABOLD; PERKTOLD, 2010) package in Python to calculate the OLS and the corresponding intervals for each scenario.

The complete analysis process is divided into three main parts: OLS regression, data binarization, and classification analysis. Each of these steps will be explained in more detail in the following sections.

3.2.1 OLS Regression

The first step of the algorithm is to perform a regression task with the missing difficulties as the data to be predicted (y) and the difficulties of the original data as the input values of the prediction (X). The regression task data are described in Table 4.

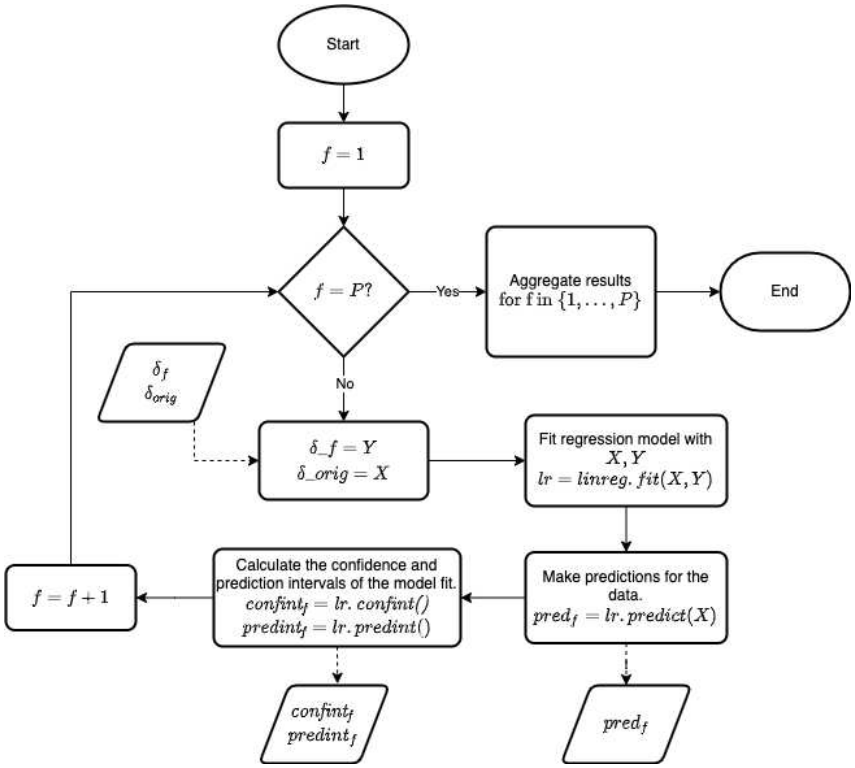
Table 4 – Description of the OLS input data

Data	**Description**
X_train	original difficulties
y_train	difficulties for missing data scenario
X_val	equal to X_train
y_val	equal to y_train

Source: Author's own elaboration

The main goal of this regression task is to fit a linear model using all the difficulty values of the scenario without any missing data and analyze how well this linear model can fit the

Figure 13 – Diagram for Algorithm 2



Source: Author's own elaboration

difficulties for each missing feature scenario. With this information at hand, we can analyze how much the difficulty value for instance i in missing feature scenario f differs from the prediction value. Besides the prediction, the confidence interval and the prediction interval are also calculated. The idea is to build a pipeline that allows us to decide, for new instances during deployment, if a feature is going to be relevant for that instance or not. The Algorithm 2 depicts this step in more detail, while Figure 13 shows the diagram representing Algorithm 2.

Algorithm 2: Regression analysis pipeline

Input: Difficulties for whole data and missing feature scenarios

Let diff_orig be the vector with the difficulties in the scenario without any missing features.

for *each feature* $f \in \{1 \ldots P\}$ **do**

1. Let diff_f be the vector with the difficulties in the scenario that f is missing;
2. Use difficulty data of the original scenario as the X and the difficulty of the missing feature scenario as the Y of the regression problem;
3. Calculate the confidence and prediction intervals for the regression problem;

end

Output: The confidence and prediction intervals for each missing feature scenario f

3.2.2 Data Binarization

The regression task described in the previous section not only outputs the predictions, but also the upper and lower *confidence* and *prediction* intervals. These intervals can be used to turn these predictions into binary labels so that classifier models can be trained on the original data to evaluate how much our solution is capable of predicting the impact of missing features for each instance. For each instance in the regression task, the prediction data obtained with the OLS regression is binarized in two different labels: 1 for when the difficulty of a given instance is above the prediction interval, and 0 otherwise.

Equation (3.4) shows how we binarize the output for instance i.

$$label(i, f) = \begin{cases} 1 & \text{if } \delta(i, f) < lower\_prediction\_interval(i, f) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The Algorithm 3 depicts this step in detail, and Figure 14 presents the diagram for the Data Binarization step.

Algorithm 3: Data binarization

Input: The confidence and prediction intervals of difficulty values for each missing feature scenario f

for *each feature* $f \in \{1 \dots P\}$ **do**

1. Let $pred(f, i)$ be the prediction of the difficulty value of instance i in the missing feature scenario for feature f;
2. Binarize the prediction $pred(f, i)$ using Equation (3.4);
3. Let $bin\_pred(f, i)$ be the binarized value for instance i and feature f;

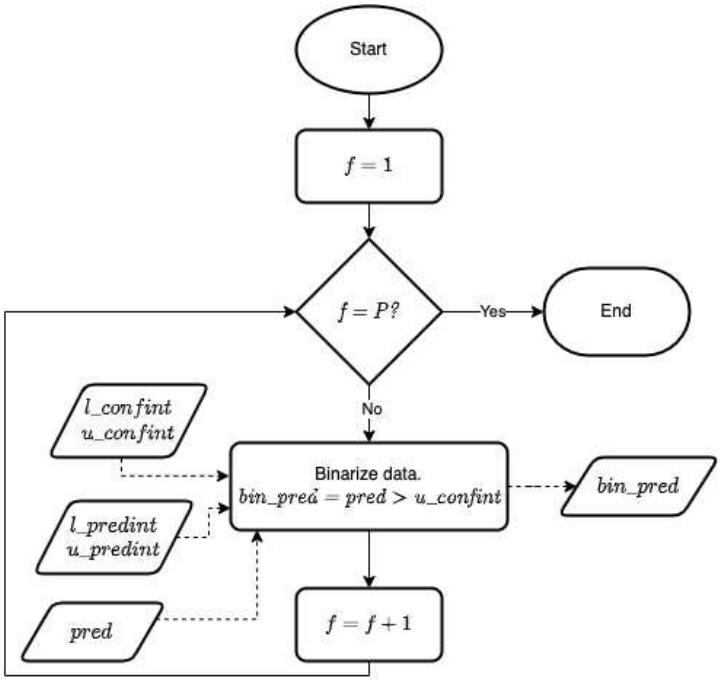
end

Output: Binary labels that indicate whether each instance is above the selected interval or not

3.2.3 Classification analysis

After getting binary labels for all instances of the dataset, we want to build a classification task to check whether IRT results are capable of differentiating which instances are the ones more impacted by the missing of each feature of the dataset. With this setup, we can infer which feature's missingness has the most impact on which instances. Algorithm 4 depicts this step in detail. Figure 15 shows the diagram of the algorithm described in Algorithm 4.

Figure 14 – Diagram for Algorithm 3



Source: Author's own elaboration

In this step, for each missing feature scenario of a given dataset D, we evaluate the performance of binary classifiers under scenarios where the data is composed by the complete data of dataset D, excluding the feature for training and test. That is, the performance of the models is evaluated in the scenario where the feature is completely gone from the data. The target values are the binarized as described in Section 3.2.2.

Chapter 4 provides details about the setup for the IRT and the OLS regression analysis described in this chapter. The experimental results are shown in Chapter 5.

Algorithm 4: Classification analysis

Input: dataset *D*, binarized labels for each scenario of dataset *D*

for *each feature* $f \in \{1 \dots P\}$ **do**

1. Get the complete data for dataset D;
2. Remove the feature f from the dataset: $D_f = D - f$;
3. Use data from D_f as features and the binarized $pred(f, i)$ as labels in a binary classification problem. That is, $X = D_f$, $y = bin(pred(f, i)$;
4. Split the data into 5 stratified cross-validation folds;

for *each fold in folds* **do**

5.1 Train classifier models (listed in Section 4.2) on all folds except <fold>;
5.2 Use classifier models to make predictions on fold <fold>;
5.3 Use the prediction probabilities to select the best prediction threshold for each classifier;

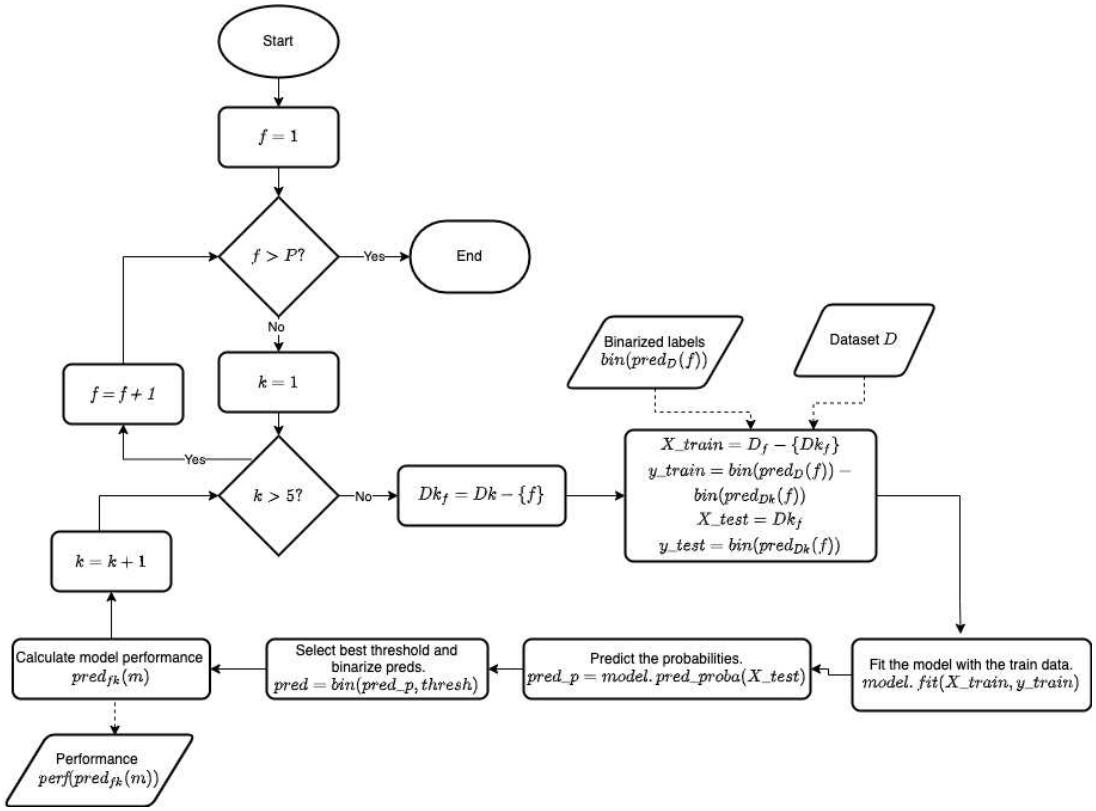
end

6. Aggregate the predictions over all folds for each classifier and each optimal prediction threshold;
7. Calculate classifier performance metrics for each configuration;
8. Compare the results to a random classifier model;

end

Output: Binary classification performance on each missing feature scenario for dataset *D*

Figure 15 – Diagram for Algorithm 4



Source: Author's own elaboration

4 EXPERIMENTAL SETUP

In this section we will describe the setup for the experimental evaluation of the solution proposed in the present work.

To run the experiments with the β^3-IRT model, we use the most recent implementation of this model, which is available at: <https://github.com/Manuelfjr/birt-gd/>. Because of restrictions imposed by the Beta distribution, which can take values in the interval $(0, 1)$ (that is, the values cannot be equal to either 0 or 1 as the interval is open), and the limitations of computer programming languages, we had to make some adaptations to the model. As the response that we are modelling is a probability, this value in theory could have both the values of 0 or 1, in some cases; when this happens, we use a small value ϵ to make the value very close to the desired number but not equal so it does not cause any problems with our formulation.

The experimental setup and configuration we used for running the experiments can be analyzed in terms of the following topics:

- Datasets
- Classifiers
- Feature selection methods used to compare
- Imputation techniques
- IRT model
- OLS regression

This chapter is organized as follows. Section 4.1 gives more detail about the datasets used in the experimental analysis. Section 4.2 lists the classifiers used as respondents in the IRT modelling of the problem. Section 4.3 presents the feature selection techniques used to create a feature ranking, which are used as baseline to compare to a feature ranking created using IRT results. Section 4.4 gives more detail about the imputation techniques used to fill the missing data. Section 4.5 describes the IRT formulation. Section 4.6 gives more detail about the OLS regression task (previously described in Section 3.2), such as the models used, and how we will evaluate and analyze the results of this task.

4.1 DATASETS

To evaluate our solution, we used a set of classification datasets from the UCI Machine Learning Repository (DUA; GRAFF, 2017). A quick description of each dataset is provided below, gathered from each respective dataset page on the UCI website.

- Contraceptive: The goal is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socioeconomic characteristics.
- Pima-diabetes: The goal is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Wheat seeds: The goal is to predict the variety of wheat seed based on geometric characteristics (area, perimeter, etc.).
- Haberman: The goal is to predict the survival status (patient survived 5 years or longer after surgery) of patients who had undergone surgery for breast cancer based on a very limited number of features.
- Cleveland heart-disease: The goal is to predict whether a patient has a heart disease or not, based on a subset of fourteen different characteristics from each patient.
- Ionosphere: The goal is to to predict some type of structure in the ionosphere based on signals received by high frequency antennas.
- Wine: The goal is to predict the type of a wine based on thirteen constituents found in each of these which, which are cultivated in the same region in Italy.
- Yeast: The goal is to predict the class protein localization of yeast data.

The datasets used in the evaluation are described in table 5. This table shows the original number of features and instances of each dataset.

Figure 16 shows the PCA with two components for each of the datasets. The different colors indicate the different classes. The Figure shows that the separation of the instances for each class is different for each dataset; for some classes a linear model might be able to do a decent job separating the instances from each class; in other datasets, the differences might be more subtle. For this reason, we believe that different families of classifiers might

Table 5 – Datasets

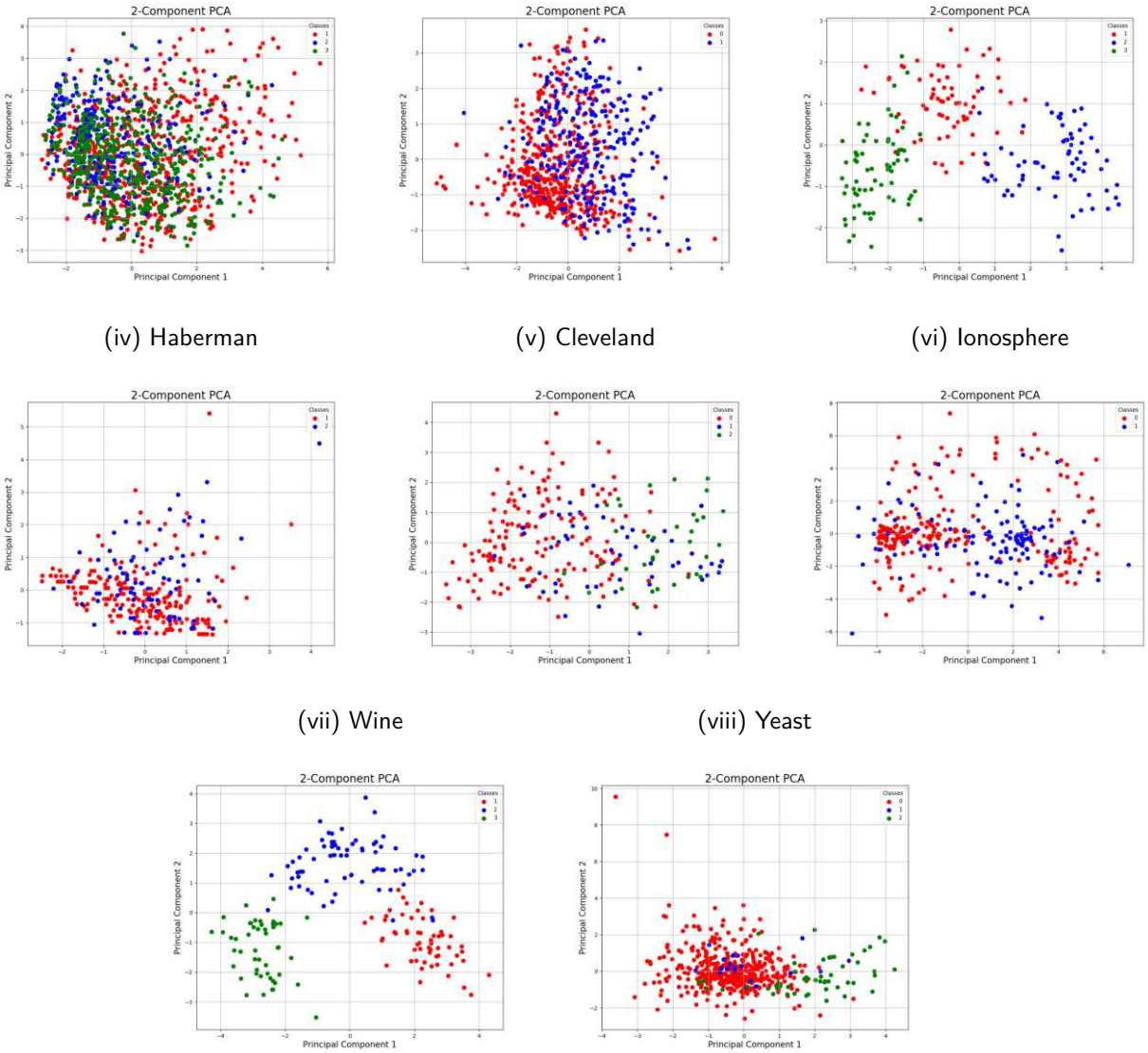
Dataset	**Nr of instances**	**Nr of features**	**Nr of classes**
Contraceptive	1473	10	3
Pima-diabetes	768	9	2
Wheat seeds	210	8	3
Haberman	306	4	2
Cleveland	303	14	5
Ionosphere	351	35	2
Wine	178	14	3
Yeast	1484	10	10

Source: Author's own elaboration

be better suited to solve each dataset, that is, a single model might not be the best for every dataset. We expect that the IRT model will capture these differences in terms of the different abilities that will be assigned to each classifier depending both on the dataset as in the missing scenario.

Figure 16 – 2-component PCA per data set

(i) Contraceptive (ii) Pima (iii) Wheat seeds



(iv) Haberman (v) Cleveland (vi) Ionosphere

(vii) Wine (viii) Yeast

Source: Author's own elaboration

4.2 CLASSIFIER MODELS

To evaluate the datasets using IRT, a subset of classifiers was selected. We tried to select a wide range of types of classification models, such as tree-based models, Naïve Bayes, K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), and linear models. The following classifiers were used:

- AdaBoost Classifier (ABC) (FREUND; SCHAPIRE, 1997) is a boosting algorithm that is based on what is referred as *weak learning algorithm*, which is a learning algorithm

that performs just slightly better than random guessing. AdaBoost is essentially an ensemble of these weak learners that uses an iterative approach to reduce the errors of these classifiers, thus making them stronger together. This model is called AdaBoost because it adjusts adaptively to the errors of misclassification, assigning higher weights to instances considered hard.

- Decision Tree (DT) (QUINLAN, 1986) is the simplest of the tree-based classification methods. The classification in this type of method happens through the Top-Down Induction of Decision Trees (TDIDT), where the classification is constructed from the root of the tree, going down through its leaves. The goal of this induction task is to come up with a classification rule, which can determine the class of any instance that is given as input to the decision tree classifier, even outside the instances that were used to train the model (that is, to construct the classification rule of the tree).

- Extra Trees Classifier (ETC) (GEURTS; ERNST; WEHENKEL, 2006), also known as Extremely Randomized Trees Classifier, is an ensemble tree supervised model that consists essentially of choosing the parameters of both attribute and cut-point partially or totally at random.

- Gradient Boosting Classifier (GBC) (FRIEDMAN, 2001) is a gradient-descent based boosting model focused on classification tasks, which is derived from formulations such as the one for AdaBoost (FREUND; SCHAPIRE, 1997). In these types of models, also called Gradient Boosting Machine (GBM), the learning happens as an iterative procedure that fits new models to more accurately estimate the target value. The idea is to construct weak learners that are correlated with the negative gradient of the loss function. The choice of loss functions is wide, and that is one reason these models are highly customizable. GBMs can be used for regression or classification tasks, depending on the formulation and the loss function used in the model.

- Gaussian Naïve Bayes (GNB) (HAND; YU, 2001) is a classifier based on the Bayes theorem for the conditional probability (combining prior and posterior probabilities for each class) and the assumption that the prediction variables are independent. Even though this assumption might be wrong in most cases, because the Bayes model is much simpler, requiring few parameters to be estimated which makes the variance of the the estimation of the conditional probability to be lower, which might compensate the increased bias.

- Gaussian Process Classifier (GPC) (WILLIAMS; RASMUSSEN, 1996) is a generative classifier model based on Gaussian processes. The predictive distribution is obtained from a joint Gaussian distribution for the outputs of training and test data, conditioned on the observation of training data.

- KNN (COVER; HART, 1967) is a classifier based on the nearest neighbor rule, which assigns to a given point p the class of the majority of its k neighbors.

- MLP (ROSENBLATT, 1958) is a type of Feed-Forward Neural Network (FFNN) that consists of at least three layers of nodes (one for the input, one hidden, and one for the output). Each node has an activation function and the model evolves and correct its mistakes using a back-propagation algorithm.

- Quadratic Discriminant Analysis (QDA) (COVER, 1965) is a classifier model with a quadratic decision surface to separate data from two or more classes. In essence, it is similar to linear classification models such as logistic regression, with the difference that the decision surface is quadratic instead, which makes it capable of separating more complex data shapes when compared to its linear counterparts.

- Random Forest Classifier (RF) (BREIMAN, 2001) is an ensemble model of several Decision Trees that are given slightly different subsets of features to classify. It is a more robust method when compared to typical single decision trees.

- Support Vector Machine Classifier (SVC) (BOSER; GUYON; VAPNIK, 1992) is a Support Vector Machine classifier, which is a prediction model whose goal is to maximize the margin that separates two different classes. SVMs initially worked as linear models, but recently these models can deal with nonlinearity by using a nonlinear kernel and applying the so-called "kernel-trick".

- XGBoost Classifier (XGB) (CHEN; GUESTRIN, 2016) is a boosting tree algorithm focused on scalable processing of large amounts of data.

- Light Gradient Boosting Models (LGBM) (KE et al., 2017a) classifier is Microsoft's implementation of a boosting tree classifier, focused on dealing with large amounts of data.

- CatBoost Classifier (CBC) (PROKHORENKOVA et al., 2018) is a boosting tree classifier that is optimized to deal with categorical features.

Table 6 sums up the types of classifiers that were described above.

Table 6 – Classifier Types

Classifier	**Type of classification**
AdaBoost	Boosting algorithm
Decision Tree	Single traditional decision tree
Extra Trees	Random tree ensemble
Gradient Boosting Classifier	Boosting algorithm
Gaussian Naive Bayes	Bayes rule classifier
Gaussian Process	Joint Gaussian mixtures
KNN	Neighboring algorithm
MLP	Neural Network
QDA	Quadratic decision function
RF	Tree bagging ensemble
SVC	Margin maximization algorithm
XGBoost	Tree boosting ensemble
LGBM	Tree boosting ensemble
Catboost	Tree boosting ensemble

Source: Author's own elaboration

In our experimental evaluation, all classifiers but XGBoost, CatBoost and LGBM are implemented in the *scikit-learn* (PEDREGOSA et al., 2011) library; XGBoost is implemented in the *xgboost* library (CHEN; GUESTRIN, 2016) and LGBM is implemented in the *LightGBM* library (KE et al., 2017b).

4.2.1 Model Parameters

Using the default values of each model is unlikely to give the best performance for that model. Tuning the parameters to find a better suited model is common practice in this case. However, because hyperparameter optimization is a very expensive, exponential task, the parameters of the models were selected using a random search. We used random search because it is much more efficient than grid search and can provide good results in a fraction of the execution time (BERGSTRA; BENGIO, 2012).

In order to ensure reproducibility, we define a seed for all tasks that might have a random factor, such as the random search for the best parameters, the train/validation splits of the datasets, and the models themselves. More details about the execution environment used to

run the experiments, such as the search space for the hyperparameter optimization and the libraries used to run the experiments is presented in Appendix A.

4.3 FEATURE SELECTION METHODS

For feature selection techniques we tried to select a set that had models with different approaches, that is, filter-based, wrapper, and embedded models. Besides these techniques, we also add tree-based models in the comparison. Table 7 shows a quick summary of the previously described feature selection techniques. In our experiments we used these feature selection techniques to estimate the ranking of each feature for a given dataset.

Table 7 – Feature selection methods

Method	**Approach**
Correlation	Filter-based
RFE_SVC	Wrapper
RFECV_SVC	Wrapper
RFE_LREG	Wrapper
RFECV_LREG	Wrapper
Ridge	Embedded
Logistic Regression	Embedded
Extra Trees	Tree-based
Random Forest	Tree-based

Source: Author's own elaboration

Correlation is the Pearson correlation, implemented in the *scipy* Python package. The RFE algorithm needs an estimator, which will be used to select the best features according to some criteria. We are using RFE both with a SVC classifier (RFE_SVC and RFECV_SVC) as well as a Logistic Regression classifier (RFE_LREG and RFECV_LREG).

Extra Trees and Random Forest are tree-based models; this means that they will have a feature importance attribute to assess the importance assigned to each feature. Although it is a linear classification model and not properly a feature selection model, we use Logistic Regression to perform feature selection. The resulting coefficients of this model can be used to evaluate the importance of a feature in a dataset. We do not use Lasso to create the feature ranking because Lasso is more appropriate to evaluate relevant / non relevant features, and not to create a feature ranking; many features can be considered as non important, and their

Lasso coefficient will be equal to zero. The alpha parameter can be changed to create a less restrictive model, but when alpha=0, Lasso behaves like a Logistic Regression, so we preferred to use only Logistic Regression instead.

Besides these classic feature selection techniques, we also use β^3-IRT to perform the ranking of the features of each dataset. We use two approaches: ability change and difficulty change. When using ability change, we measure how much the mean ability of all classifiers changed from the scenario without any missing features to each missing feature scenario; we consider that classifiers should have lower ability values the more relevant the feature missing is. For the difficulty change technique, we apply a similar method to evaluate the importance of the feature, but the main difference is that the difficulty is not a parameter of the models but rather the instances of the dataset; we consider that the mean difficulty of the instances should be higher the more relevant a feature is for that dataset.

4.4 IMPUTATION TECHNIQUES

For this work, we decided to introduce missing data one feature at a time. For the application of our solution, this is the best way to evaluate the impact of missing one given feature. This purpose of this work is not to understand the impact of random missing data in the evaluation of classifiers, but rather to understand the impact of missing one single feature at test time. One of the ideas is to understand whether or not collecting the data that was missing and thus imputed is worth it. When data is missing, there is a cost for gathering this data, and the objective is to understand if imputing the missing data can offer satisfactory results or if the stakeholders should gather the missing data instead.

For the experiments, three types of data imputation were used: simple, iterative, and KNN. The Simple imputation is performed by using either the mean, the median, or the mode of the training set for a given missing feature. The mean is used when the feature is numerical and continuous (float values); the median is used when the feature is numerical and discrete (integer values); the mode is used when the feature is categorical.

The Iterative method used in the experiments is an implementation of the MICE package (BUUREN; GROOTHUIS-OUDSHOORN, 2011) in Python, which is available in the *statsmodels* Python library (SEABOLD; PERKTOLD, 2010) under the name *MICEData*. Since this imputation method can be used for both multivariate and univariate imputation scenarios, when using the univariate imputation, each variable has its own imputation model, which we believe is appro-

priate to use with our experimental setup of one missing feature at a time. Unlike the Simple imputation, where only data for the missing feature is given (and necessary) to the imputation model, the Iterative imputer uses the whole data to create a model for a given feature.

The KNN imputation is performed using the *sklearn* class *KNNImputer*. We set the number of neighbors to $k = 5$ for all scenarios in all datasets evaluated. The constant value for the number of neighbors was chosen to simplify the experiments, but it is possible to choose a more appropriate value of k by performing cross-validation to tune this parameter for each dataset. Similarly to the Iterative imputation method, the KNN also uses information about the whole training data to select the k nearest neighbors of a given instance.

4.5 IRT MODEL

As it was previously discussed in this dissertation, to evaluate classifier models we need an IRT model that is capable to deal with bounded continuous responses to model the probability output of each classifier model for each instance being evaluated.

In IRT models, the optimization phase is usually performed using a Expectation-Maximization (EM) iterative approach. We use the β^3-IRT model proposed by Chen et al. (2019). We use an implementation [1] that uses Gradient Descent (GD) to perform this optimization iteration. It is implemented using *Tensorflow 2*, and is capable of running in parallel or Graphics Processing Unit (GPU) to lower the computational time of the IRT analysis. The parameters used in our evaluation are described in Table 8; any other parameters of this model used their respective default values. The IRT problem configuration is described in Table 9.

Table 8 – Parameters used for BIRT

Parameter	**Value**
Learning rate	1.0
Number of initializations	2500
Total number of iterations	5000

Source: Author's own elaboration

β^3-IRT implements a 2-PL IRT model, which means that results are given in terms of:

- Model ability: how capable is the model at correctly predicting the real class of the instances.

[1] https://github.com/Manuelfjr/birt-gd/

Table 9 – BIRT configuration in classification tasks

IRT Parameter	**Mapping**
Respondent	Classifier model
Item	Instances of a dataset
Response	Probability of classifier *c* correctly classifying instance *i* with class *k*

Source: Author's own elaboration

- Instance difficulty: how difficult it is for a classifier to correctly predict the probability of the real class of the instances.
- Instance discrimination: how discriminative is an instance at separating the best from the worst classifiers, that is, if the prediction of the set of classifiers is consistent for instance i, with better classifiers predicting the probability of the real class correctly, and worse classifiers not predicting the real class correctly.

4.6 OLS REGRESSION ANALYSIS

In this section the details of the evaluation setup for each step of the OLS regression analysis are explained. The OLS regression analysis is split into three steps: OLS Regression, Data Binarization, and Classification Analysis.

4.6.1 OLS Regression

The OLS regression is simply a linear regression, that is, the points of the prediction fall within a prediction line. The figure shows that the number of instances that fall inside the confidence interval as well as the size of the confidence interval can largely vary. This happens because for some of the features (and thus the missing feature scenario) it becomes harder to predict using a simple model how the difficulty of an instance will be affected if that feature is imputed. As it is expected, the type of imputation can also change the behavior of the difficulty in each scenario and for each instance, meaning that a better method of imputing data might results in a better prediction for the difficulty of that instance.

The OLS model separates the predicted values into:

- Inside confidence interval

- Below low confidence interval
- Above upper confidence interval
- Inside prediction interval
- Below low prediction interval
- Above upper prediction interval

For our evaluation, we want to analyze if the model can predict which instances will not be inside either the confidence or the prediction intervals. In the next section we explain how we use this information to binarize the results.

4.6.2 Data Binarization

In the previous section it was mentioned that the predictions of the OLS regression can be inside or outside the confidence and the prediction intervals, which are calculated by the model during the model fit.

As our goal is to find the instances that would or would not be most benefited by the gathering of the missing information, we have the choice to binarize the data in terms of the confidence or prediction interval, and the data being above or below these intervals. Analysing the datasets used in the evaluation, it became clear that binarizing the data using the prediction interval would always result in a much more imbalanced classification problem than when we used the confidence interval to do the same. Because we mostly want to find out which instances are the most impacted by the missing information, we binarize the data by separating the ones that are above the upper confidence interval (class=1) from the instances that are below that (class=0).

In our experiments, we are only considering the confidence interval for the OLS predictions. The main reason for this is that in some scenarios, the binarization using the prediction interval instead of the confidence interval led to a class distribution where 100% of the instances were assigned the same class, while zero instances were assigned the opposite class. Because we need at least some variety of data to be able to split the data into K different training/validation folds, the dataset resulting from the binarization needs to contain at least K elements of each class. Because this was not always the case for the binarization using the prediction interval, we decided to use the confidence interval instead.

The binarization of the data is performed for each missing feature scenario of each dataset. For this reason, the data distribution will be different for each missing feature scenario, even for the same dataset. This means that, in some cases, the classification problem for two different missing scenarios can be a balanced problem (with the two classes with about 50% of the data), while for the other it can be an imbalanced problem (with one class being the majority and one class being the minority). We chose not to remove or add new data (undersampling and oversampling, respectively), but rather use a optimal threshold for each classifier that maximizes the F1-score, which is a middle ground between precision and recall scores.

4.6.3 Classification Analysis

The following classification models were used to evaluate the classification task:

- Logistic Regression
- SVC with RBF kernel
- SVC with linear kernel
- Random Forest
- LGBM
- CatBoost
- XGBoost
- DummyClassifier

We selected a range of different families of models in order to see if at least one of them would be better than a random choice. Each classifier is optimized by using the prediction threshold that maximizes the F1-score for the training data. The F1-score is defined in Equation (4.1)). Even though the best prediction threshold is selected, the Area Under the Receiver Operating Curve (ROC-AUC) score does not change for a given binary classifier model, because it is the area under the curve for several prediction thresholds, indicating the True Positive Rate (TPR) and False Positive Rate (FPR) for each threshold within the curve. In this sense, if a model has a high ROC-AUC value, it is expected that it will have a threshold that optimizes

the precision or the recall score, which a practitioner can select according to what is best for the problem at hand.

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{4.1}$$

We evaluate the classifiers on each scenario using ROC-AUC score, because it gives a good information about the performance of classifiers for a binary problem, and it is a more straight-forward way to evaluate the results of the models comparing to a random baseline, as random classifiers will have a ROC-AUC score of about 0.5. The *DummyClassifier* model is used as a way to simulate a random classifier. The DummyClassifier is a model from the *sklearn* library that always predicts a random label for each instance, so its ROC-AUC score is expected to always be close to 0.5, which is the baseline to evaluate if the other classifier models are able of providing better results than just a random prediction.

As each missing scenario is evaluated individually, there are performance results for each model in each of these scenarios as well. We expect that if the model is predicting better than the random classifier, it is a good indication that we can rely on this solution to make a decision about whether a missing feature should be gathered for any given instance. If the model outputs 1, it means that that feature is relevant for that instance and should be gathered; on the other hand, if the output is 0, it means that imputing the missing data for that instance has given good enough results, and the missing information does not need to be gathered.

In the next chapter we will show the evaluation results of the experiments that were executed with the setup described in the current chapter.

5 EXPERIMENTAL RESULTS

In this chapter, we present the results of executing the workflow for IRT in a missing feature scenario over some selected classification datasets. This chapter is organized as follows. Section 5.1 presents a set of feature ranking techniques and compares a feature ranking created by each of these methods and by our proposed solution, in the different imputation scenarios. Section 5.2 shows how the IRT method proposed in this dissertation can be used to analyze an instance individually. Section 5.3 presents a case study with two datasets, to show the IRT results in more detail. Section 5.4 presents a OLS model fit and prediction of instance difficulties. Finally, Section 5.5 presents a deeper discussion on the results obtained in this chapter.

5.1 COMPARISON TO FEATURE SELECTION TECHNIQUES

Using the feature selection techniques previously described in Section 4.3, the feature ranking is created for each of those feature selection techniques. The ranking for the β^3-IRT solution can also depend on the imputation technique used to fill the missing values for each missing feature.

The β^3-IRT proposal for evaluating the feature ranking is to use both the ability change and difficulty change values for each scenario and for each dataset to evaluate the features that are the most impactful on these scores.

Table 10 shows the feature ranking for the Contraceptive dataset with Simple imputation and Table 11 with KNN imputation. Figure 17 shows the heatmap for the feature ranking of the Contraceptive dataset with Simple Imputation, and Figure 18 shows the heatmap for KNN Imputation. The heatmap figures show more clearly the differences between the feature ranking techniques; it is clear that for both imputation scenarios, the proposed solutions (Beta-IRT_ab and Beta-IRT_diff) are more similar to the tree-based techniques and to the Correlation filter technique, although there is a mismatch for the top features. One interesting thing to notice is that the top feature when considering the difficulty change is different in Simple Imputation (w_age) and KNN Imputation (w_edu). We believe that this change in ranking is expected given that the quality of imputation can be very impactful on the results of the respondents in a IRT problem.

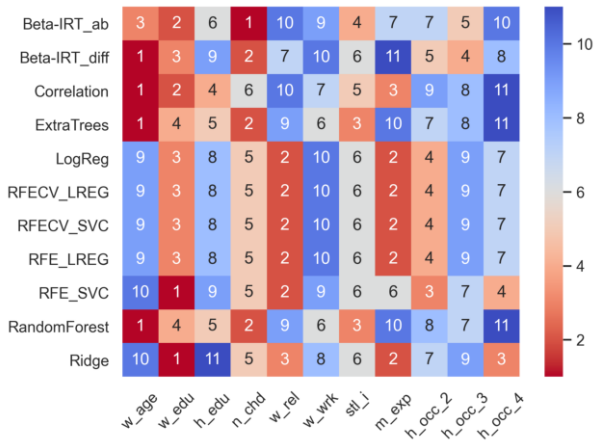
Table 12 shows the feature ranking for the Pima dataset with Simple imputation and Table 13 with KNN imputation. Figure 19 shows the heatmap of the feature ranking for the Pima dataset with Simple Imputation, and Figure 20 shows the heatmap for KNN Imputation. For the Pima dataset there was also a change in the top features when considering the Simple and KNN Imputation scenarios. For Simple Imputation, the rankings of Beta-IRT_ab and Beta-IRT_diff are very similar, while for the KNN Imputation the rankings were more distinct from each other. In special, for Beta-IRT_ab the pl_glu feature dropped a lot in the ranks, which means that when a better imputation technique is used (such as KNN), missing this feature is less impactful.

Table 10 – Median feature ranking for Contraceptive with Simple imputation

	w_age	w_edu	h_edu	n_chd	w_rel	w_wrk	stl_i	m_exp	h_occ_2	h_occ_3	h_occ_4
Beta-IRT_ab	3	2	6	1	10	9	4	7	7	5	10
Beta-IRT_diff	1	3	9	2	7	10	6	11	5	4	8
Correlation	1	2	4	6	10	7	5	3	9	8	11
ExtraTrees	1	4	5	2	9	6	3	10	7	8	11
LogReg	9	3	8	5	2	10	6	2	4	9	7
RandomForest	1	4	5	2	9	6	3	10	8	7	11
RFECV_LREG	9	3	8	5	2	10	6	2	4	9	7
RFECV_SVC	9	3	8	5	2	10	6	2	4	9	7
RFE_LREG	9	3	8	5	2	10	6	2	4	9	7
RFE_SVC	10	1	9	5	2	9	6	6	3	7	4
Ridge	10	1	11	5	3	8	6	2	7	9	3

Source: Author's own calculation

Figure 17 – Heatmap of median feature ranking for Contraceptive with Simple imputation



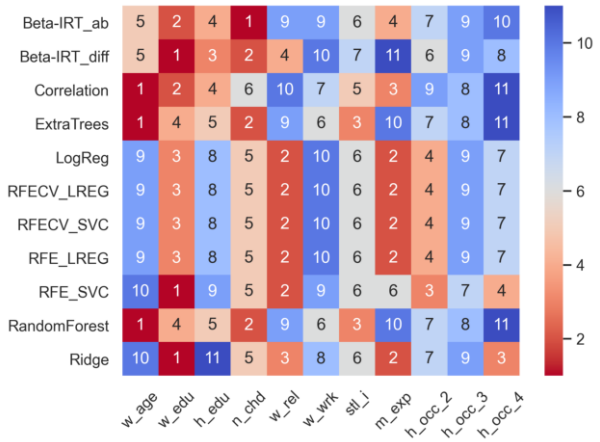
Source: Author's own elaboration

Table 11 – Median feature ranking for Contraceptive with KNN imputation

	w_age	w_edu	h_edu	n_chd	w_rel	w_wrk	stl_i	m_exp	h_occ_2	h_occ_3	h_occ_4
Beta-IRT_ab	5	2	4	1	9	9	6	4	7	9	10
Beta-IRT_diff	5	1	3	2	4	10	7	11	6	9	8
Correlation	1	2	4	6	10	7	5	3	9	8	11
ExtraTrees	1	4	5	2	9	6	3	10	7	8	11
RFECV_LREG	9	3	8	5	2	10	6	2	4	9	7
RFECV_SVC	9	3	8	5	2	10	6	2	4	9	7
RandomForest	1	4	5	2	9	6	3	10	7	8	11
RFE_LREG	9	3	8	5	2	10	6	2	4	9	7
RFE_SVC	10	1	9	5	2	9	6	6	3	7	4
Ridge	10	1	11	5	3	8	6	2	7	9	3

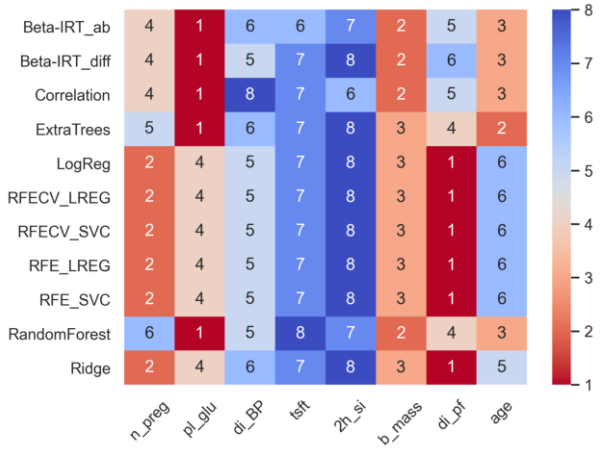
Source: Author's own calculation

Figure 18 – Heatmap of median feature ranking for Contraceptive with KNN imputation



Source: Author's own elaboration

Figure 19 – Heatmap of median feature ranking for Pima with Simple imputation



Source: Author's own elaboration

Table 12 – Median feature ranking for Pima with Simple Imputation

	n_preg	pl_glu	di_BP	tsft	2h_si	b_mass	di_pf	age
Beta-IRT_ab	4	1	6	6	7	2	5	3
Beta-IRT_diff	4	1	5	7	8	2	6	3
Correlation	4	1	8	7	6	2	5	3
ExtraTrees	5	1	6	7	8	3	4	2
LogReg	2	4	5	7	8	3	1	6
RandomForest	6	1	5	8	7	2	4	3
RFECV_LREG	2	4	5	7	8	3	1	6
RFECV_SVC	2	4	5	7	8	3	1	6
RFE_LREG	2	4	5	7	8	3	1	6
RFE_SVC	2	4	5	7	8	3	1	6
Ridge	2	4	6	7	8	3	1	5

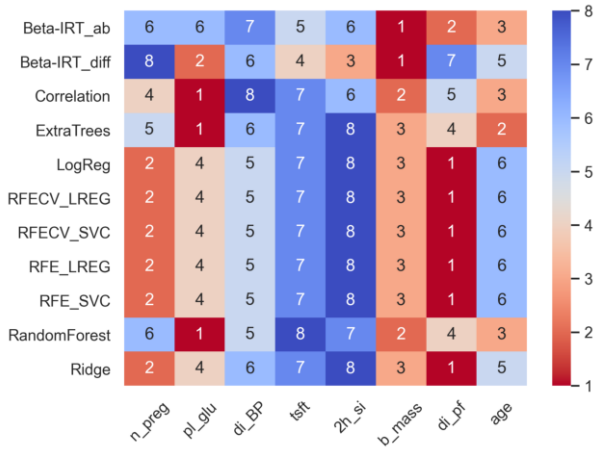
Source: Author's own calculation

Table 13 – Median feature ranking for Pima with KNN Imputation

	n_preg	pl_glu	di_BP	tsft	2h_si	b_mass	di_pf	age
Beta-IRT_ab	6	6	7	5	6	1	2	3
Beta-IRT_diff	8	2	6	4	3	1	7	5
Correlation	4	1	8	7	6	2	5	3
ExtraTrees	5	1	6	7	8	3	4	2
LogReg	2	4	5	7	8	3	1	6
RandomForest	6	1	5	8	7	2	4	3
RFECV_LREG	2	4	5	7	8	3	1	6
RFECV_SVC	2	4	5	7	8	3	1	6
RFE_LREG	2	4	5	7	8	3	1	6
RFE_SVC	2	4	5	7	8	3	1	6
Ridge	2	4	6	7	8	3	1	5

Source: Author's own calculation

Figure 20 – Heatmap of median feature ranking for Pima with Knn imputation



Source: Author's own elaboration

For the Contraceptive dataset, the ranking results show that the ranking proposed by the β^3-IRT is not identical to any other method compared. Also, the results when we are analyzing the difficulty or the ability changes over the scenarios are not equal, even though they have some similarities; the top 3 features of β^3-IRT considering the ability change are the same as the ones considering the difficulty change, although the order varies. These features are not in the top three of any other methods, but we can see some similarities with the rankings produced by our solution and the tree-based rankings.

For the Pima dataset, the results show that the features ranked as the most important by the β^3-IRT regarding both ability and difficulty changes were pl_glu and b_mass. All RFE-based models presented identical feature rankings.

It is interesting to observe that for the Pima dataset, specially, that the ranking can change when analyzing the results for the two different imputation techniques. This makes sense, because the dataset resulting from the imputation can be very different, which will affect the performance of the models when predicting the class of the instances.

5.2 INSTANCE-LEVEL ANALYSIS

One of the main advantages of our proposed solution when compared to other approaches is that the analysis can be made at the instance level, and not only at feature level.

At training level, the IRT approach can provide a clear way of analyzing the importance each feature can have over each instance. As in this scenario we do have the real data for all features and all instances, we can really analyze how the difficulty and discrimination of an instance changes when each of the features is missing, and how the imputation method chosen impacts this result as well.

For a more real world application of feature gathering, we imagine that the features might not be available at test time. Once we run the IRT method over all the training instances of a dataset, splitting the dataset in the way we showed in the previous section, with different folds used for training and validation, where the IRT parameters are calculated for each instance of the validation set, for each fold.

Both the missing feature and the imputation technique used to fill out the missing values can have an impact on the difficulty (and discrimination) of an instance. Table 14 shows the different values of difficulty an instance had in different missing feature scenarios for all the imputation techniques used in the experiment, for the Contraceptive dataset. Figure 21

shows the same results for an instance of the Pima dataset. Both tables show that for the better imputation methods (KNN and Iterative), the difference in the difficulty values is less pronounced. As for the simple imputation scenario, the difference is more accentuated in the features considered as more relevant, that is, in these scenarios, the difficulty of the instance increases. For the Contraceptive dataset, a significant increase in the difficulty can be seen in the feature scenarios w_age, w_edu, n_chd, and stl_i. This indicates that for these scenarios, it gets more difficult to predict the correct class of the instance, which is arguably because missing the feature information impacts the quality of the prediction task.

As for the Pima dataset, a similar scenario can be observed. The highest change in difficulty was for the feature b_mass, over all imputation scenarios. Besides the features b_mass and di_BP, all other features had a lower difficulty value in at least one of the imputation scenarios. This might have happened because these features are not so important to correctly classify this specific instance, when comparing to di_BP and b_mass. This analysis shows that the solution we proposed can also be used to evaluate instances on a individual basis, which can be a very important step in a data analysis pipeline.

Table 14 – Instance difficulty values for different scenarios: Contraceptive dataset

	simple	knn	iterative
original	0.190	0.190	0.190
w_age	0.393	0.203	0.351
w_edu	0.300	0.240	0.352
h_edu	0.189	0.189	0.353
n_chd	0.200	0.182	0.352
w_rel	0.190	0.190	0.353
w_wrk	0.190	0.190	0.353
stl_i	0.208	0.190	0.352
m_exp	0.190	0.190	0.352
h_occ_2	0.189	0.178	0.353
h_occ_3	0.189	0.190	0.352
h_occ_4	0.190	0.190	0.353

Source: Author's own calculation

Figure 21 – Instance difficulty values for different scenarios: Pima dataset

	simple	knn	iterative
original	0.306	0.306	0.306
n_preg	0.272	0.307	0.287
pl_glu	0.255	0.295	0.305
di_BP	0.323	0.341	0.339
tsft	0.319	0.309	0.276
2h_si	0.279	0.307	0.309
b_mass	0.474	0.476	0.505
di_pf	0.326	0.356	0.297
age	0.232	0.294	0.296

Source: Author's own calculation

5.3 CASE STUDY

In this section, we will show the experimental results of two of the datasets in more detail. The chosen datasets were the Contraceptive dataset and the Pima Diabetes dataset.

We selected the Contraceptive dataset for the following reasons: mostly integer / discrete features, number of features is not too large and are easily interpretable, and its target variable has more than two labels (multiclass classification problem). The second dataset we chose for the case study, the Pima Diabetes dataset, has some characteristics which are similar to the Contraceptive dataset, such as the number of features is not too large and are easily interpretable, but this dataset has a target with only two classes (binary classification problem), and the features are mostly continuous, unlike most of the features in the Contraceptive dataset.

5.3.1 Instance analysis

Table 15 shows an example of instance imputation for the Contraceptive dataset and Table 16 shows an example of imputation of an instance of the Pima Diabetes dataset, for each imputation technique used. It is relevant to note here that we grouped the imputed values together to be easier to see and analyze the results, but in the training pipeline that we have defined, only one feature is missing (and thus imputed) at a time.

As we have seen in the initial discussions about the imputation techniques, it is clear that

the simple imputation is the weaker method, as it is not specific for the instance but rather for the whole variable, so the imputation will always be of a value that is more in the middle of the distribution, thus not imputing a very similar value for instances with more extreme values. Take for instance the feature 'w_age', which is the age of the woman; the Simple Imputer imputes the mean value of the training dataset, which indicates that most women in this survey were older than the one represented by this specific instance. The KNN, as it takes the value based on the nearest instances to this one, imputes a value that is very close to the real one; the iterative imputation, because it runs the imputation process several times while trying to minimize the error, also imputes a value that is similar to the one in the instance without any missing data.

Although the features are different and even the type of the features is different (mostly discrete in the Contraceptive dataset and mostly continuous in the Pima dataset), the Pima dataset shows similar results regarding the imputation of each missing feature.

Table 15 – Imputation for an instance: Contraceptive

	w_age	w_edu	h_edu	n_chd	w_rel	w_wrk	stl_i	m_exp	h_occ_2	h_occ_3	h_occ_4
simple	32	4	4	3	1	1	4	0	0	0	0
knn	17	3	4	1	1	1	4	0	0	1	0
iterative	39	1	2	2	1	1	1	1	0	1	0
original	16	3	4	1	1	1	3	0	0	1	0

Source: Author's own calculation

Table 16 – Imputation for an instance: Pima Diabetes

	n_preg	pl_glu	di_BP	tsft	2h_si	b_mass	di_pf	age
simple	3	117.500	72.000	23.000	27.000	32.135	0.481	29
knn	1	111.000	80.000	28.000	0.000	31.540	0.589	25
iterative	2	85.000	60.000	39.000	120.000	32.400	0.560	24
original	1	108.000	88.000	19.000	0.000	27.100	0.400	24

Source: Author's own calculation

5.3.2 Difficulty, discrimination, and ability change

One of the ways we can evaluate our proposed solution is by analyzing the ability and difficulty values in each missing feature scenario, comparing these metrics to the case with no

missing data. We expect that the most relevant features will be the ones where the change both in instance difficulty as in model ability will be the largest. For a relevant feature, we expect that the mean difficulty of the instances will be higher, while the mean ability of the classifiers will be lower. Instances will be more difficult to classify correctly if relevant data is missing, so the classifiers are expected to make more mistakes in these scenarios. In this section we experimentally demonstrate this intuition about the behavior of these two parameters.

First of all, we analyze the mean values for the difficulty and discrimination values of instances in the Contraceptive dataset and the Pima dataset. Table 17 shows the difficulty values for each scenario of the Contraceptive dataset with Simple imputation; Table 18 shows the difficulty values for the Pima dataset with Simple imputation.

For the Contraceptive dataset, the mean of the difficulty values increased more significantly for features w_age, w_edu, and n_chd. Interestingly, while the minimum difficulty value increased for the dataset, the maximum difficulty value found decreased, specially for the aforementioned features. This might indicate that the most difficult instances decreased their difficulty score, maybe because the simple imputer makes them be closer to the average, at least on the feature that was imputed. For the Pima dataset, we see a similar scenario, where the difficulties for pl_glu and b_mass are highest in the mean, although the maximum values decreased.

For the discrimination, the observed changes are different than for the difficulties. The mean discrimination is lower for the most relevant features, in general.

Table 19 shows discrimination values for the Contraceptive dataset with Simple imputation, and finally, Table 20 shows discrimination values for the Pima dataset with Simple imputation.

Analyzing the difficulty, we can observe that for the features considered as most relevant, the mean difficulty of the instances of the dataset increased, as well as the minimum difficulty; however, the maximum difficulty value has decreased for some features considered relevant and stayed the same for features considered as less important.

As for the discrimination, the mean decreased in all scenarios when compared to the original, for the Simple Imputation. It is interesting to observe the minimum values of discrimination: for the Contraceptive, negative discrimination values were found for the stl_i missing feature scenario, while for relevant features such as w_age, the minimum discrimination has increased significantly.

It is interesting to observe how these two metrics behave when the imputation scenario is different (KNN or Iterative). The difficulty dispersion for the other imputation techniques and

Table 17 – Mean difficulty per scenario: Contraceptive with Simple Imputation

	mean	min	max
original	0.5027	0.1004	0.9508
w_age	0.5046	0.1199	0.8592
w_edu	0.5043	0.1049	0.8974
h_edu	0.5027	0.0998	0.9353
n_chd	0.5043	0.1189	0.9543
w_rel	0.5028	0.1003	0.9509
w_wrk	0.5026	0.1002	0.9504
stl_i	0.5029	0.1021	0.9503
m_exp	0.5025	0.1068	0.9334
h_occ_2	0.5030	0.0999	0.9507
h_occ_3	0.5034	0.0967	0.9521
h_occ_4	0.5027	0.1004	0.9508

Source: Author's own calculation

Table 18 – Mean difficulty per scenario: Pima with Simple Imputation

	mean	min	max
original	0.468	0.128	0.963
n_preg	0.468	0.131	0.963
pl_glu	0.477	0.125	0.947
di_BP	0.467	0.118	0.964
tsft	0.467	0.129	0.961
2h_si	0.467	0.133	0.963
b_mass	0.472	0.134	0.937
di_pf	0.467	0.130	0.963
age	0.470	0.149	0.954

Source: Author's own calculation

the rest of the datasets can be found in Appendix B and the discrimination dispersion can be found in Appendix C.

Table 21 shows the mean ability per model and scenario (calculated as the mean over the 5 validation folds) for the Contraceptive dataset and Table 22 shows these results for the Pima dataset. The results are shown for the Simple imputation technique for both datasets.

The results show that the abilities are overall much lower in the Contraceptive dataset than in the Pima dataset, even for the complete data (*original* column). Besides AdaBoost, all models in the Pima dataset have higher abilities than in the Contraceptive dataset. The

Table 19 – Mean discrimination per scenario: Contraceptive with Simple Imputation

	mean	min	max
original	1.0443	0.0615	1.6167
w_age	1.0327	0.2171	1.5714
w_edu	1.0356	0.1301	1.5050
h_edu	1.0400	0.0249	1.5979
n_chd	1.0434	0.0656	1.5851
w_rel	1.0437	0.0618	1.6191
w_wrk	1.0443	0.0613	1.6150
stl_i	1.0375	-0.0378	1.5723
m_exp	1.0428	0.0617	1.6190
h_occ_2	1.0413	0.1558	1.6151
h_occ_3	1.0372	0.0586	1.6157
h_occ_4	1.0445	0.0610	1.6165

Source: Author's own calculation

Table 20 – Mean discrimination per scenario: Pima with Simple Imputation

	mean	min	max
original	0.977	-0.434	1.783
n_preg	0.980	-0.362	1.773
pl_glu	0.933	-0.759	1.814
di_BP	0.985	-0.414	1.800
tsft	0.981	-0.389	1.791
2h_si	0.977	-0.392	1.777
b_mass	0.972	-0.696	1.745
di_pf	0.986	-0.529	1.841
age	0.988	-0.768	1.851

Source: Author's own calculation

reason for this might be the fact that Contraceptive has three target labels, while Pima has two.

It is interesting to analyze that there are missing feature scenario s where the ability of a given classifier can increase. This happens for a lot of scenarios for the AdaBoost classifier, for instance, both in the Contraceptive and the Pima datasets. For the case of AdaBoost, it is not a good classifier for neither dataset, so this difference can be related to noise in the model. A classifier with a higher ability, such as the QDA or the GNB in the Pima dataset, usually see a decrease in the ability for most missing features scenarios. However, there are

scenarios where the ability does not change much or changes very little, and we associate this phenomena with the fact that this feature should be less relevant for the dataset globally.

Table 21 – Ability per model and scenario: Contraceptive with Simple Imputation

	original	w_age	w_edu	h_edu	n_chd	w_rel	w_wrk	stl_i	m_exp	h_occ_2	h_occ_3	h_occ_4
AdaB	0.321	0.328	0.332	0.322	0.32	0.32	0.321	0.326	0.323	0.321	0.32	0.32
DT	0.451	0.403	0.393	0.447	0.396	0.451	0.452	0.45	0.447	0.452	0.448	0.451
ETC	0.422	0.393	0.378	0.423	0.387	0.423	0.423	0.421	0.42	0.422	0.422	0.422
GBC	0.447	0.417	0.415	0.444	0.397	0.447	0.447	0.441	0.444	0.443	0.442	0.447
GNB	0.412	0.396	0.391	0.396	0.419	0.413	0.409	0.396	0.403	0.407	0.403	0.414
GPC	0.396	0.389	0.375	0.397	0.379	0.396	0.397	0.39	0.397	0.394	0.393	0.396
KNN	0.429	0.406	0.396	0.423	0.4	0.43	0.427	0.416	0.429	0.427	0.425	0.428
MLP	0.459	0.435	0.426	0.461	0.41	0.459	0.459	0.446	0.458	0.456	0.457	0.46
QDA	0.458	0.422	0.429	0.453	0.437	0.458	0.457	0.445	0.458	0.457	0.457	0.458
RF	0.423	0.396	0.394	0.417	0.384	0.423	0.424	0.417	0.42	0.422	0.421	0.423
SVC	0.405	0.365	0.388	0.403	0.396	0.404	0.404	0.398	0.406	0.404	0.404	0.404
XGB	0.438	0.412	0.403	0.437	0.389	0.438	0.438	0.434	0.437	0.437	0.437	0.438
LGBM	0.442	0.413	0.407	0.443	0.393	0.443	0.442	0.433	0.44	0.438	0.436	0.442
CatB	0.428	0.401	0.396	0.425	0.385	0.428	0.428	0.421	0.428	0.427	0.427	0.428

Source: Author's own calculation

Table 22 – Ability per model and scenario: Pima with Simple Imputation

	original	n_preg	pl_glu	di_BP	tsft	2h_si	b_mass	di_pf	age
AdaB	0.368	0.368	0.376	0.362	0.366	0.367	0.381	0.362	0.378
DT	0.635	0.643	0.641	0.63	0.645	0.639	0.563	0.644	0.57
ETC	0.596	0.585	0.554	0.594	0.597	0.596	0.58	0.593	0.576
GBC	0.694	0.69	0.634	0.69	0.697	0.693	0.639	0.694	0.654
GNB	0.728	0.723	0.684	0.719	0.729	0.727	0.721	0.719	0.732
GPC	0.566	0.561	0.502	0.566	0.564	0.569	0.559	0.563	0.565
KNN	0.614	0.617	0.524	0.614	0.624	0.656	0.622	0.609	0.618
MLP	0.498	0.491	0.447	0.537	0.483	0.516	0.515	0.491	0.493
QDA	0.709	0.703	0.669	0.718	0.714	0.719	0.708	0.707	0.721
RF	0.633	0.624	0.593	0.636	0.636	0.636	0.59	0.64	0.593
SVC	0.609	0.612	0.521	0.61	0.607	0.611	0.613	0.608	0.612
XGB	0.669	0.664	0.603	0.673	0.671	0.669	0.611	0.663	0.636
LGBM	0.634	0.629	0.572	0.63	0.639	0.633	0.59	0.638	0.602
CatB	0.64	0.638	0.581	0.639	0.642	0.645	0.604	0.633	0.62

Source: Author's own calculation

Besides the analysis of the difficulty change on each missing feature scenario, we believe that analysing how the ability of the classifiers change over these scenarios can help to understand the overall relevance of the features for each dataset. We believe that if the mean ability drops when comparing the original scenario without missing data and a missing feature scenario, it means that the performance of the classifiers is being negatively impacted, which means that missing those values is making a significant difference in the predictions.

Table 23 shows the ability change values per scenario for the Contraceptive dataset with Simple Imputation and Table 24 with KNN Imputation. Table 26 shows the changes for the

Pima dataset with Simple Imputation and Table 25 for the KNN Imputation.

For the Contraceptive dataset, comparing the changes on these two imputation scenarios shows how much the quality of the imputation can affect the overall ability of the classifiers. The KNN imputer is capable of imputing values that are more similar to the original ones, so the classifier abilities change less because the original and the imputed instances are more similar. When analyzing the case for the Simple imputer, however, the ability changes are much higher, because the imputation by using the mean, median, or mode for all instances can significantly change how the classifier would predict its class, specially for the most relevant features (w_age, w_edu, and n_chd in this case). From these results, we can infer that it is easier to analyze the overall relevance of features when using a worst imputer, but locally, for each instance, it can be more worth it to try imputing values with a better imputer and analyzing, on an instance basis, if the imputation is enough or if it would be better to gather the missing data, if possible.

For the Pima dataset, the two features with more significant changes when comparing to the original data in the Simple imputation (pl_glu and b_mass) also presented changes a lot lower in the KNN imputation scenario. For both datasets, the standard deviation, maximum, and minimum values are also lower (considering absolute values), which indicates that the KNN imputation scenario is more stable.

Table 23 – Ability change (%) per scenario: Contraceptive with Simple Imputation

Scenario	Mean	Median	Std	Min	Max
w_age	-5.468	-5.319	3.475	-11.078	2.901
w_edu	-5.783	-5.977	4.172	-13.241	3.330
h_edu	-0.818	-0.282	1.339	-3.458	0.944
n_chd	-6.788	-8.676	4.375	-12.451	1.979
w_rel	0.023	0.015	0.416	-0.884	1.107
w_wrk	-0.088	0.039	0.414	-1.111	0.389
stl_i	-1.334	-1.564	1.108	-3.504	1.458
m_exp	-0.192	-0.243	0.387	-0.841	0.560
h_occ_2	-0.382	-0.320	0.784	-1.763	1.833
h_occ_3	-0.077	0.080	0.478	-0.894	0.884
h_occ_4	0.056	-0.010	0.179	-0.040	0.674

Source: Author's own calculation

We implicitly used the metrics of difficulty change and ability change to rank the features based on their relevance according to how much missing a given feature has impacted the

Table 24 – Ability change (%) per scenario: Contraceptive with KNN Imputation

Scenario	Mean	Median	Std	Min	Max
w_age	0.007	-0.037	0.379	-0.357	1.305
w_edu	-0.331	-0.210	0.567	-1.457	0.677
h_edu	-0.396	-0.112	0.799	-3.003	0.202
n_chd	-1.001	-1.171	0.860	-1.979	1.247
w_rel	0.080	0.025	0.205	-0.063	0.791
w_wrk	-0.062	-0.034	0.188	-0.696	0.108
stl_i	-0.098	-0.164	0.513	-0.703	1.512
m_exp	-0.057	-0.003	0.220	-0.644	0.225
h_occ_2	-0.093	-0.082	0.152	-0.412	0.112
h_occ_3	0.223	0.273	0.237	-0.240	0.595
h_occ_4	0.056	-0.010	0.179	-0.040	0.674

Source: Author's own calculation

Table 25 – Ability change (%) per scenario: Pima with KNN Imputation

Scenario	Mean	Median	Std	Min	Max
n_preg	0.726	0.259	1.767	-1.064	5.633
pl_glu	0.505	0.201	1.092	-0.583	3.799
di_BP	0.776	0.771	1.184	-1.038	2.693
tsft	0.501	-0.032	1.188	-0.953	3.520
2h_si	0.049	-0.055	0.280	-0.136	0.978
b_mass	-1.521	-1.655	2.281	-6.411	3.761
di_pf	-0.315	-0.910	3.067	-3.272	8.156
age	-0.036	-0.160	1.003	-1.514	2.655

Source: Author's own calculation

performance of classifiers (ability) or the overall difficulty of the instances of the dataset. This concept of feature ranking is worked with more details in section

To compare the ranking of the features, we use a critical difference diagram. The goal of the Critical Difference Diagram (CDD) analysis is to see which features have a significant difference in the abilities of the classifiers, given the different missing feature scenarios. Figure 22 shows the CDD for the feature ranking of the Contraceptive dataset, and Figure 23 shows the CDD for the feature ranking of the Pima dataset. The results for the other datasets can be seen in Appendix F.

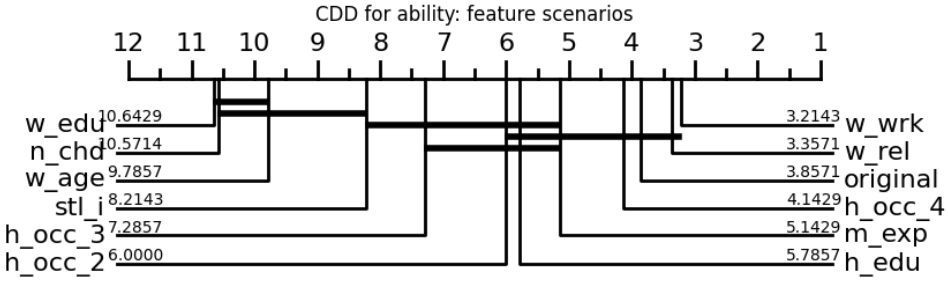
As for the Contraceptive dataset, w_edu is the feature where the abilities are the lowest,

Table 26 – Ability change (%) per scenario: Pima with Simple Imputation

Scenario	Mean	Median	Std	Min	Max
n_preg	-0.973	-1.296	2.479	-3.546	6.929
pl_glu	-7.659	-9.293	8.269	-15.202	14.608
di_BP	1.822	1.207	3.563	-1.494	13.424
tsft	0.712	0.372	2.007	-2.328	5.236
2h_si	0.805	0.318	1.372	-1.079	4.408
b_mass	-4.404	-3.943	4.839	-13.746	2.757
di_pf	-0.701	-0.551	1.561	-3.099	3.145
age	-2.924	-3.910	3.446	-7.787	2.630

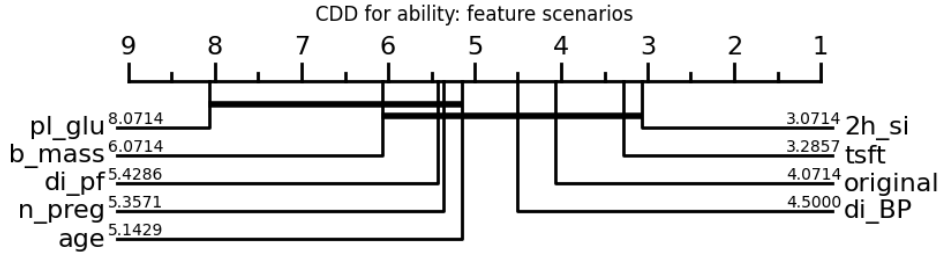
Source: Author's own calculation

Figure 22 – CDD of feature ranking: Contraceptive dataset with Simple imputation



Source: Author's own elaboration

Figure 23 – CDD of feature ranking: Pima dataset with Simple imputation



Source: Author's own elaboration

then n_chd and w_age second. w_rel, h_occ_4, m_exp, and h_edu do not show any statistically significant differences when compared to the original data without missing features. Abilities in the scenario where the feature w_wrk is missing seem to actually be higher than when no data is missing, which might indicate that this feature might introduce more noise than relevant information in the prediction task.

For the Pima dataset, the one features where the abilities had statistically significant changes when compared to the original scenario were the features pl_glu and b_mass.

5.3.3 Imputation analysis

We already showed a quick case study of the imputation for a single instance of the Contraceptive and for the Pima dataset, and commented on how each technique behaves when imputing the missing values and what we could expect from that. Taking this intuition in mind, here we present the overall values for the distance between the original values and the values imputed with each imputation technique. We chose the euclidean distance metric, as it is one of the simplest and easiest to understand. It is important to take into account that the scope of values of each feature is different; here we are showing the real values, not normalized nor standardized, so it is evident that the mean distance for the feature 'w_age' (age of the woman) of the Contraceptive dataset should naturally be higher than the difference for the feature 'n_chd' (number of children born).

Table 27 shows the euclidean distance between the original values of the Contraceptive dataset and the values imputed by each imputation technique, per feature. Table 28 shows the euclidean distance for the Pima dataset.

For both datasets the general results are similar. It can be seen that for this case, the KNN imputation technique is the one with the lower distance to the original data, for all features of the dataset. This makes sense also considering the instance analysis we discussed in Section 5.2, where the values imputed by the KNN imputer were the most similar (and sometimes even equal) to the original values. Of course, there are instances that will be very similar to the original values, while others can be imputed values that are quite different, to a point where one or more classifiers can assign a different class prediction for that instance. But in the ability change analysis that we have shown earlier, it is clear that the abilities change much less when the data is imputed with KNN imputer than when they are imputed with the Simple imputer. The Iterative imputer shows euclidean distances very similar to the Simple imputer. We believe that this might happen because this imputation technique is better when the missing data occurs over several features of the dataset, not on a single one at a time.

Table 27 – Euclidean distance per imputation for Contraceptive dataset

	Simple	KNN	Iterative
w_age	310.620	17.205	347.702
w_edu	55.453	18.439	38.807
h_edu	38.039	18.276	33.000
n_chd	90.366	21.977	103.663
w_rel	14.663	10.344	17.407
w_wrk	18.974	14.213	22.825
stl_i	49.940	19.596	46.336
m_exp	10.440	9.487	12.767
h_occ_2	20.494	12.728	17.607
h_occ_3	23.979	13.077	17.972
h_occ_4	5.196	5.196	5.916

Source: Author's own calculation

Table 28 – Euclidean distance per imputation for Pima dataset

	Simple	KNN	Iterative
n_preg	96.213	79.756	95.221
pl_glu	894.113	240.551	1093.571
di_BP	542.860	194.861	667.208
tsft	447.037	173.009	504.083
2h_si	3506.350	359.825	3827.313
b_mass	218.699	136.138	271.034
di_pf	9.227	9.638	11.954
age	346.253	184.366	346.730

Source: Author's own calculation

5.4 REGRESSION ANALYSIS

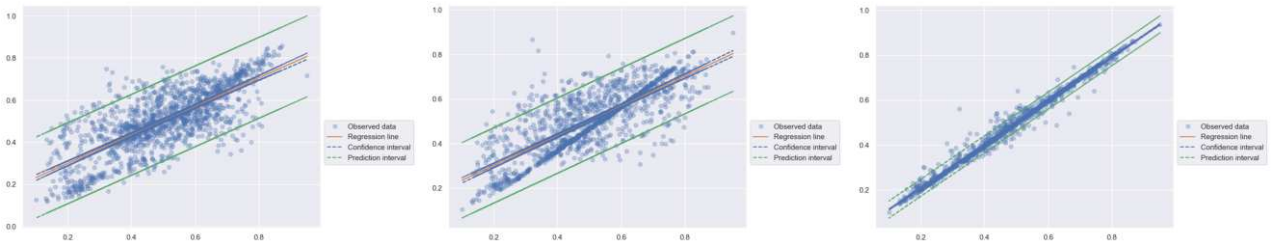
In order to understand how the difficulty changes in different scenarios, we run a regression task that takes as input the difficulty values of the original data and the corresponding difficulties of each missing feature scenario.

Figure 97 shows OLS model fit with the whole data for Contraceptive dataset using simple imputation and Figure 101, using the KNN imputation method. Figure 98 shows OLS model fit with the whole data for Pima dataset using simple imputation and Figure 102, using the KNN imputation method.

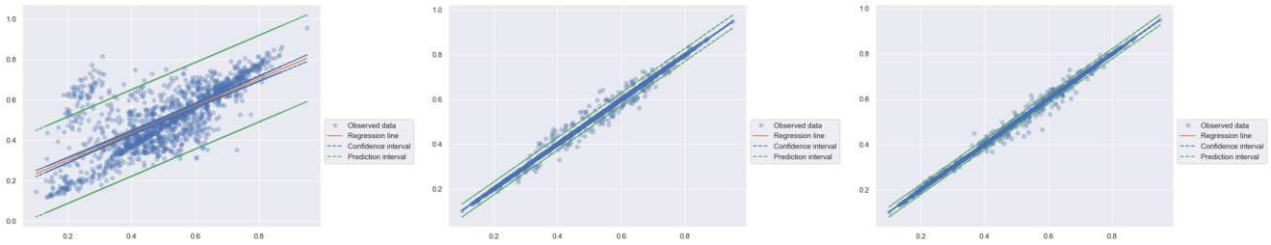
For both datasets, it can be observed that there is much more dispersion in the difficulty values when the imputation technique is the Simple Imputer. The KNN imputation shows a lot less dispersion, however the dispersion is still significant for some of the features. In the Contraceptive dataset, these features are w_edu and n_chd, and for the Pima dataset these features are pl_glu and b_mass.

Figure 24 – OLS model fit for Contraceptive dataset - Simple Imputation

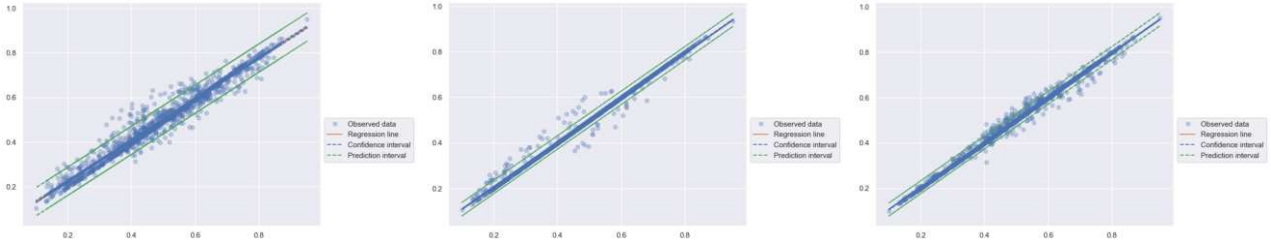
(i) w_age (ii) w_edu (iii) h_edu



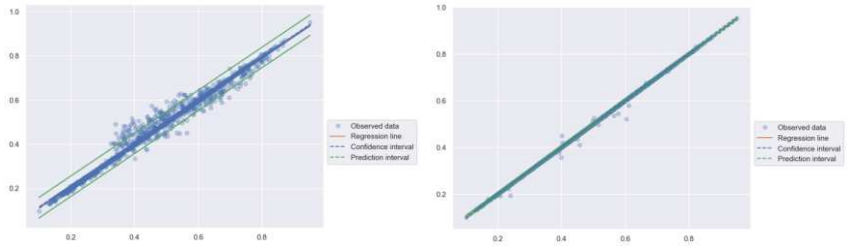
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



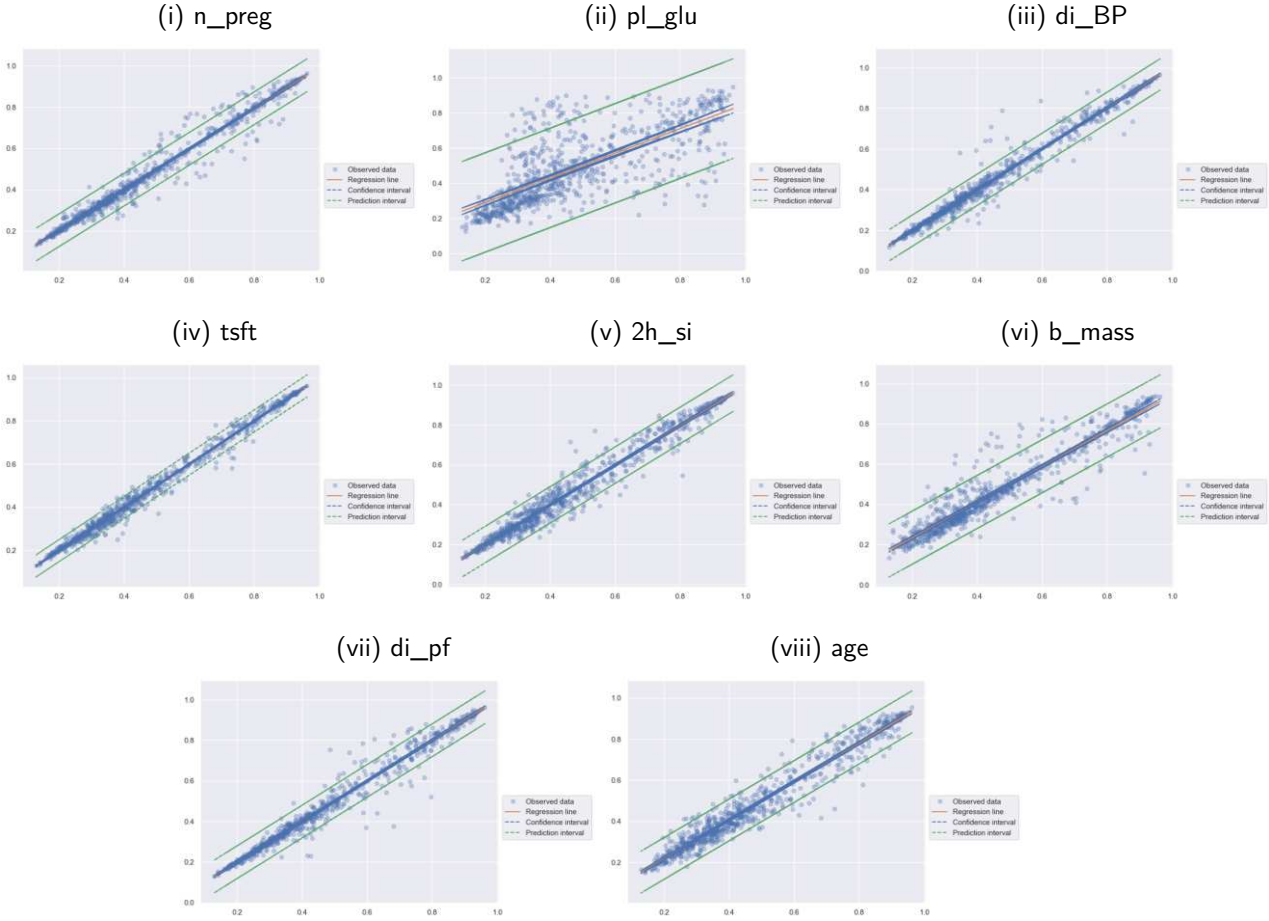
(x) h_occ_3 (xi) h_occ_4



Source: Author's own elaboration

Table 29 shows the ROC-AUC scores for each classifier for the Contraceptive dataset with Simple imputation. Table 30 shows the ROC-AUC scores for each classifier for the Pima dataset with Simple imputation. Both tables show the scores when binarizing the data considering the

Figure 25 – OLS model fit for Pima dataset - Simple Imputation



Source: Author's own elaboration

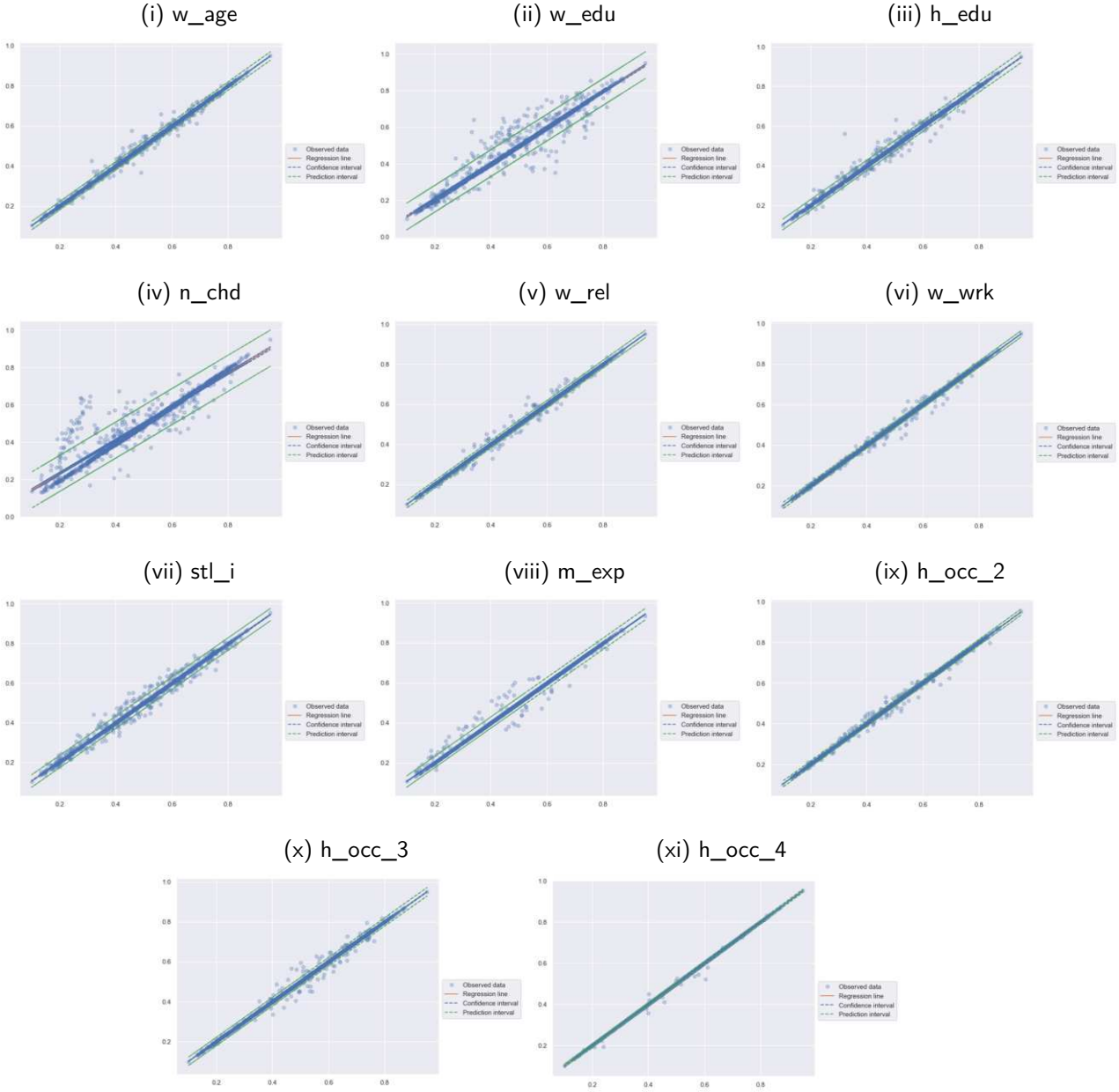
threshold of the data being above the confidence interval.

For the Contraceptive dataset, the ROC-AUC values are higher only for features such as h_occ_2 and h_occ_3, which means that it would be hard for a model to tell if a instance would require the missing data to be gathered.

for the Pima dataset, the most relevant feature, pl_glu is the one with the highest ROC-AUC score for all models, which means the models can tell with a good precision whether an instance requires the missing data to be gathered or just filled with an imputation technique.

The classification analysis for the other datasets can be seen in Appendix G. For all datasets, it can be seen that the ROC-AUC for the classifier models is usually above the baseline score, but in some cases the ROC-AUC is only slightly above this baseline (say, around 0.55). We believe that this happens because in some scenarios it is not so clear which are the instances that would benefit from gathering the real missing data when compared to having that value imputed instead. Although the OLS model gives a cleaner visualization of the data, it might

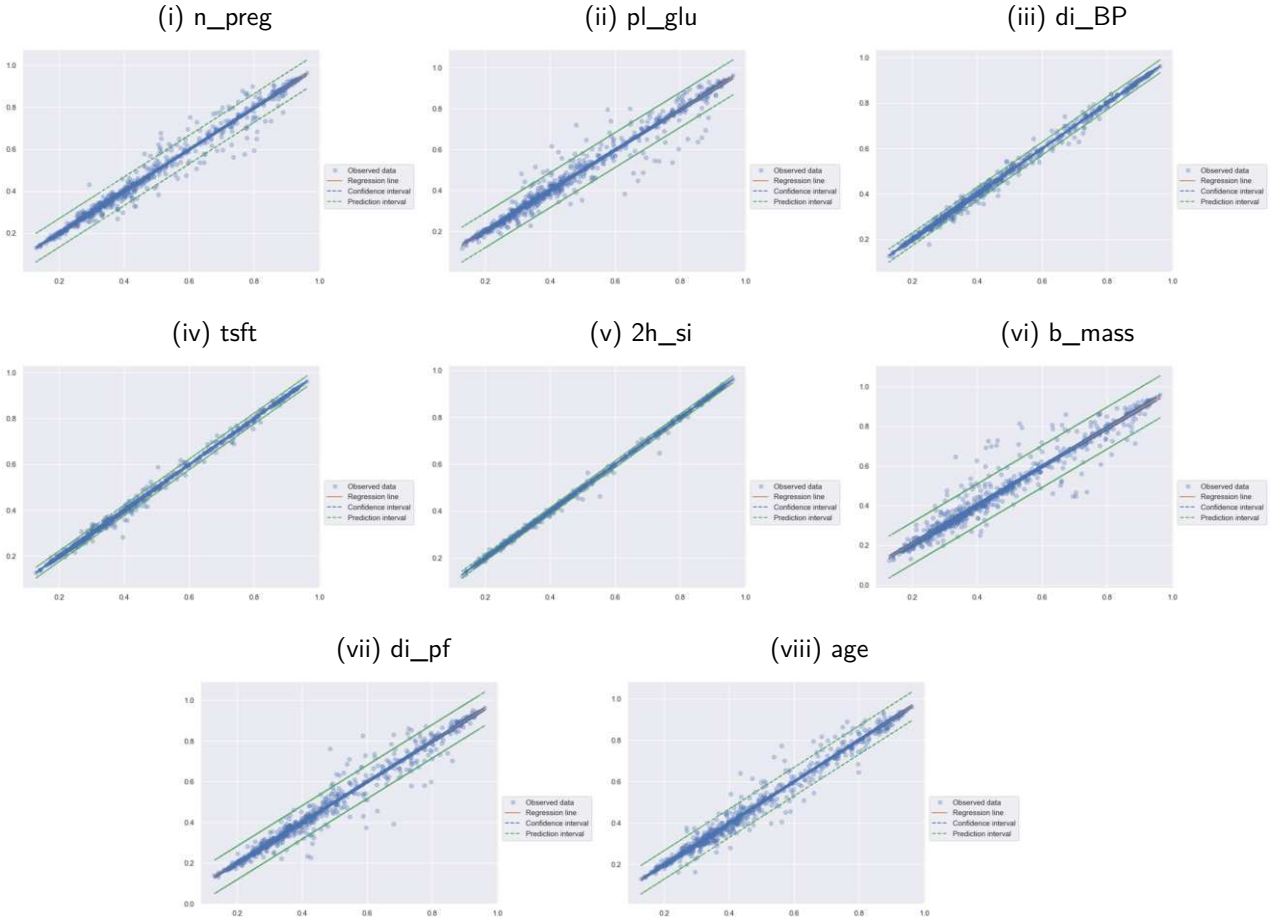
Figure 26 – OLS model fit for Contraceptive dataset - KNN Imputation



Source: Author's own elaboration

be better in terms of performance to use a more robust model such as a XGBoost regression model to get the predictions for the difficulty values before the binarization and classification tasks. We leave this analysis with different models to future works.

Figure 27 – OLS model fit for Pima dataset - KNN Imputation



Source: Author's own elaboration

Table 29 – OLS results for Contraceptive with Simple Imputation

	w_age	w_edu	h_edu	n_chd	w_rel	w_wrk	stl_i	m_exp	h_occ_2	h_occ_3	h_occ_4
log_reg	0.602	0.617	0.565	0.577	0.599	0.563	0.573	0.547	0.700	0.686	0.587
svc_rbf	0.603	0.615	0.582	0.549	0.498	0.491	0.522	0.524	0.715	0.686	0.571
svc_linear	0.602	0.619	0.563	0.540	0.457	0.489	0.512	0.538	0.691	0.668	0.563
rf	0.581	0.591	0.585	0.586	0.565	0.524	0.559	0.580	0.669	0.661	0.574
lgbm	0.607	0.602	0.597	0.615	0.570	0.567	0.598	0.592	0.680	0.684	0.600
catboost	0.626	0.619	0.608	0.625	0.604	0.572	0.606	0.598	0.711	0.704	0.612
xgboost	0.614	0.609	0.583	0.600	0.577	0.558	0.583	0.580	0.671	0.673	0.595
dummy_strat	0.502	0.491	0.497	0.504	0.500	0.481	0.511	0.490	0.495	0.513	0.496
dummy_prior	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500

Source: Author's own calculation

Table 30 – OLS results for Pima with Simple Imputation

	n_preg	pl_glu	di_BP	tsft	2h_si	b_mass	di_pf	age
log_reg	0.608	0.734	0.549	0.566	0.594	0.599	0.555	0.552
svc_rbf	0.431	0.712	0.506	0.600	0.603	0.623	0.489	0.620
svc_linear	0.600	0.729	0.499	0.561	0.584	0.584	0.549	0.506
rf	0.668	0.725	0.563	0.583	0.603	0.630	0.605	0.661
lgbm	0.651	0.726	0.551	0.585	0.560	0.617	0.585	0.641
catboost	0.683	0.745	0.569	0.589	0.595	0.621	0.619	0.665
xgboost	0.649	0.714	0.544	0.584	0.551	0.597	0.556	0.646
dummy_strat	0.483	0.474	0.482	0.467	0.512	0.474	0.476	0.511
dummy_prior	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500

Source: Author's own calculation

5.5 DISCUSSION

In this chapter we have shown the IRT results for classification tasks using a set of classifiers as respondents and framing instances of each dataset as the items in an IRT formulation. We discussed why traditional performance metrics are hard to choose and that they do not provide much information on each instance. We use a missing feature scenario for the test set where the values are imputed, one feature at a time, with three different imputation techniques. This approach helps us understand the impact of the missing features and at the the same time evaluate the quality of the imputation technique being evaluated. A case study with two datasets was used to help understand how IRT can give a better understanding of models performance on an instance basis. We use OLS regression to understand which instances are the most impacted by the missing of each of the features of the dataset. With this, we can predict how a new instance would behave in the missing scenario and can take a better decision about feature gathering for that specific instance: should the information on that missing feature be gathered or is imputation of the missing information enough to get a good classification result? The IRT analysis can help us understand how to deal with this kind of scenario.

6 CONCLUSION AND FUTURE WORK

In this dissertation we proposed BIFI, a workflow that uses β^3-IRT to measure feature relevance in classification datasets with missing data. BIFI results in two different outputs: ability change and difficulty change, which were used as a tool to assess the importance of each feature and to build a feature ranking metric whose results were compared to traditional feature selection techniques.

We used β^3-IRT to evaluate the performance of classifiers in a missing feature scenario, where the classification models are trained with the training set with all features, but tested in scenarios where one feature is missing at a time, with the corresponding values in the test set being imputed with values from the training set. A k-fold cross validation workflow is used so all instances will be in the test set at some point. The outputs of this workflow are used to evaluate the relevance of each feature for each dataset. This analysis with β^3-IRT is more powerful than other solutions because the results can be analyzed considering the changes in each instance of the dataset.

The results of the experiments with BIFI provided interesting insights about feature relevance in a missing feature scenario. We showed, experimentally, that the difficulty and discrimination parameters of an instance change according to the importance a feature has on that specific instance. Furthermore, choosing a more powerful imputation technique can help to tighten the gap between the scenarios of missing feature and having complete information about the instance. It was shown that for some instances, using a good imputation for a missing feature value can be a good compromise between not having that feature and the costs involved in gathering that information, if that option is available.

Results showed that decision trees and linear models are more sensitive to missing data, while models such as XGBoost and AdaBoost seem to be more robust. This information is relevant if one is planning on developing an ensemble with a subset of classifiers. Understanding the strengths and weaknesses of models for each dataset can be important when choosing the best classifiers to compose the ensemble.

Using OLS regression, we investigated the relationship between the original difficulty values of each instance and how this value changes in each missing feature scenario. Then, with our proposed dichotomization of these values based on the confidence interval and the prediction interval, we analyzed whether it was possible to predict the relevance of a feature for a given

instance based on the difficulty values of other instances used to train the model. The experimental results have shown that we can predict this with a good level of confidence, which means that we have a solution that can be used to analyze instances and their relationship with the features of a dataset during the deployment of the model, for instances that were not seen during the training steps.

6.1 IDEAS FOR FUTURE WORK

Further experiments can be made by using the instance difficulty and discrimination per scenario to analyze where, in the feature space, each feature has more impact; we believe that instances that have higher difficulty values are the ones closer to the decision boundary, and are probably the instances more affected by missing data in a relevant feature. It could also be useful to understand if it would be worth the effort to collect the data for a given instance, if the missing feature data is more relevant for any given instance of the dataset. To do this, we could apply the method proposed in Maguedong-Djoumessi and Hernández-Orallo (2013), Hernandez-Orallo et al. (2016) to BIFI. It would also be interesting to analyze how different imputation methods can affect the values of ability and difficulty of classifiers and instances in each missing feature scenario.

- Cost study of feature gathering

  In order to have a better evaluation of the feature gathering, different costs can be associated to each feature, and even different costs depending on other features. This can provide a more realistic view about the cost that it would take to gather each missing feature for each instance.

- Deeper study on instance analysis

  The experiments could be enriched by analyzing the changes in each instance with more depth. Each instance can be analyzed in terms of missing feature and imputation technique used. It is possible to make an analysis, for each instance, about how the performance of classifiers are affected by these different setups.

- Use of cost-sensitive models to evaluate the OLS regression analysis

  The cost of gathering each feature in a dataset can be different, and it can be even have different costs for different instances. The use of cost-sensitive models can help to have

a more detailed understanding of the costs of gathering missing data, depending on the importance that a given feature has for each individual instance of the dataset, as well as the cost (which could be in terms of money, time, access difficulty, etc.) associated with trying to collect the missing information when compared to imputing the missing value. We believe that a cost-sensitive analysis approach would be an important next step for this present work.

- Different missing data approaches

  In Section 2.2, we have shown that there are different types of missing data. In our proposed solution, we only used one type of missing data, where the data is MNAR. We could insert different types of missing data in the datasets and evaluate how the results would change with this modification.

- Using this workflow to assess the relevance of features in regression tasks

  Using the formulation proposed in Moraes et al. (2020), it would also be possible to evaluate the relevance of each feature for regression tasks, with few changes to the experimental setup. Although in this paper the authors chose to use PCA as a dimensionality reduction tool for the datasets in order to simplify the analysis of noise impact, if the whole dataset is used, we believe that the application of the proposed workflow to regression tasks is straight-forward.

REFERENCES

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. v. 13, n. 10, p. 281–305, 2012. Available at: <http://jmlr.org/papers/v13/bergstra12a.html>.

BIRNBAUM, A. L. Some latent trait models and their use in inferring an examinee's ability. 1968. Publisher: Addison-Wesley. Available at: <https://ci.nii.ac.jp/naid/10011544105/en/>.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. Association for Computing Machinery, 1992. (COLT '92), p. 144–152. ISBN 0-89791-497-X. Event-place: Pittsburgh, Pennsylvania, USA. Available at: <https://doi.org/10.1145/130385.130401>.

BREIMAN, L. Random forests. v. 45, n. 1, p. 5–32, 2001. ISSN 1573-0565. Available at: <https://doi.org/10.1023/A:1010933404324>.

BUUREN, S. V.; BRAND, J. P.; GROOTHUIS-OUDSHOORN, C. G.; RUBIN, D. B. Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, Taylor Francis, v. 76, n. 12, p. 1049–1064, 2006. Available at: <https://doi.org/10.1080/10629360600810434>.

BUUREN, S. van; GROOTHUIS-OUDSHOORN, K. mice: Multivariate Imputation by Chained Equations in R. v. 45, n. 3, p. 67, 2011. Edition: 2011-12-12 ISBN: 1548-7660 Type: 10.18637/jss.v045.i03. Available at: <https://www.jstatsoft.org/v045/i03>.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. v. 40, n. 1, p. 16–28, 2014. ISBN: 0045-7906 Publisher: Elsevier.

CHEN, T.; GUESTRIN, C. XGBoost: A scalable tree boosting system. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016. (KDD '16), p. 785–794. ISBN 978-1-4503-4232-2. Available at: <http://doi.acm.org/10.1145/2939672.2939785>.

CHEN, Y.; FILHO, T. S.; PRUDÊNCIO, R. B.; DIETHE, T.; FLACH, P. β^3-IRT: A new item response model and its applications. In: *The 22nd International Conference on Artificial Intelligence and Statistics*. [S.l.]: PMLR, 2019. p. 1013–1021. ISBN 2640-3498.

CHEN, Z.; AHN, H. Item response theory based ensemble in machine learning. v. 17, n. 5, p. 621–636, 2020. ISSN 1751-8520. Available at: <https://doi.org/10.1007/s11633-020-1239-y>.

COVER, T. M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, EC-14, n. 3, p. 326–334, 1965.

COVER, T. M.; HART, P. E. Nearest neighbor pattern classification. v. 13, n. 1, p. 21–27, 1967. ISSN 1557-9654.

DING, C.; PENG, H. Minimum redundancy feature selection from microarray gene expression data. In: *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*. [S.l.: s.n.], 2003. p. 523–528.

DUA, D.; GRAFF, C. *UCI Machine Learning Repository*. 2017. Available at: <http://archive.ics.uci.edu/ml>.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. v. 55, n. 1, p. 119–139, 1997. ISSN 0022-0000. Available at: <https://www.sciencedirect.com/science/article/pii/S002200009791504X>.

FRIEDMAN, J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, v. 29, n. 5, p. 1189–1232, Oct. 2001. ISSN 0090-5364, 2168-8966. Available at: <https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full>.

GEURTS, P.; ERNST, D.; WEHENKEL, L. Extremely randomized trees. v. 63, n. 1, p. 3–42, 2006. ISSN 1573-0565. Available at: <https://doi.org/10.1007/s10994-006-6226-1>.

HAMBLETON, R. K.; SWAMINATHAN, H.; ROGERS, H. J. *Fundamentals of item response theory*. [S.l.]: Sage Publications, 1991. ISBN 0-8039-3646-X 978-0-8039-3646-1 0-8039-3647-8 978-0-8039-3647-8.

HAND, D. J.; YU, K. Idiot's bayes: Not so stupid after all? v. 69, n. 3, p. 385–398, 2001. ISSN 03067734, 17515823. Publisher: [Wiley, International Statistical Institute (ISI)]. Available at: <http://www.jstor.org/stable/1403452>.

HERNANDEZ-ORALLO, J.; MARTÍNEZ-USÓ, A.; PRUDÊNCIO, R.; KULL, M.; FLACH, P.; AHMED, C.; LACHICHE, N. Reframing in context: A systematic approach for model reuse in machine learning. v. 29, p. 1–16, 2016.

KANDANAARACHCHI, S.; SMITH-MILES, K. *Comprehensive Algorithm Portfolio Evaluation using Item Response Theory*. [S.l.: s.n.], 2020.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. v. 30. Available at: <https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>.

KE, G.; MENG, Q.; FINLEY, T.; WANG, T.; CHEN, W.; MA, W.; YE, Q.; LIU, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In: GUYON, I.; LUXBURG, U. V.; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. v. 30. Available at: <https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf>.

KLINE, A.; KLINE, T.; ABAD, Z. S. H.; LEE, J. Novel feature selection for artificial intelligence using item response theory for mortality prediction. In: *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. [S.l.: s.n.], 2020. p. 5729–5732. ISBN 2694-0604. Journal Abbreviation: 2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC).

KURSA, M. B.; RUDNICKI, W. R. Feature Selection with the Boruta Package. *Journal of Statistical Software*, v. 36, n. 11, p. 1–13, 2010. Available at: <https://www.jstatsoft.org/index.php/jss/article/view/v036i11>.

LALOR, J. P.; WU, H.; YU, H. Building an evaluation scale using item response theory. v. 2016, p. 648–657, 2016. Available at: <https://pubmed.ncbi.nlm.nih.gov/28004039>.

LORD, F.; NOVICK, M.; BIRNBAUM, A. *Statistical theories of mental test scores*. [S.l.]: Addison-Wesley, 1968. (Statistical theories of mental test scores.).

MAGUEDONG-DJOUMESSI, C. P.; HERNÁNDEZ-ORALLO, J. Test cost and misclassification cost trade-off using reframing. abs/1305.7111, 2013. Available at: <http://arxiv.org/abs/1305.7111>.

MARTÍNEZ-PLUMED, F.; HERNÁNDEZ-ORALLO, J. AI results for the atari 2600 games : difficulty and discrimination using IRT. In: . [S.l.: s.n.], 2017.

MARTÍNEZ-PLUMED, F.; PRUDÊNCIO, R. B. C.; MARTÍNEZ-USÓ, A.; HERNÁNDEZ-ORALLO, J. Making sense of item response theory in machine learning. In: *Proceedings of the Twenty-Second European Conference on Artificial Intelligence*. IOS Press, 2016. (ECAI'16), p. 1140–1148. ISBN 978-1-61499-671-2. Event-place: The Hague, The Netherlands. Available at: <https://doi.org/10.3233/978-1-61499-672-9-1140>.

MARTÍNEZ-PLUMED, F.; PRUDÊNCIO, R. B. C.; MARTÍNEZ-USÓ, A.; HERNÁNDEZ-ORALLO, J. Item response theory in AI: Analysing machine learning classifiers at the instance level. v. 271, p. 18–42, 2019. ISSN 0004-3702. Available at: <https://www.sciencedirect.com/science/article/pii/S0004370219300220>.

MASTERS, G. N. A rasch model for partial credit scoring. v. 47, n. 2, p. 149–174, 1982. ISSN 1860-0980. Available at: <https://doi.org/10.1007/BF02296272>.

MCKNIGHT, P. E.; MCKNIGHT, K. M.; SIDANI, S.; FIGUEREDO, A. J. *Missing data: A gentle introduction*. [S.l.]: Guilford Press, 2007.

MIAO, J.; NIU, L. A survey on feature selection. v. 91, p. 919–926, 2016. ISBN: 1877-0509 Publisher: Elsevier.

MORAES, J. V. C.; REINALDO, J. T. S.; PRUDENCIO, R. B. C.; FILHO, T. M. S. Item response theory for evaluating regression algorithms. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2020. p. 1–8. ISBN 2161-4407. Journal Abbreviation: 2020 International Joint Conference on Neural Networks (IJCNN).

MURAKI, E. A GENERALIZED PARTIAL CREDIT MODEL: APPLICATION OF AN EM ALGORITHM. v. 1992, n. 1, p. i–30, 1992. ISSN 23308516. Available at: <https://onlinelibrary.wiley.com/doi/10.1002/j.2333-8504.1992.tb01436.x>.

NOEL, Y.; DAUVIER, B. A beta item response model for continuous bounded responses. v. 31, n. 1, p. 47–73, 2007. ISSN 0146-6216. Publisher: SAGE Publications Inc. Available at: <https://doi.org/10.1177/0146621605287691>.

OLIVEIRA, C.; TENÓRIO, C. C.; PRUDÊNCIO, R. Item response theory to estimate the latent ability of speech synthesizers. In: *ECAI*. [S.l.: s.n.], 2020.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PROKHORENKOVA, L.; GUSEV, G.; VOROBEV, A.; DOROGUSH, A. V.; GULIN, A. CatBoost: Unbiased boosting with categorical features. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. [S.l.]: Curran Associates Inc., 2018. (NIPS'18), p. 6639–6649. Event-place: Montréal, Canada.

QUINLAN, J. R. Induction of decision trees. v. 1, n. 1, p. 81–106, 1986. ISSN 1573-0565. Available at: <https://doi.org/10.1007/BF00116251>.

RIZOPOULOS, D. ltm: An r package for latent variable modeling and item response analysis. v. 17, n. 5, p. 25, 2006. Edition: 2006-09-01 ISBN: 1548-7660 Type: 10.18637/jss.v017.i05. Available at: <https://www.jstatsoft.org/v017/i05>.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. v. 65 6, p. 386–408, 1958.

RUBIN, D. B. Inference and missing data. *Biometrika*, Oxford University Press, v. 63, n. 3, p. 581–592, 1976.

SAMEJIMA, F. Estimation of latent ability using a response pattern of graded scores. 1969.

SAMEJIMA, F. A general model for free-response data. *Psychometrika Monograph Supplement*, 1972.

SAMEJIMA, F. Normal ogive model on the continuous response level in the multidimensional latent space. v. 39, n. 1, p. 111–121, 1974. ISSN 1860-0980. Available at: <https://doi.org/10.1007/BF02291580>.

SEABOLD, S.; PERKTOLD, J. statsmodels: Econometric and statistical modeling with python. In: *9th Python in Science Conference*. [S.l.: s.n.], 2010.

SONG, F.; GUO, Z.; MEI, D. Feature selection using principal component analysis. In: IEEE. *2010 international conference on system science, engineering design and manufacturing informatization*. [S.l.], 2010. v. 1, p. 27–30.

WILLIAMS, C.; RASMUSSEN, C. Gaussian processes for regression. In: TOURETZKY, D.; MOZER, M. C.; HASSELMO, M. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1996. v. 8. Available at: <https://proceedings.neurips.cc/paper/1995/file/7cce53cf90577442771720a370c3c723-Paper.pdf>.

APPENDIX A – EXECUTION ENVIRONMENT

A.1 MODEL HYPERPARAMETER SEARCH SPACE

The search space for the classification models is shown below. The random state parameter was set to the fixed value of 42, whenever available.

- AdaBoost
  - n_estimators: [50, 100, 500],
  - learning_rate: [0.1, 1, 10],
- Decision Tree
  - criterion: ['gini', 'entropy'],
  - max_depth: [3, 5, 7, None],
  - max_features: ['sqrt', 'log2', None],
- Extra Trees
  - max_depth: [3, 5, 7, None],
  - n_estimators: [100, 200, 500],
  - max_features: ['sqrt', 'log2', None],
- Gradient Boosting
  - loss: ['deviance'],
  - n_estimators: [50, 100, 500],
  - max_features: ['sqrt', 'log2', None],
  - max_depth: [3, 5],
  - min_samples_leaf: [1],
  - min_samples_split: [2]
- Gaussian Naive Bayes
  - priors: [None],

  - var_smoothing: [1e-8, 1e-9, 1e-10]

- Gaussian Process
  - kernel: [1.0*RBF(1.0), 1.0*Matern(1.0)],

- KNN
  - n_neighbors:[2, 3, 4, 5, 6, 7, 8, 9],
  - weights: ['uniform', 'distance'],

- MLP
  - hidden_layer_sizes: [(50,50,50), (50,100,50), (100,)],
  - activation: ['tanh', 'relu'],
  - solver: ['sgd', 'adam'],
  - alpha: [0.0001, 0.05],
  - learning_rate: ['constant', 'adaptive'],
  - max_iter: [1000, 5000, 10000],
  - random_state: [42],

- QDA
  - reg_param: [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]

- Random Forest
  - n_estimators: [100, 200, 500],
  - max_features: ['sqrt', 'log2', None], can use 'None' too
  - max_depth: [3, 5, 7, None],
  - criterion: ['gini', 'entropy'],
  - random_state: [42],

- SVC
  - kernel: ['rbf'],
  - probability: [True],

  - random_state: [42],

- XGBoost
  - min_child_weight: [1, 5, 10],
  - gamma: [0.5, 1, 1.5, 2, 5],
  - subsample: [0.6, 0.8, 1.0],
  - colsample_bytree: [0.6, 0.8, 1.0],
  - max_depth: [3, 4, 5],
  - random_state: [42],

- CatBoost
  - depth: [3,1,2,6,4,5,7,8,9,10],
  - iterations: [250,100,500,1000],
  - learning_rate: [0.03,0.001,0.01,0.1,0.2,0.3],
  - l2_leaf_reg: [3,1,5,10,100],

- LightGBM
  - boosting_type: ['gbdt', 'dart', 'goss'],
  - min_child_weight: [1, 5, 10],
  - subsample: [0.6, 0.8, 1.0],
  - colsample_bytree: [0.6, 0.8, 1.0],
  - max_depth: [3, 4, 5],
  - verbosity: [0],
  - random_state: [42],

A.2 PYTHON LIBRARIES

The following Python libraries and corresponding versions are needed in the execution environment to run the experiments described in this dissertation:

- absl-py==0.12.0

- anyio==2.2.0
- appnope==0.1.2
- argon2-cffi==20.1.0
- astunparse==1.6.3
- async-generator==1.10
- attrs==20.3.0
- autorank==1.1.2
- Babel==2.9.0
- backcall==0.2.0
- baycomp==1.0.2
- birt-gd==0.1.3
- birt-sgd==0.1.28
- bleach==3.3.0
- Boruta==0.3
- cachetools==4.2.1
- catboost==1.0.0
- category-encoders==2.2.2
- certifi==2020.6.20
- cffi==1.14.5
- chardet==4.0.0
- click==8.0.2
- cycler==0.10.0
- decorator==5.0.5

- defusedxml==0.7.1
- entrypoints==0.3
- flatbuffers==1.12
- fonttools==4.28.1
- gast==0.3.3
- google-auth==1.28.0
- google-auth-oauthlib==0.4.4
- google-pasta==0.2.0
- graphviz==0.17
- grpcio==1.32.0
- h5py==2.10.0
- idna==2.10
- imbalanced-learn==0.8.1
- import-ipynb==0.1.3
- ipykernel==5.5.3
- ipython==7.22.0
- ipython-genutils==0.2.0
- ipywidgets==7.6.3
- jedi==0.18.0
- Jinja2==2.11.3
- joblib==1.0.1
- json5==0.9.5
- jsonschema==3.2.0

- jupyter==1.0.0
- jupyter-client==6.1.12
- jupyter-console==6.4.0
- jupyter-core==4.7.1
- jupyter-packaging==0.7.12
- jupyter-server==1.5.1
- jupyterlab==3.0.12
- jupyterlab-pygments==0.1.2
- jupyterlab-server==2.4.0
- jupyterlab-widgets==1.0.0
- Keras-Preprocessing==1.1.2
- kiwisolver==1.3.1
- lightgbm==3.3.0
- Markdown==3.3.4
- MarkupSafe==1.1.1
- matplotlib==3.5.0
- missingpy==0.2.0
- mistune==0.8.4
- mpmath==1.2.1
- mrmr @ git+https://github.com/smazzanti/mrmr@74987fc32319edb838a2f864b16c509dc351ea9a
- nbclassic==0.2.6
- nbclient==0.5.3
- nbconvert==6.0.7

- nbformat==5.1.3
- nest-asyncio==1.5.1
- networkx==2.6.3
- notebook==6.3.0
- numpy==1.19.5
- oauthlib==3.1.0
- opt-einsum==3.3.0
- packaging==20.9
- pandas==1.2.3
- pandocfilters==1.4.3
- parso==0.8.2
- patsy==0.5.1
- pexpect==4.8.0
- pickleshare==0.7.5
- Pillow==8.1.2
- plotly==5.3.1
- prometheus-client==0.10.0
- prompt-toolkit==3.0.18
- protobuf==3.15.6
- ptyprocess==0.7.0
- pyasn1==0.4.8
- pyasn1-modules==0.2.8
- pycparser==2.20

- pydot==1.4.2
- pydotplus==2.0.2
- Pygments==2.8.1
- pyparsing==2.4.7
- pyrsistent==0.17.3
- python-dateutil==2.8.1
- pytictoc==1.5.1
- pytz==2021.1
- pyzmq==22.0.3
- qtconsole==5.0.3
- QtPy==1.9.0
- requests==2.25.1
- requests-oauthlib==1.3.0
- rsa==4.7.2
- scikit-learn==0.24.1
- scipy==1.6.2
- seaborn==0.11.1
- Send2Trash==1.5.0
- setuptools-scm==6.3.2
- six==1.15.0
- sklearn==0.0
- sniffio==1.2.0
- statsmodels==0.12.2

- sympy==1.7.1
- tenacity==8.0.1
- tensorboard==2.4.1
- tensorboard-plugin-wit==1.8.0
- tensorflow==2.4.1
- tensorflow-estimator==2.4.0
- termcolor==1.1.0
- terminado==0.9.4
- testpath==0.4.4
- threadpoolctl==2.1.0
- tomli==1.2.2
- tornado==6.1
- tqdm==4.59.0
- traitlets==5.0.5
- typing-extensions==3.7.4.3
- urllib3==1.26.4
- wcwidth==0.2.5
- webencodings==0.5.1
- Werkzeug==1.0.1
- widgetsnbextension==3.5.1
- wrapt==1.12.1
- xgboost==1.2.1

APPENDIX B – DIFFICULTY DISPERSION PER SCENARIO

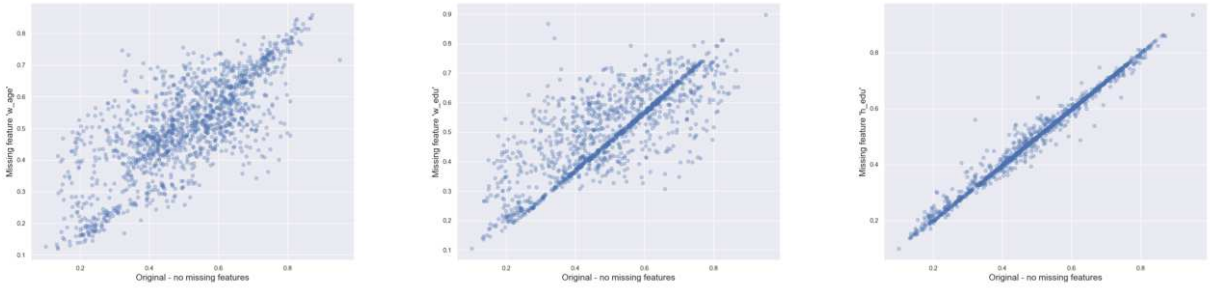
In this chapter we show the one-to-one difficulty values per scenario, for each dataset. We split the graphs for each imputation type: simple, KNN, and iterative.

B.1 SIMPLE

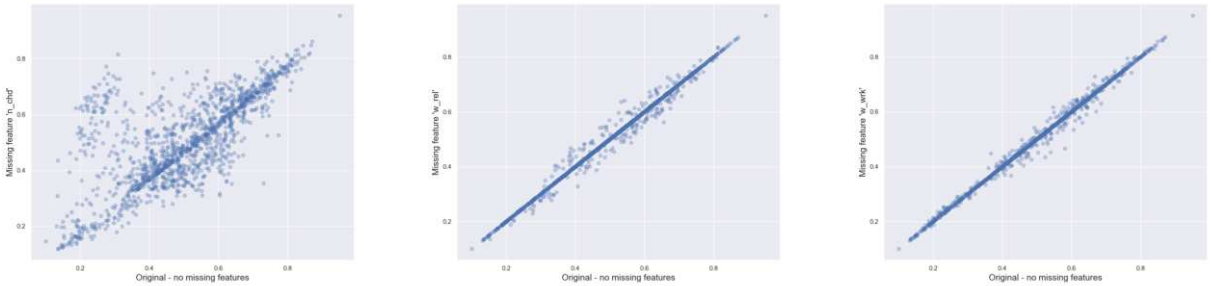
Figure 28 shows the results for the Contraceptive dataset, Figure 29 for the Pima dataset, Figure 30 for the Wheat dataset, Figure 31 for the Haberman dataset, Figure 32 for the Cleveland dataset, Figure 32 for the Ionosphere dataset, Figure 31 for the Wine dataset, and finally, Figure 31 shows the results for the Yeast dataset.

Figure 28 – Difficulty dispersion for the Contraceptive dataset - Simple Imputation

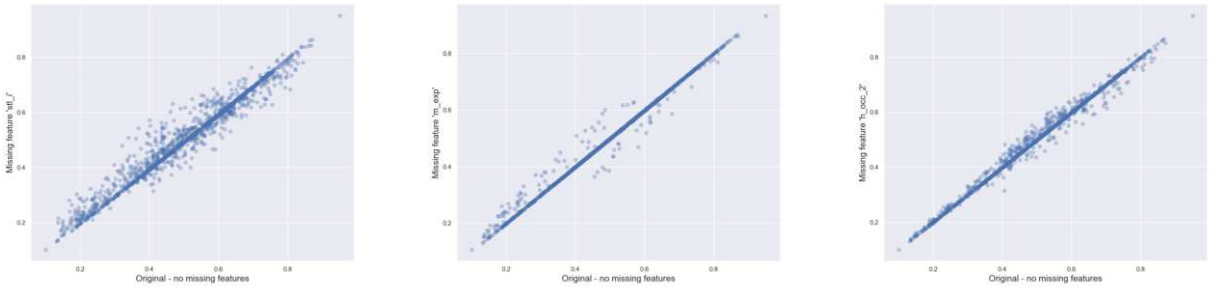
(i) w_age (ii) w_edu (iii) h_edu



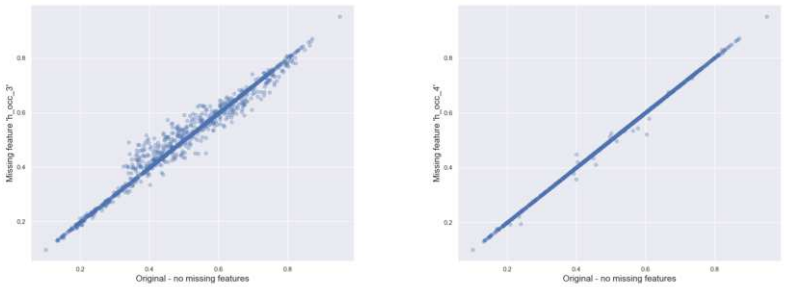
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



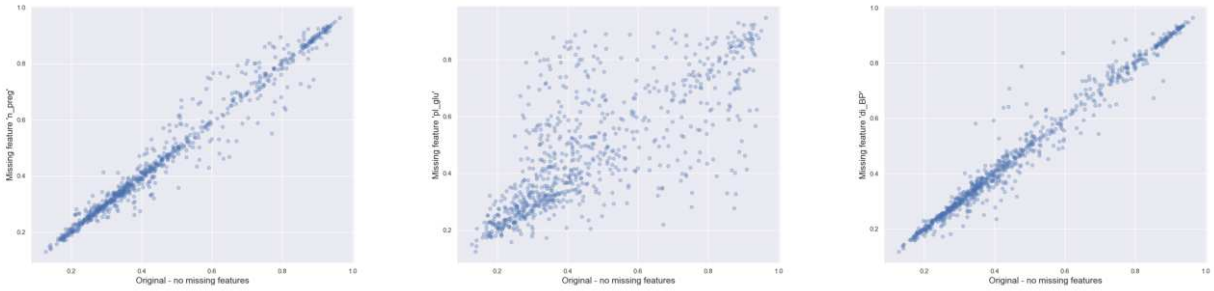
(x) h_occ_3 (xi) h_occ_4



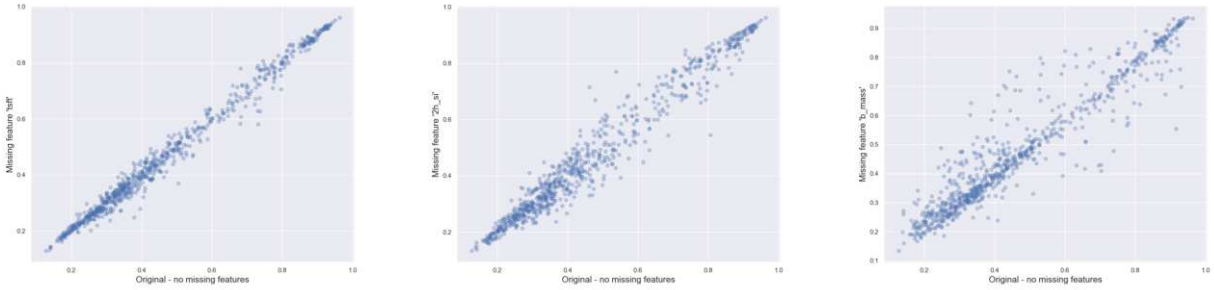
Source: Author's own elaboration

Figure 29 – Difficulty dispersion for the Pima dataset - Simple Imputation

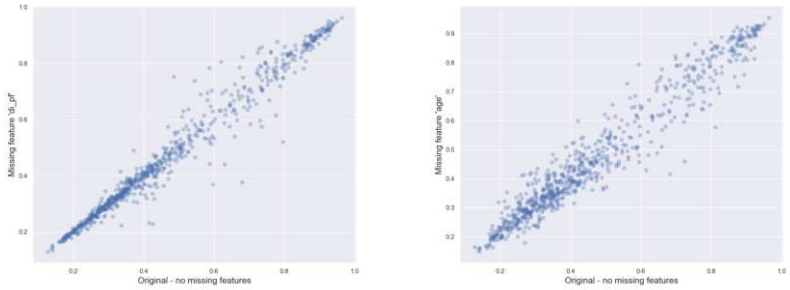
(i) n_preg (ii) pl_glu (iii) di_BP



(iv) tsft (v) 2h_si (vi) b_mass



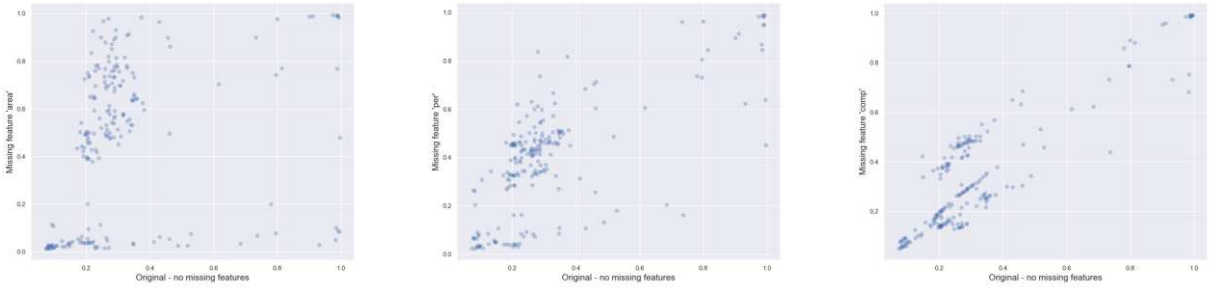
(vii) di_pf (viii) age



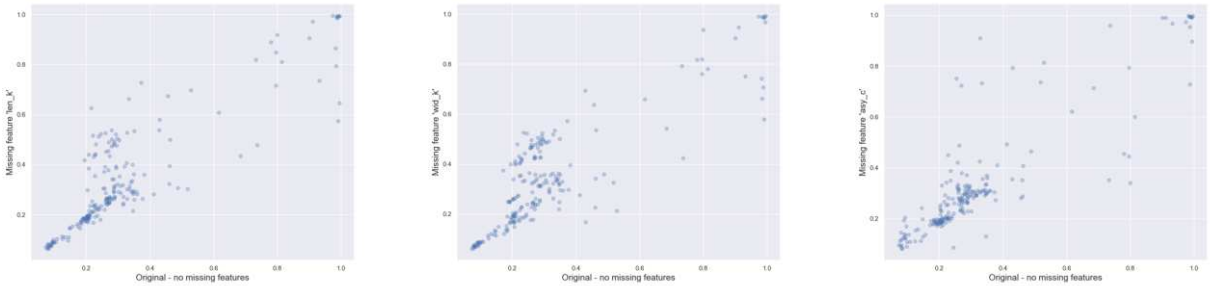
Source: Author's own elaboration

Figure 30 – Difficulty dispersion for the Wheat dataset - Simple Imputation

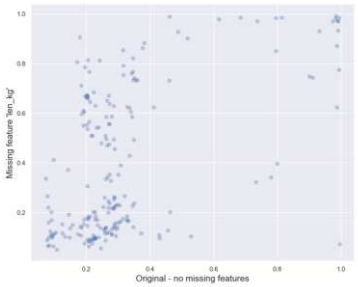
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



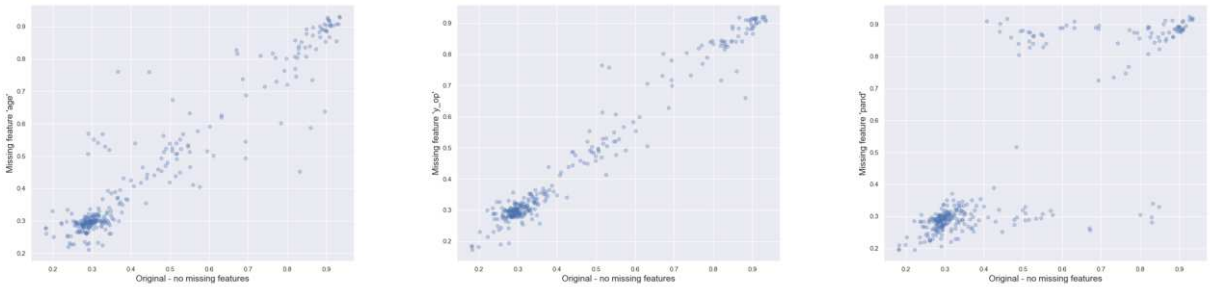
(vii) len_kg



Source: Author's own elaboration

Figure 31 – Difficulty dispersion for the Haberman dataset - Simple Imputation

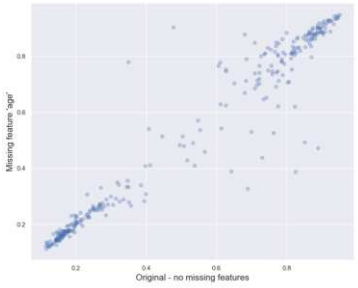
(i) age (ii) y_op (iii) pand



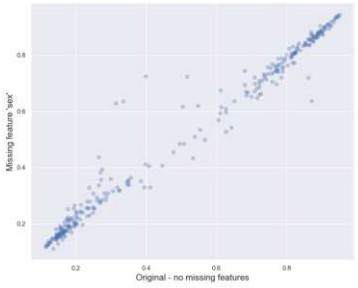
Source: Author's own elaboration

Figure 32 – Difficulty dispersion for the Cleveland dataset - Simple Imputation

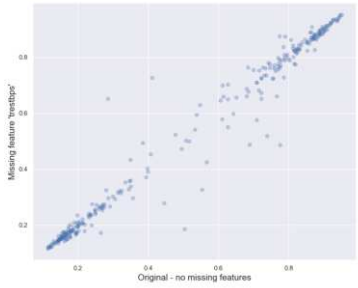
(i) age



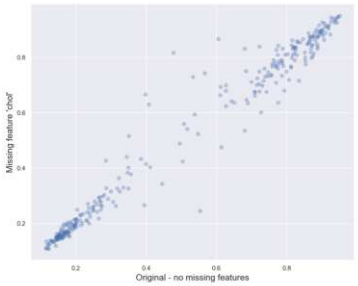
(ii) sex



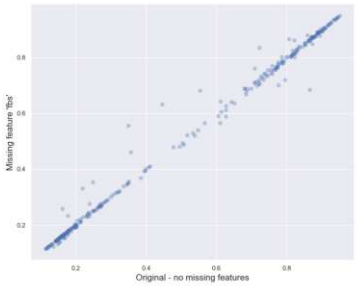
(iii) trestbps



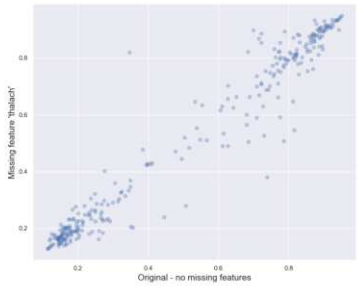
(iv) chol



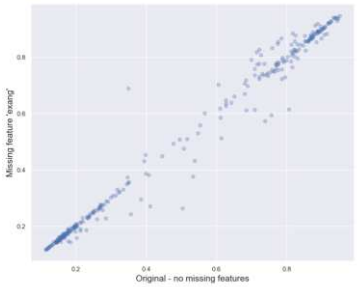
(v) fbs



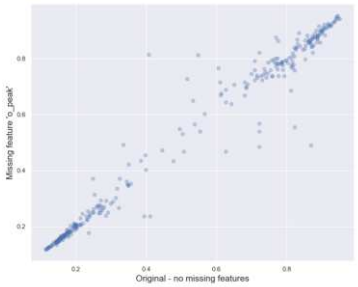
(vi) thalach



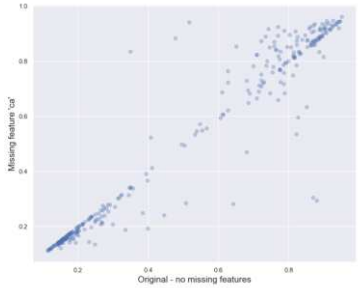
(vii) exang



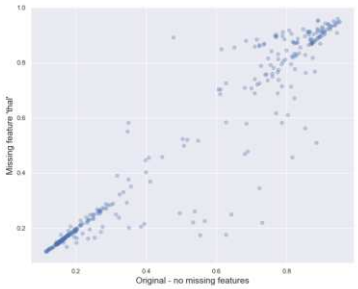
(viii) o_peak



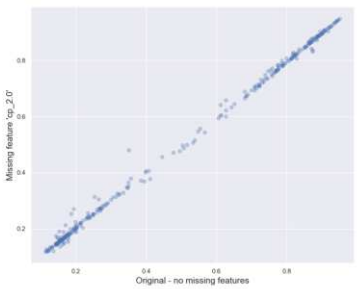
(ix) ca



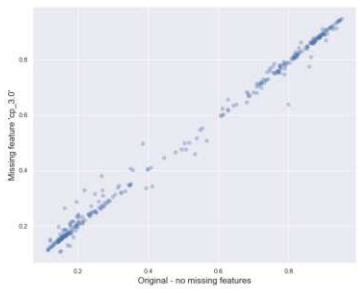
(x) thal

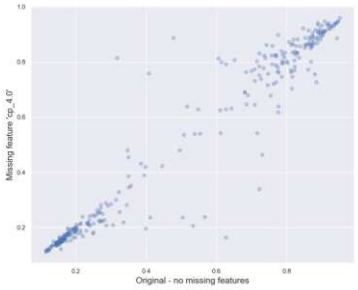


(xi) cp_2.0

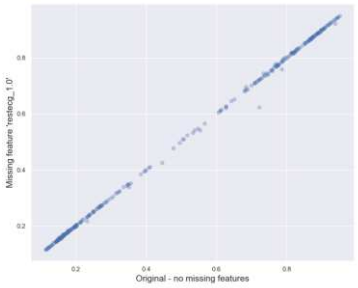


(xii) cp_3.0

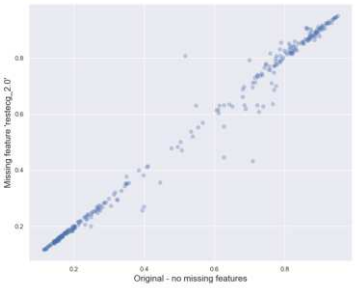




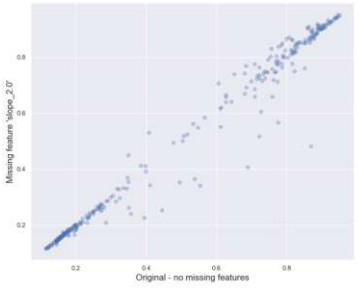
(xiii) cp_4.0



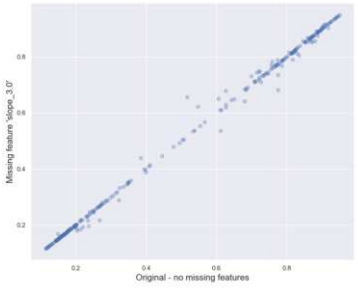
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

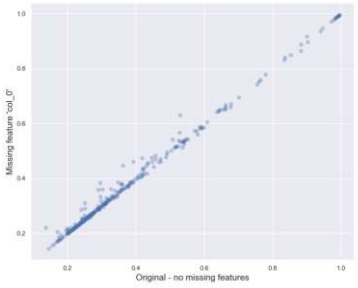


(xvii) slope_3.0

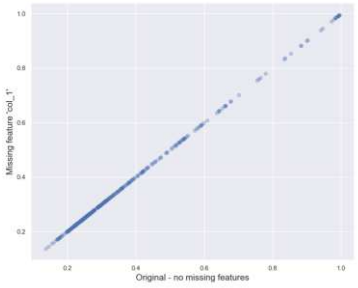
Source: Author's own elaboration

Figure 32 – Difficulty dispersion for the Ionosphere dataset - Simple Imputation

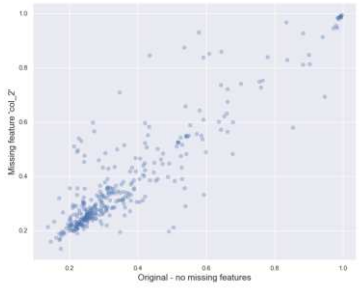
(i) col_0



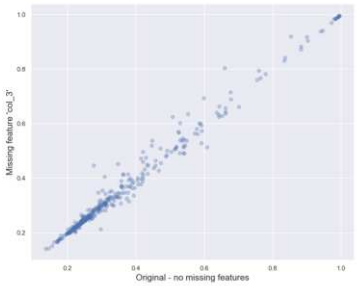
(ii) col_1



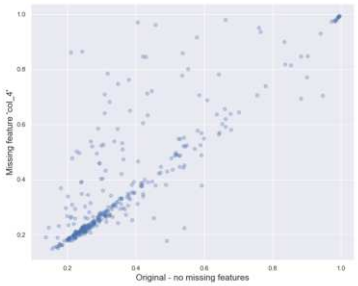
(iii) col_2



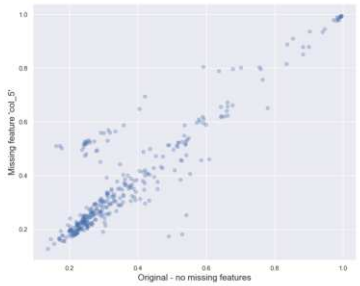
(iv) col_3



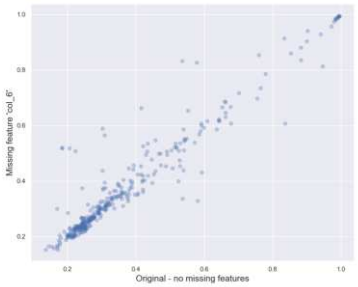
(v) col_4



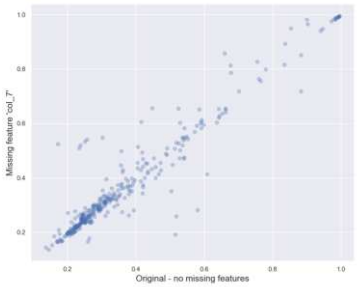
(vi) col_5



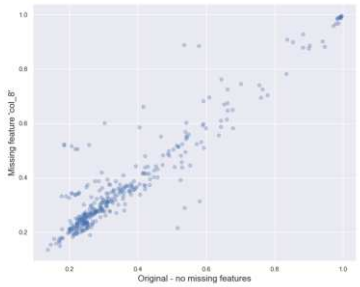
(vii) col_6



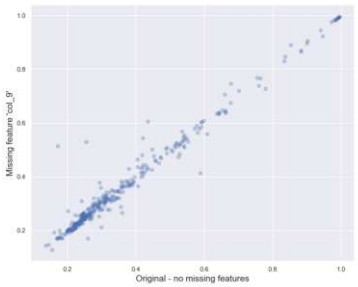
(viii) col_7



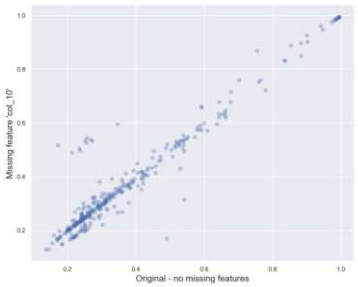
(ix) col_8



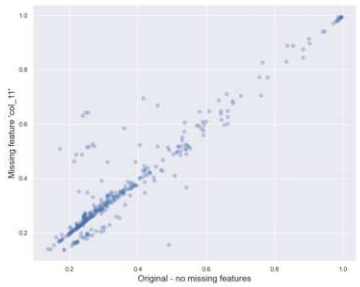
(x) col_9

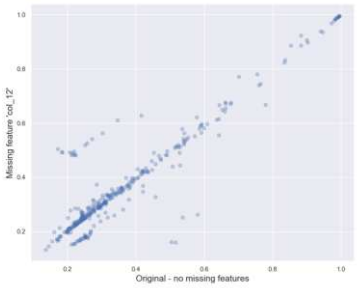


(xi) col_10

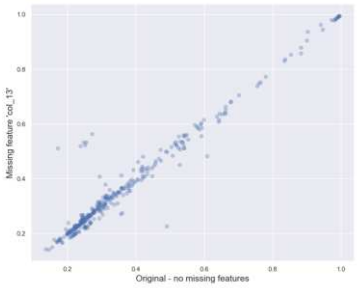


(xii) col_11

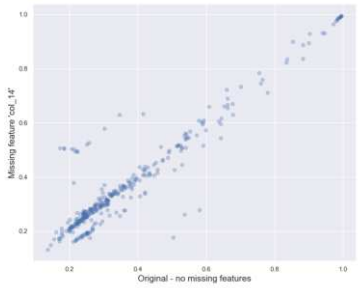




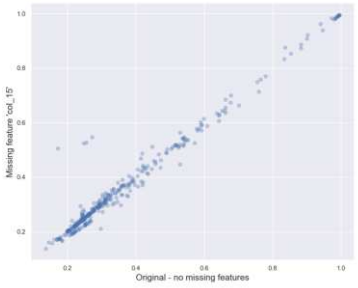
(xiii) col_12



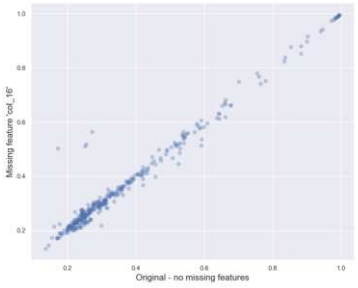
(xiv) col_13



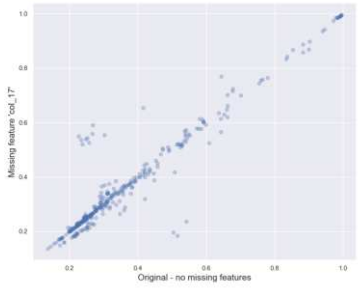
(xv) col_14



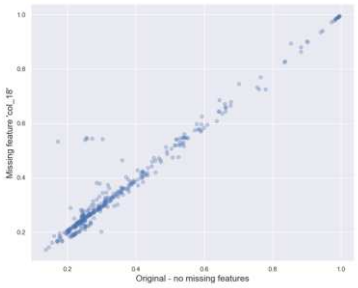
(xvi) col_15



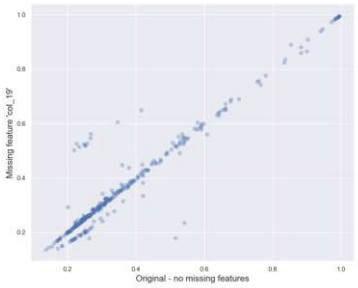
(xvii) col_16



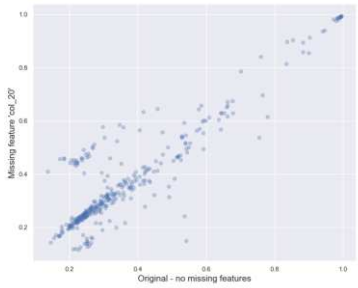
(xviii) col_17



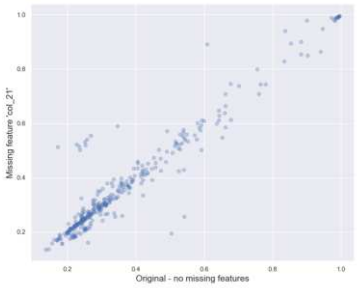
(xix) col_18



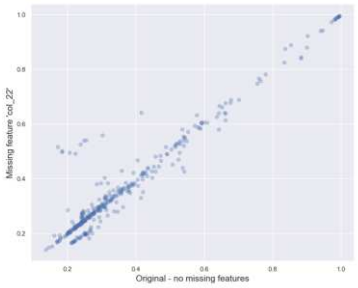
(xx) col_19



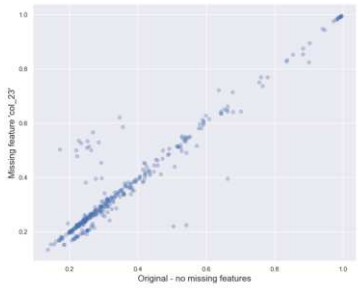
(xxi) col_20



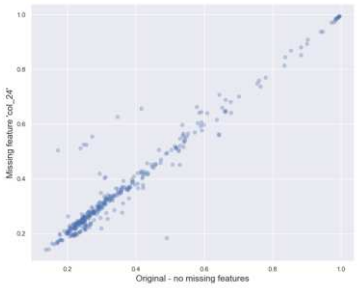
(xxii) col_21



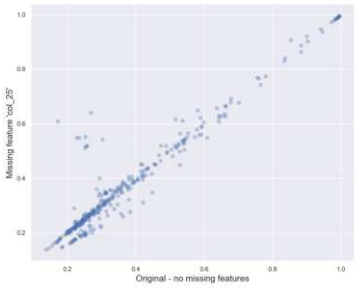
(xxiii) col_22



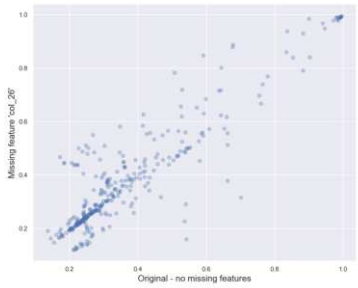
(xxiv) col_23



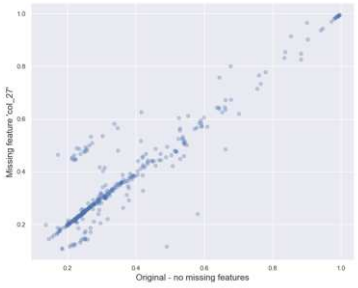
(xxv) col_24



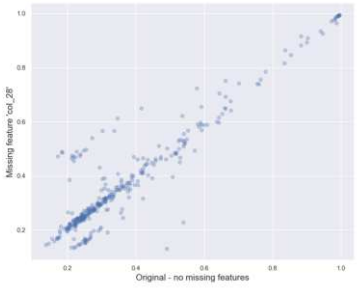
(xxvi) col_25



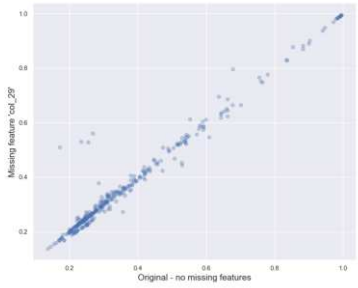
(xxvii) col_26



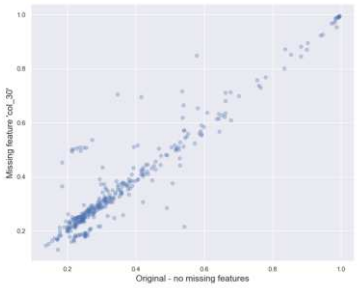
(xxviii) col_27



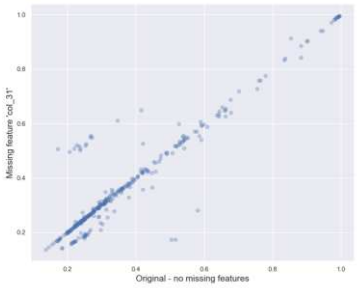
(xxix) col_28



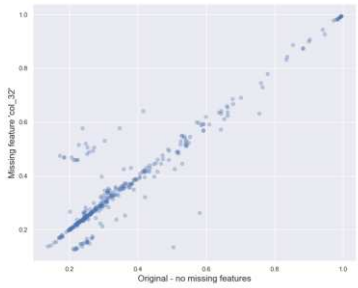
(xxx) col_29



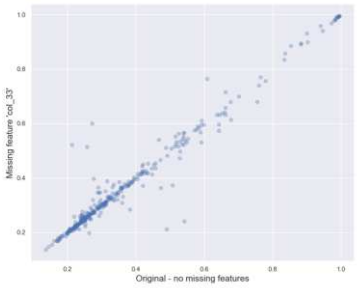
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

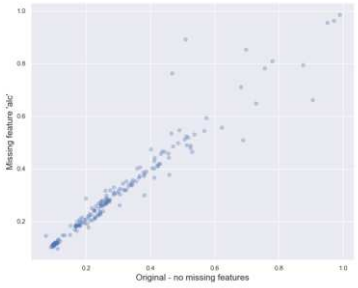


(xxxiv) col_33

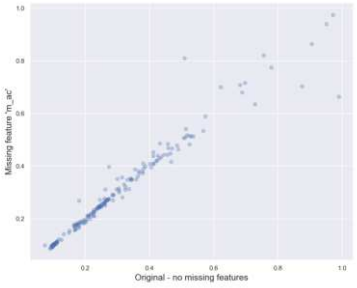
Source: Author's own elaboration

Figure 31 – Difficulty dispersion for the Wine dataset - Simple Imputation

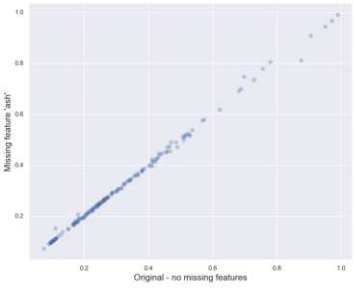
(i) alc



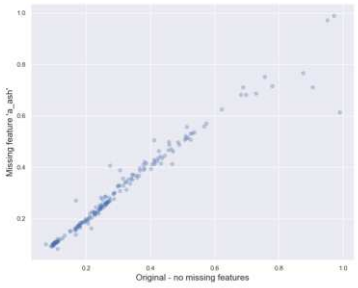
(ii) m_ac



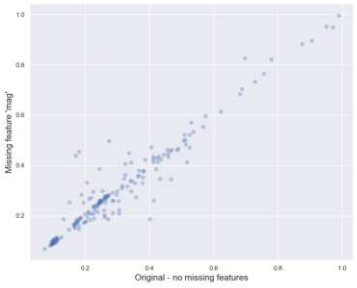
(iii) ash



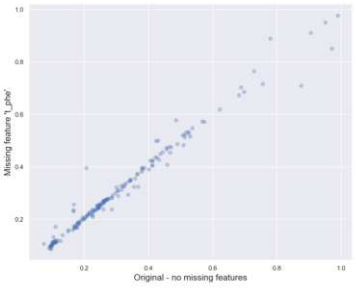
(iv) a_ash



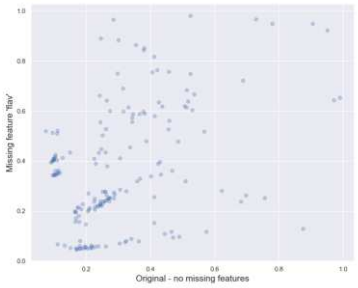
(v) mag



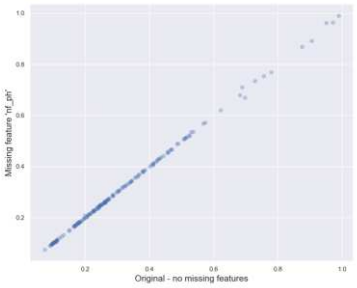
(vi) t_phe



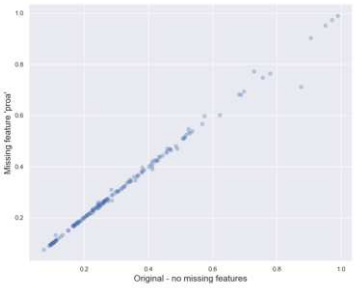
(vii) flav



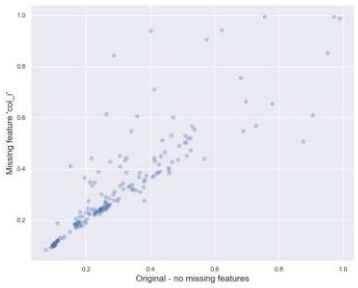
(viii) nf_ph



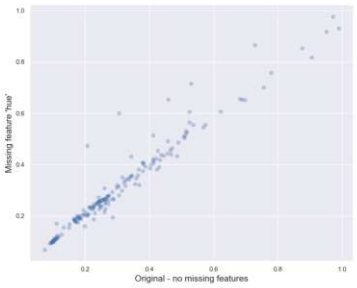
(ix) proa



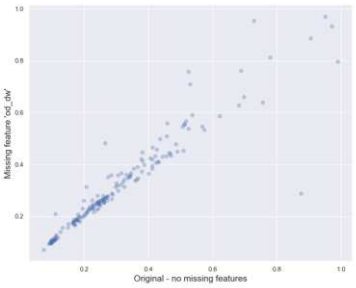
(x) col_i

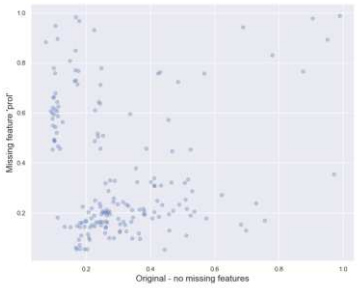


(xi) hue



(xii) od_dw



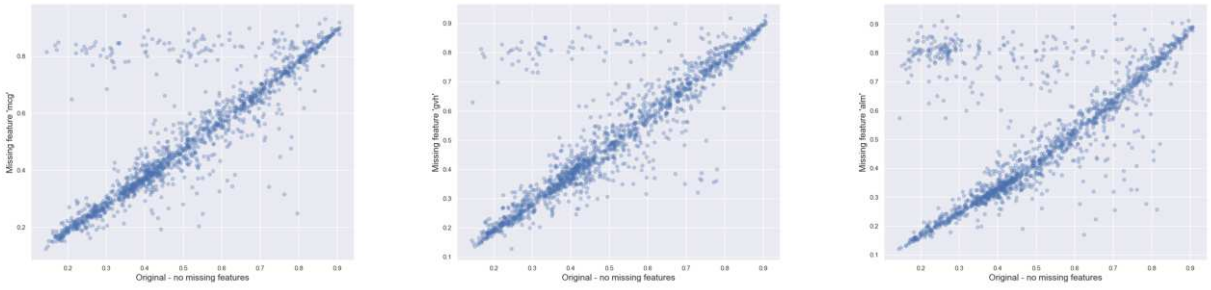


(xiii) prol

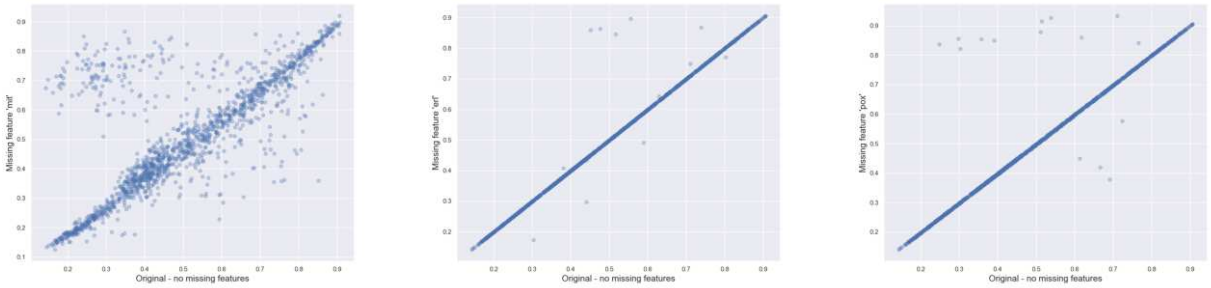
Source: Author's own elaboration

Figure 31 – Difficulty dispersion for the Yeast dataset - Simple Imputation

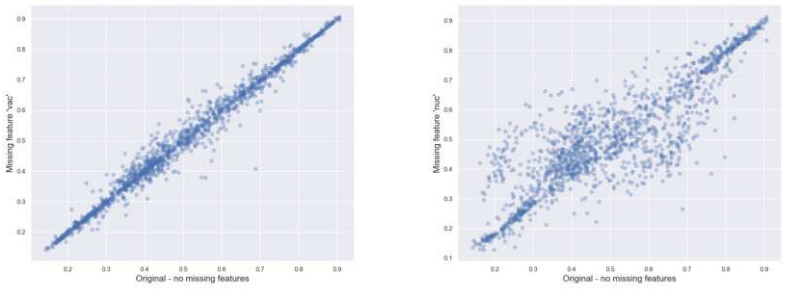
(i) mcg (ii) gvh (iii) alm



(iv) mit (v) erl (vi) pox



(vii) vac (viii) nuc



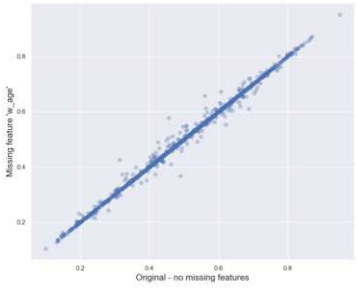
Source: Author's own elaboration

B.2 KNN

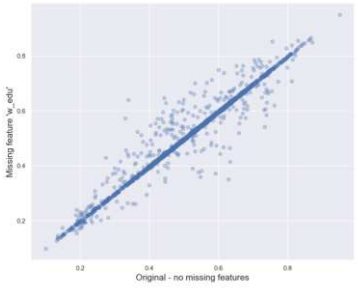
Figure 32 shows the results for the Contraceptive dataset, Figure 33 for the Pima dataset, Figure 34 for the Wheat dataset, Figure 35 for the Haberman dataset, Figure 36 for the Cleveland dataset, Figure 36 for the Ionosphere dataset, Figure 35 for the Wine dataset, and finally, Figure 35 shows the results for the Yeast dataset.

Figure 32 – Difficulty dispersion for the Contraceptive dataset - KNN Imputation

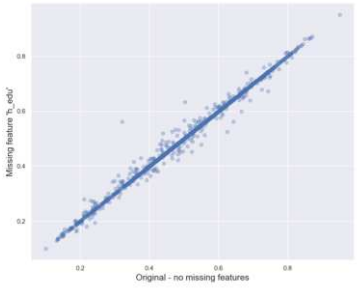
(i) w_age



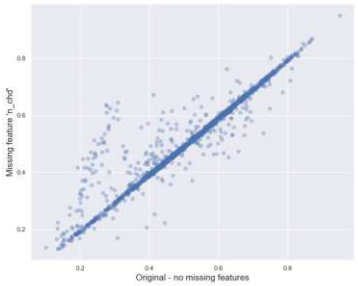
(ii) w_edu



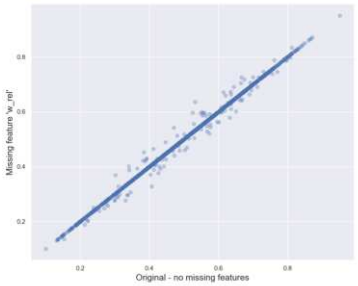
(iii) h_edu



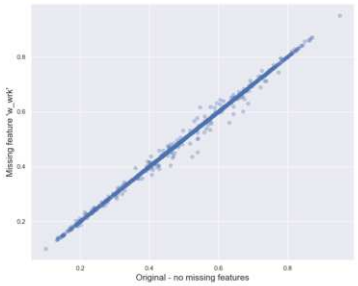
(iv) n_chd



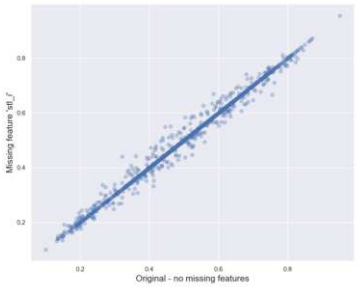
(v) w_rel



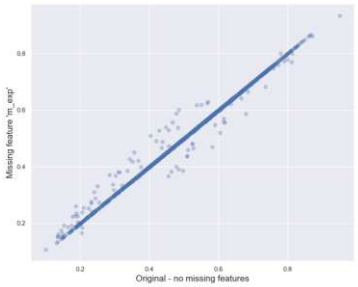
(vi) w_wrk



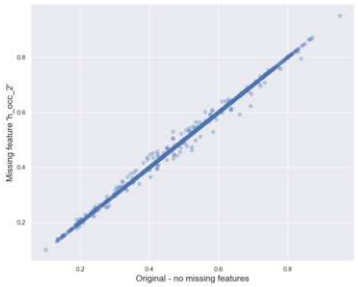
(vii) stl_i



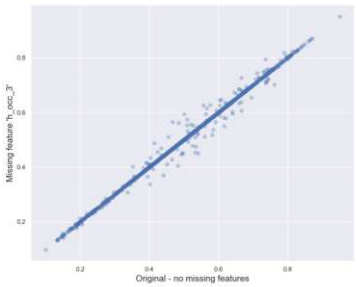
(viii) m_exp



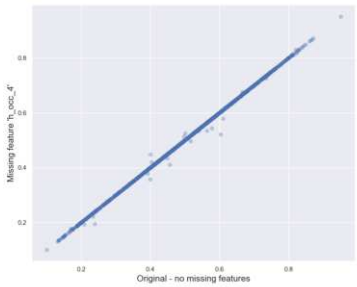
(ix) h_occ_2



(x) h_occ_3



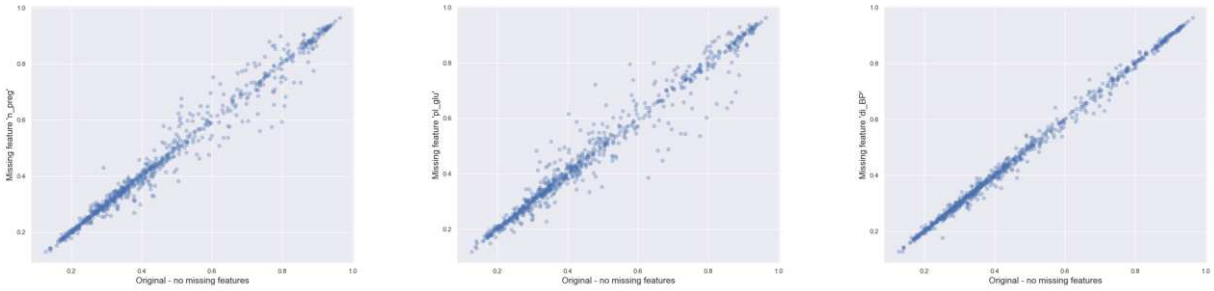
(xi) h_occ_4



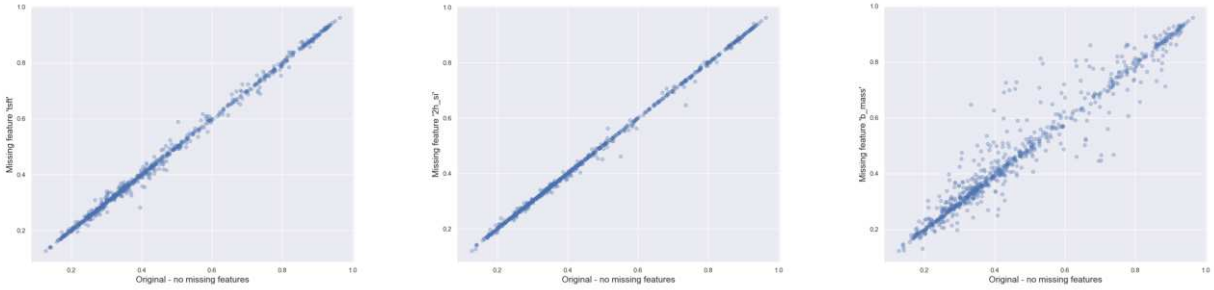
Source: Author's own elaboration

Figure 33 – Difficulty dispersion for the Pima dataset - KNN Imputation

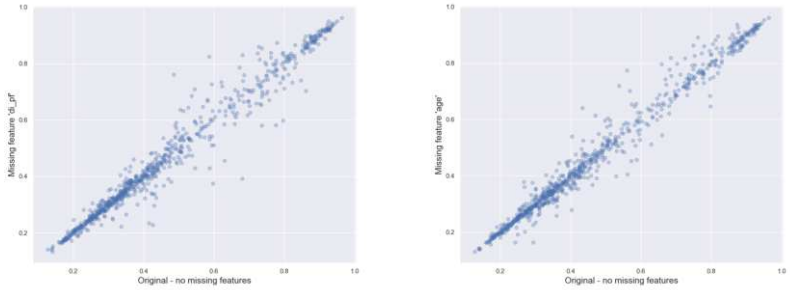
(i) n_preg (ii) pl_glu (iii) di_BP



(iv) tsft (v) 2h_si (vi) b_mass



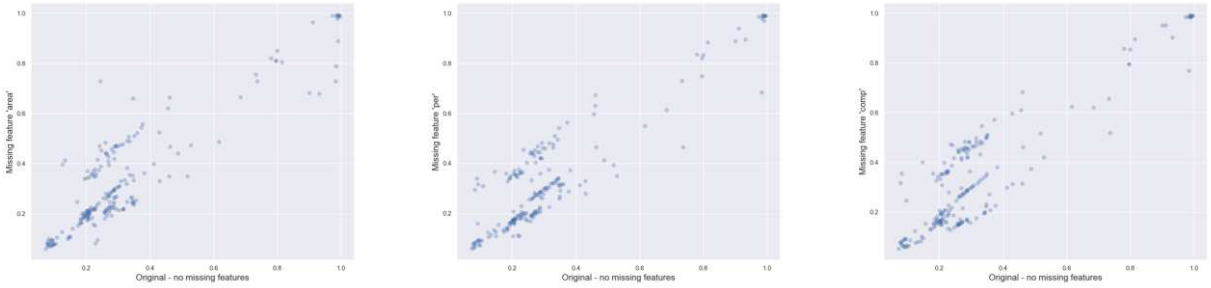
(vii) di_pf (viii) age



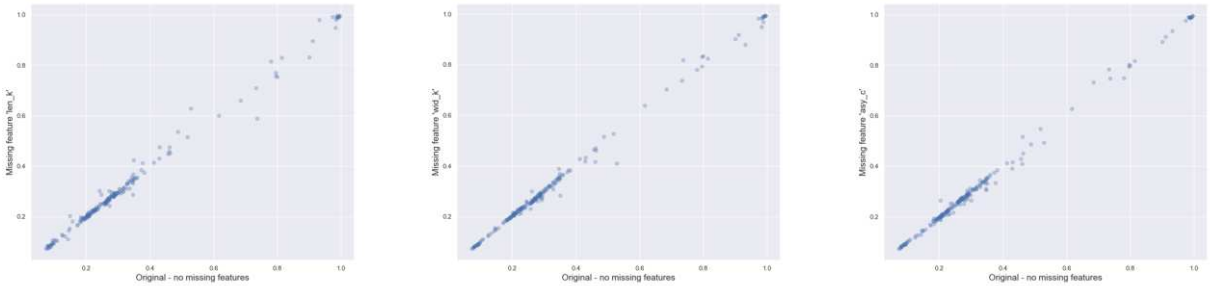
Source: Author's own elaboration

Figure 34 – Difficulty dispersion for the Wheat dataset - KNN Imputation

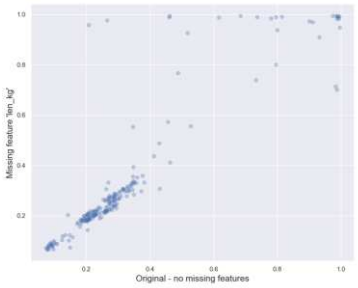
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



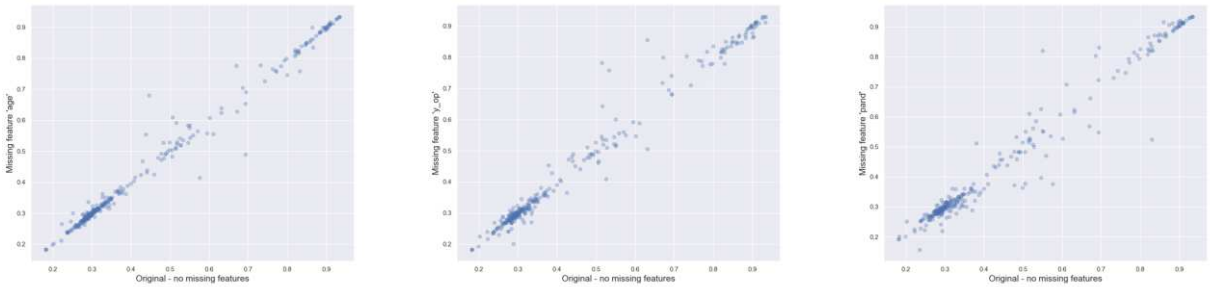
(vii) len_kg



Source: Author's own elaboration

Figure 35 – Difficulty dispersion for the Haberman dataset - KNN Imputation

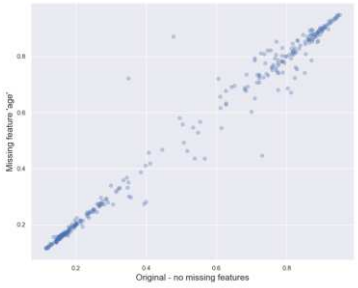
(i) age (ii) y_op (iii) pand



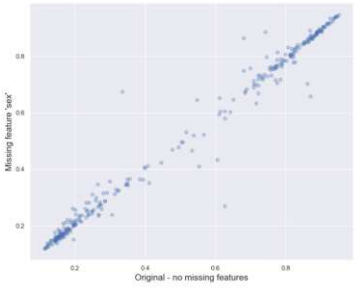
Source: Author's own elaboration

Figure 36 – Difficulty dispersion for the Cleveland dataset - KNN Imputation

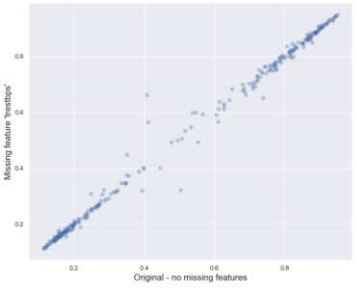
(i) age



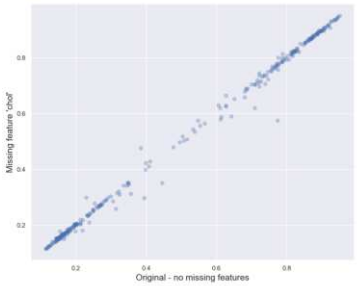
(ii) sex



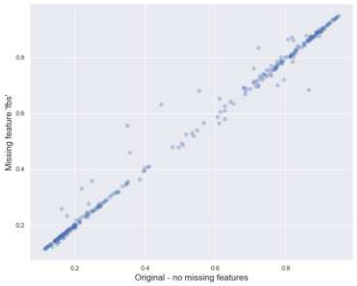
(iii) trestbps



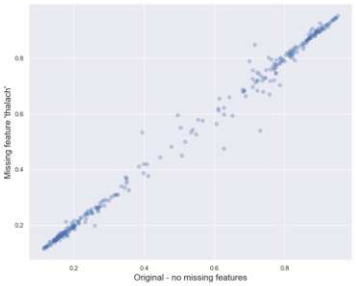
(iv) chol



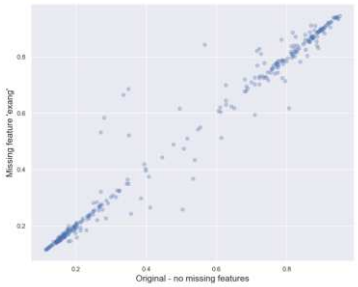
(v) fbs



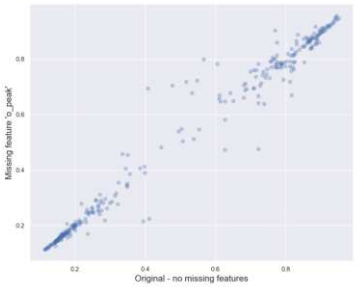
(vi) thalach



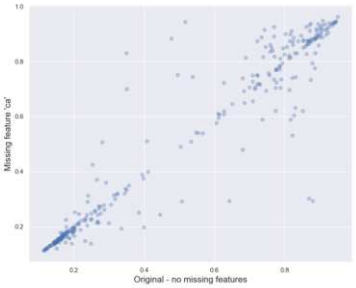
(vii) exang



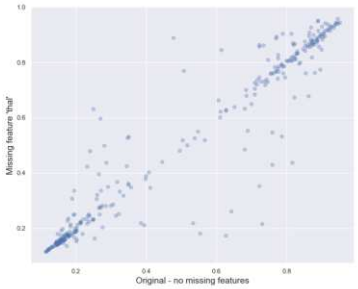
(viii) o_peak



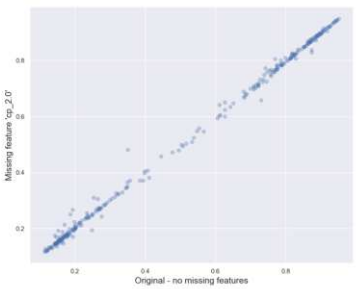
(ix) ca



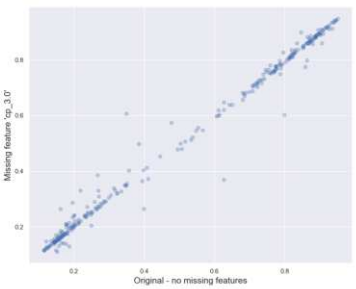
(x) thal

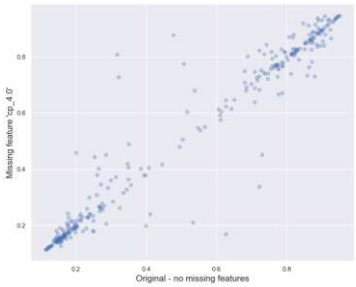


(xi) cp_2.0

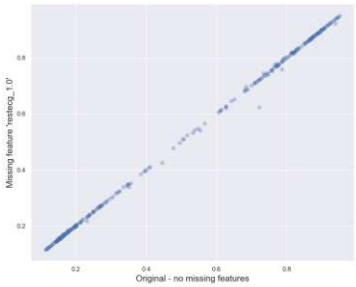


(xii) cp_3.0

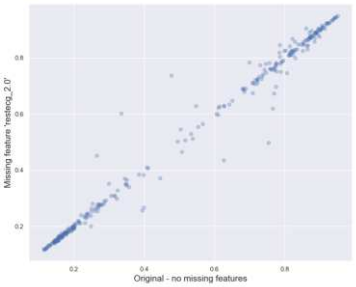




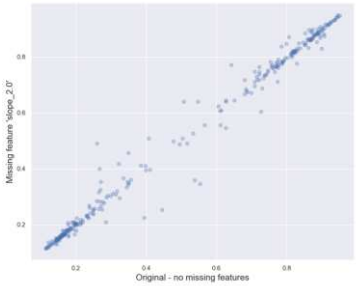
(xiii) cp_4.0



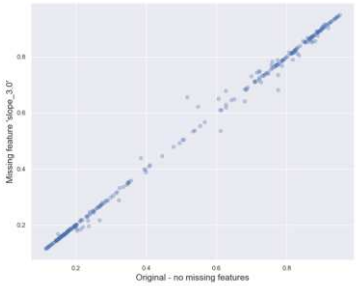
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

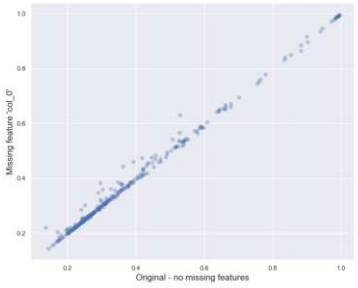


(xvii) slope_3.0

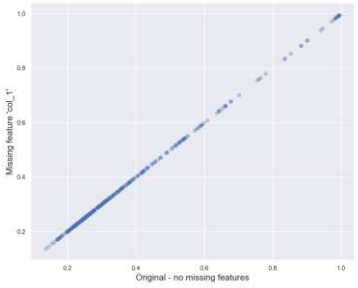
Source: Author's own elaboration

Figure 36 – Difficulty dispersion for the Ionosphere dataset - KNN Imputation

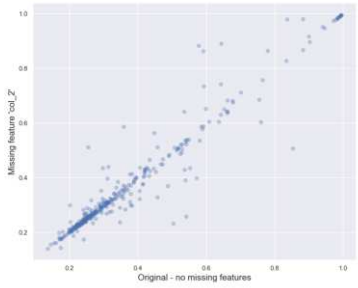
(i) col_0



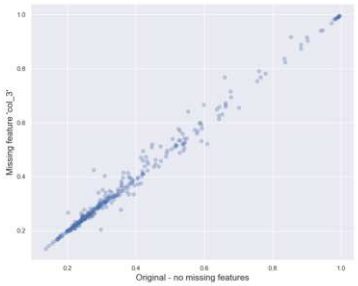
(ii) col_1



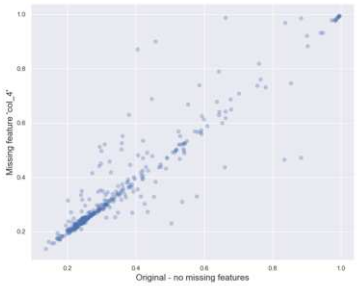
(iii) col_2



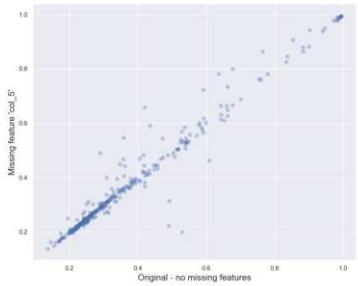
(iv) col_3



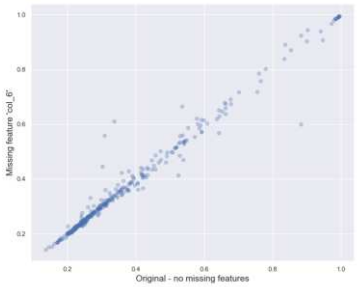
(v) col_4



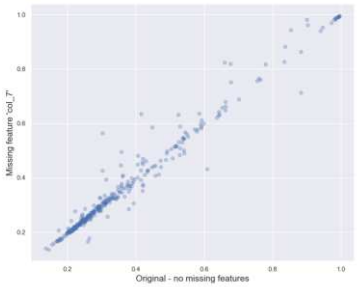
(vi) col_5



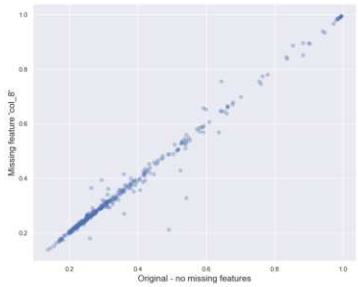
(vii) col_6



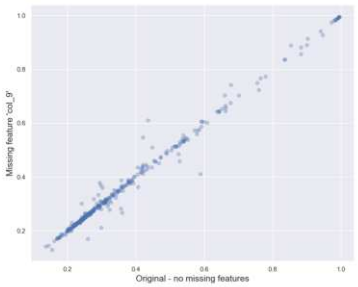
(viii) col_7



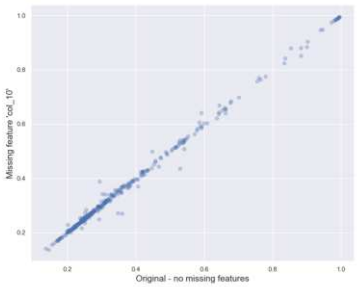
(ix) col_8



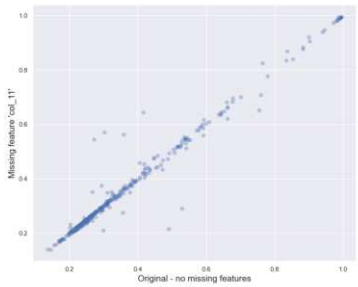
(x) col_9

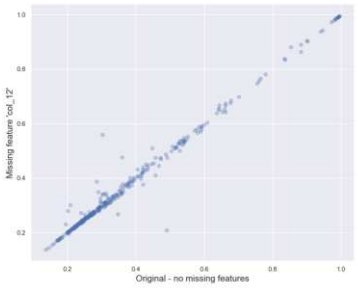


(xi) col_10

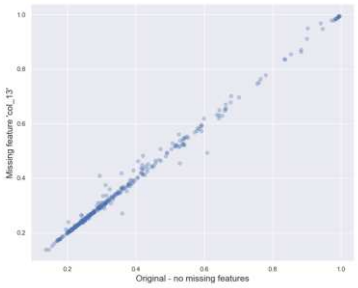


(xii) col_11

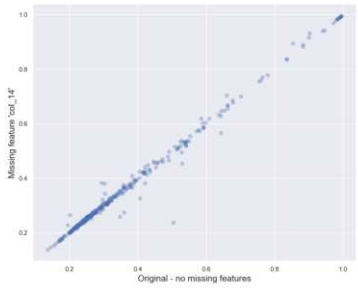




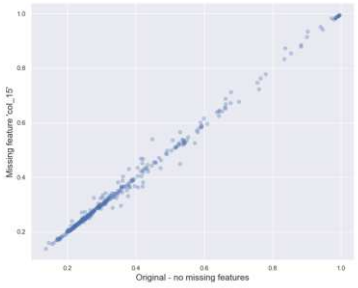
(xiii) col_12



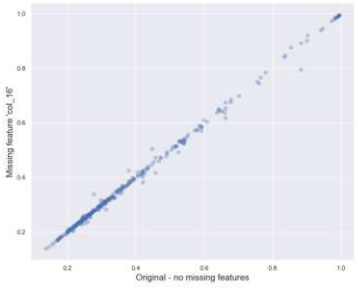
(xiv) col_13



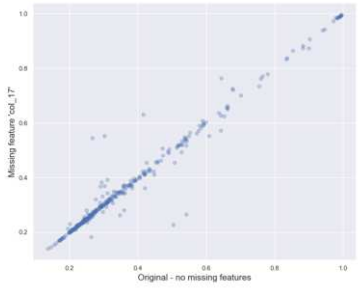
(xv) col_14



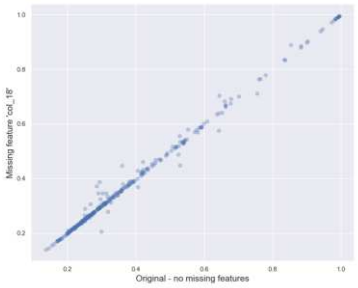
(xvi) col_15



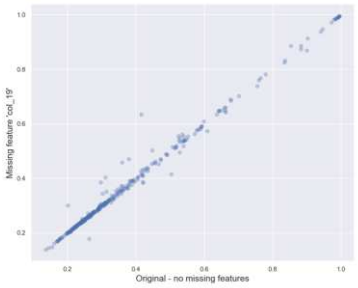
(xvii) col_16



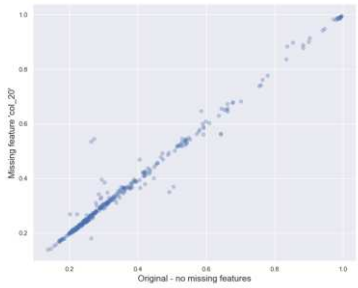
(xviii) col_17



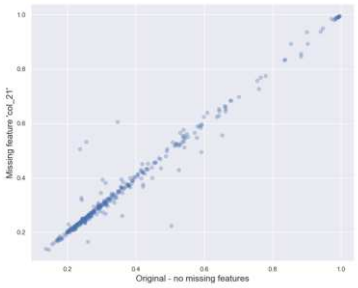
(xix) col_18



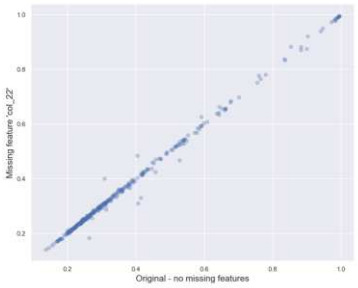
(xx) col_19



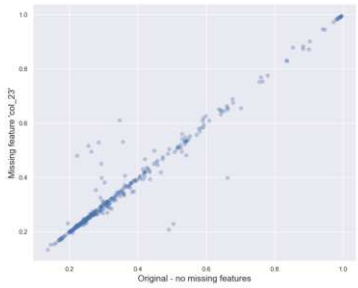
(xxi) col_20



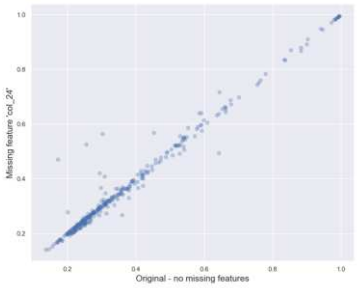
(xxii) col_21



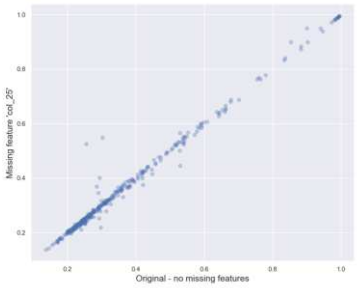
(xxiii) col_22



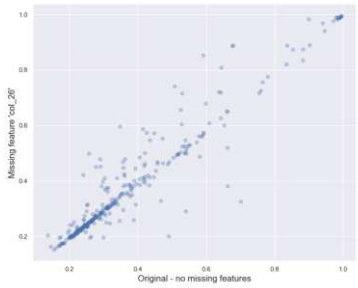
(xxiv) col_23



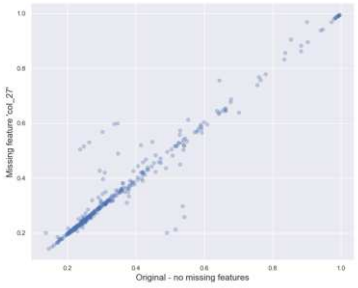
(xxv) col_24



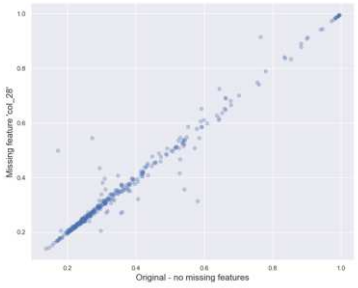
(xxvi) col_25



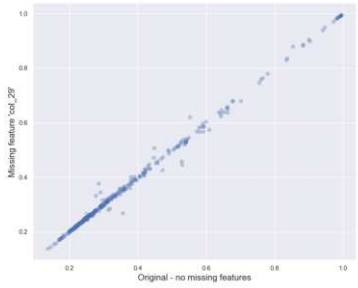
(xxvii) col_26



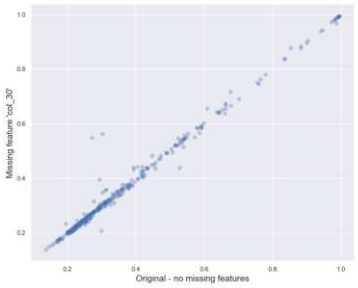
(xxviii) col_27



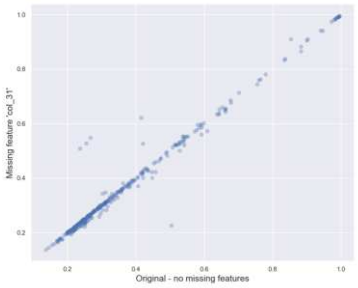
(xxix) col_28



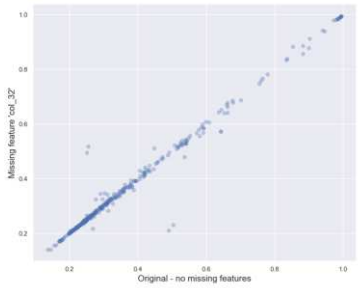
(xxx) col_29



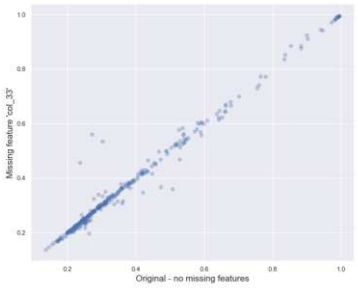
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

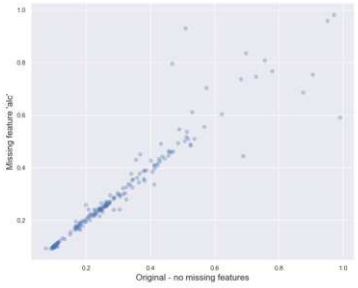


(xxxiv) col_33

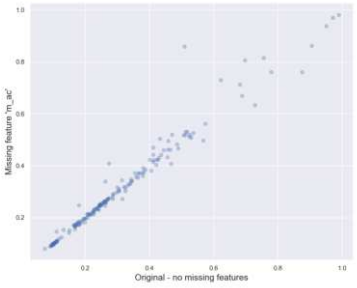
Source: Author's own elaboration

Figure 35 – Difficulty dispersion for the Wine dataset - KNN Imputation

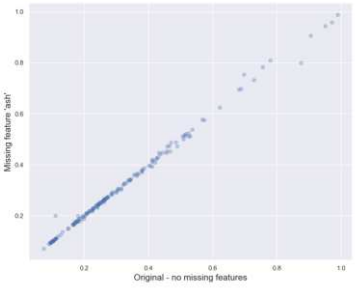
(i) alc



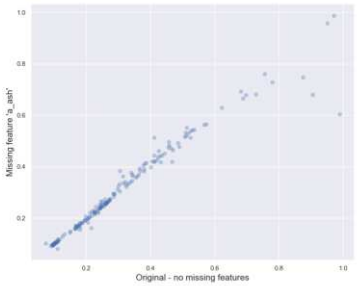
(ii) m_ac



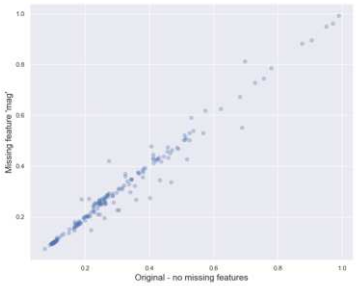
(iii) ash



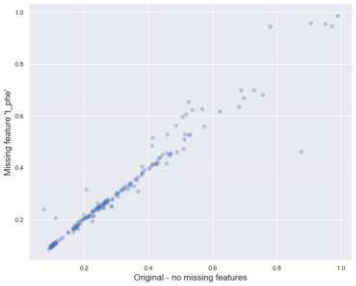
(iv) a_ash



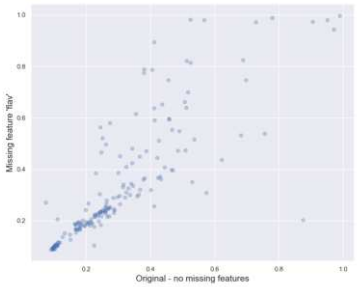
(v) mag



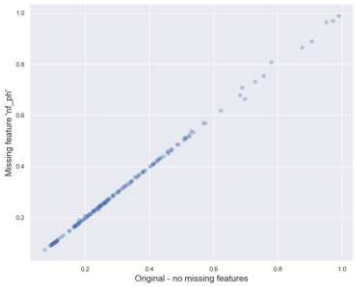
(vi) t_phe



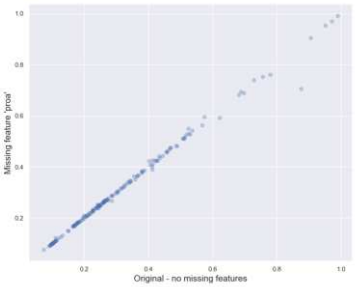
(vii) flav



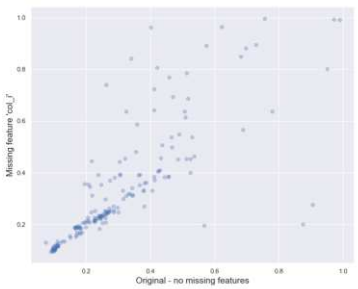
(viii) nf_ph



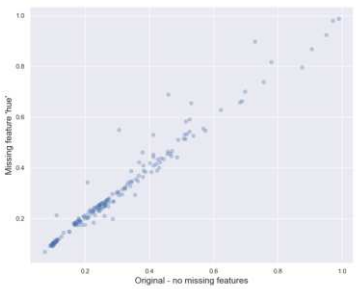
(ix) proa



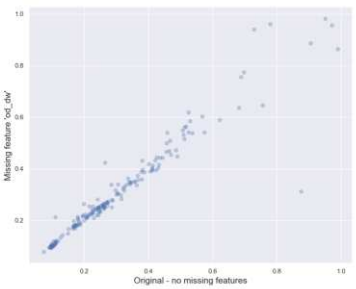
(x) col_i

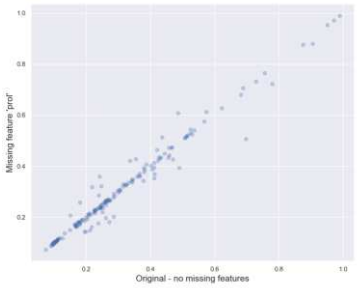


(xi) hue



(xii) od_dw



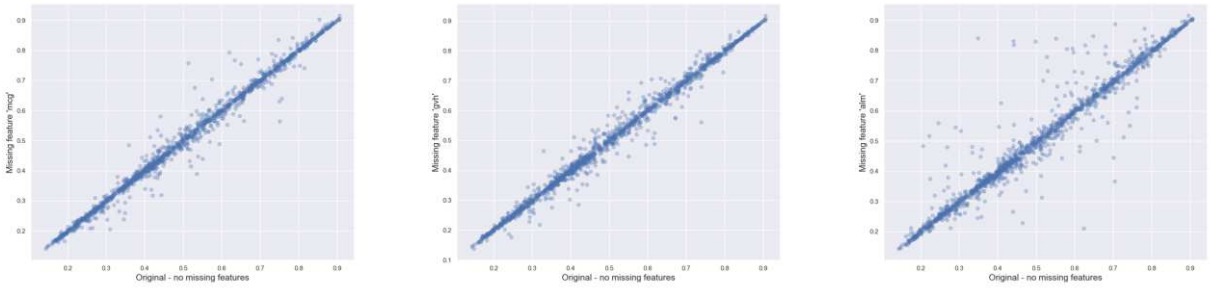


(xiii) prol

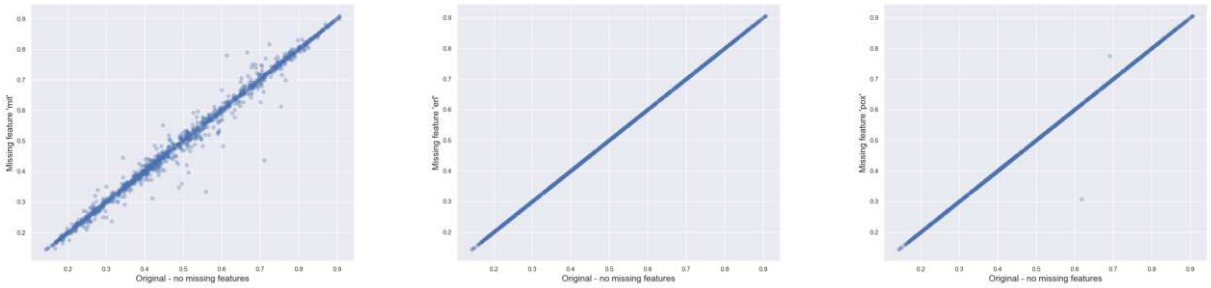
Source: Author's own elaboration

Figure 35 – Difficulty dispersion for the Yeast dataset - KNN Imputation

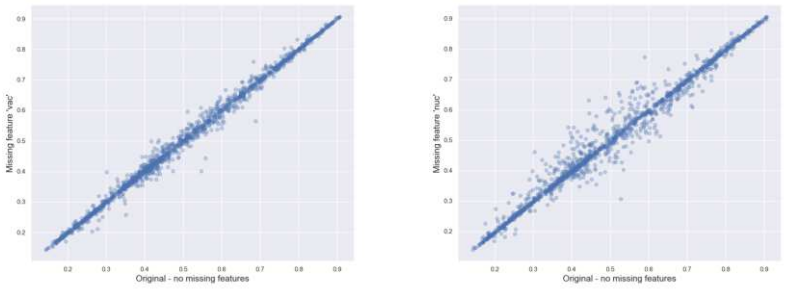
(i) mcg (ii) gvh (iii) alm



(iv) mit (v) erl (vi) pox



(vii) vac (viii) nuc



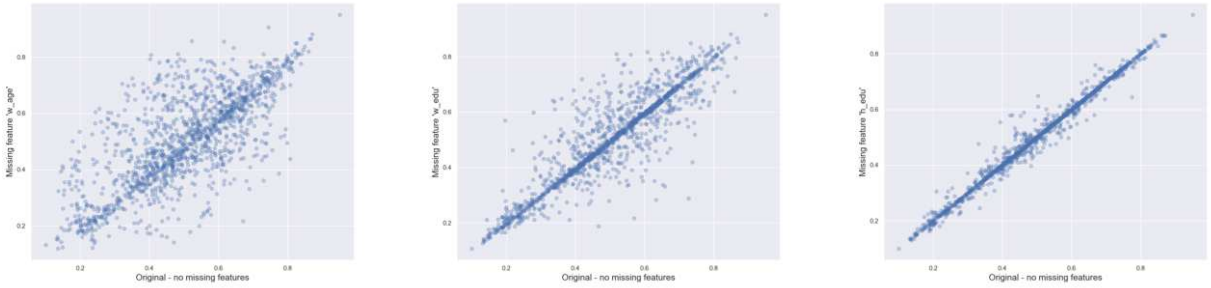
Source: Author's own elaboration

B.3 ITERATIVE

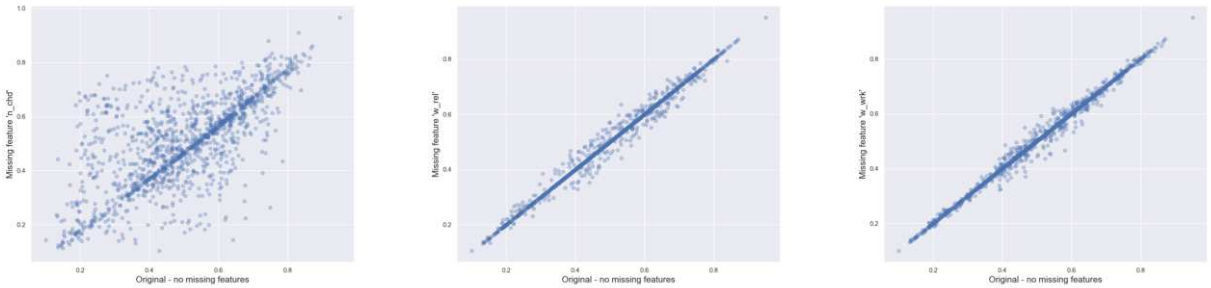
Figure 36 shows the results for the Contraceptive dataset, Figure 37 for the Pima dataset, Figure 38 for the Wheat dataset, Figure 39 for the Haberman dataset, Figure 40 for the Cleveland dataset, Figure 40 for the Ionosphere dataset, Figure 39 for the Wine dataset, and finally, Figure 39 shows the results for the Yeast dataset.

Figure 36 – Difficulty dispersion for the Contraceptive dataset - Iterative Imputation

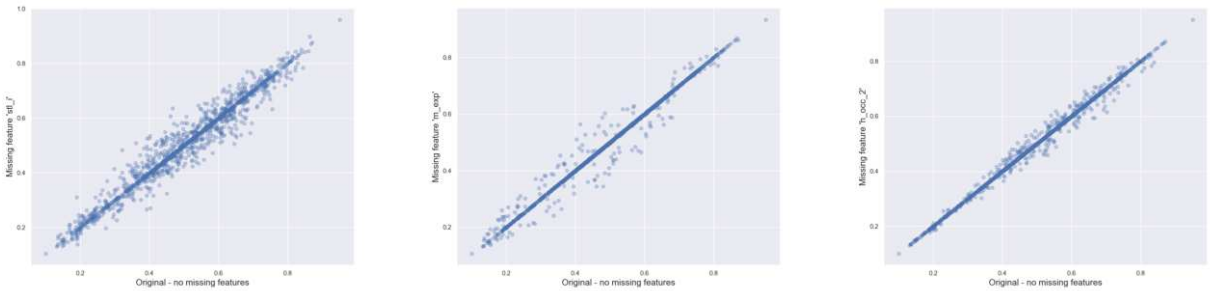
(i) w_age (ii) w_edu (iii) h_edu



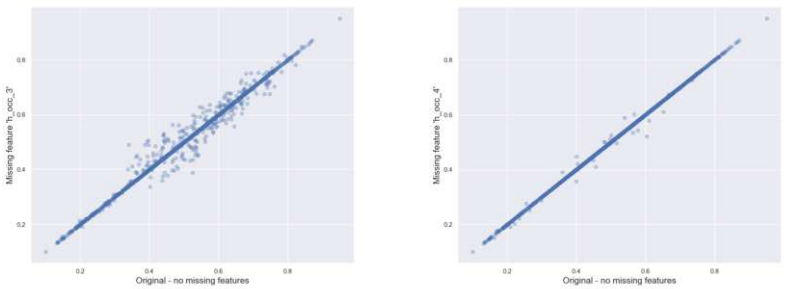
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



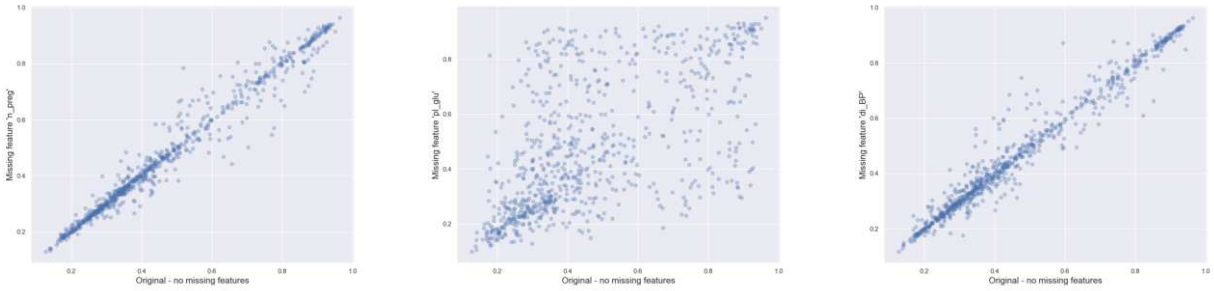
(x) h_occ_3 (xi) h_occ_4



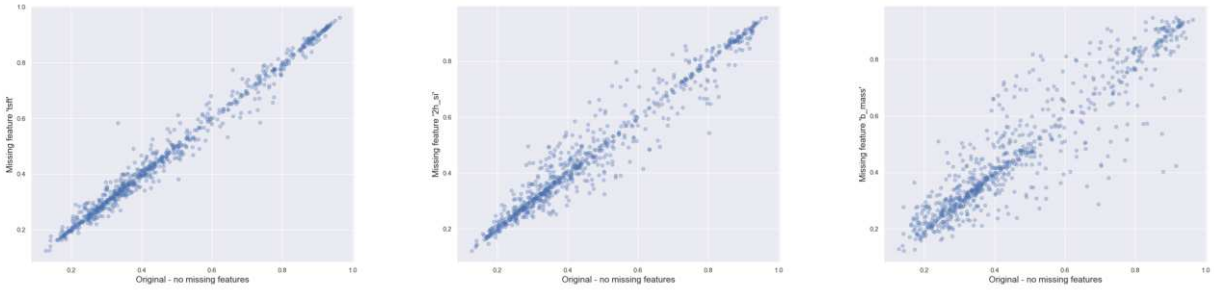
Source: Author's own elaboration

Figure 37 – Difficulty dispersion for the Pima dataset - Iterative Imputation

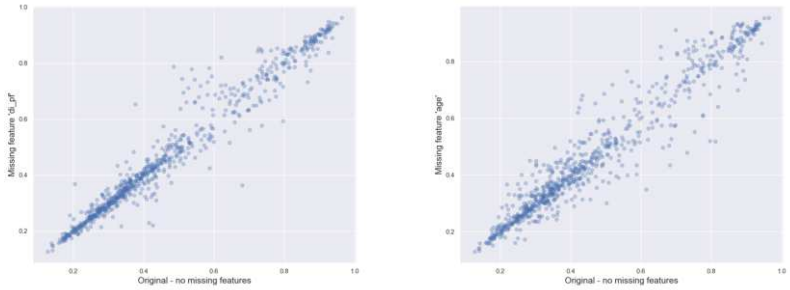
(i) n_preg (ii) pl_glu (iii) di_BP



(iv) tsft (v) 2h_si (vi) b_mass



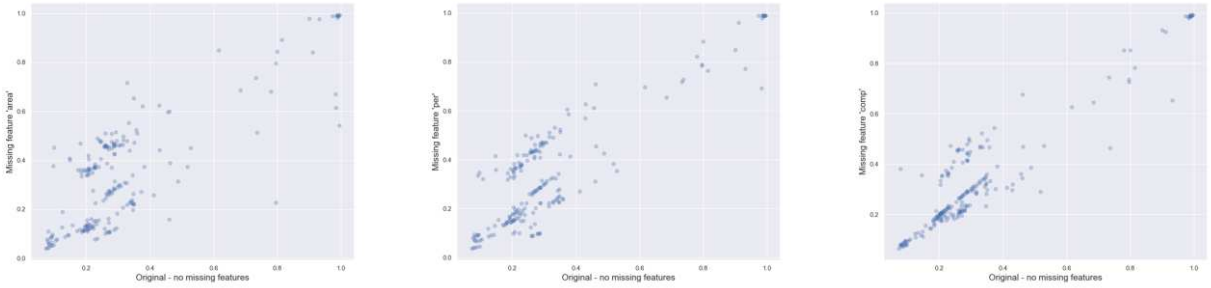
(vii) di_pf (viii) age



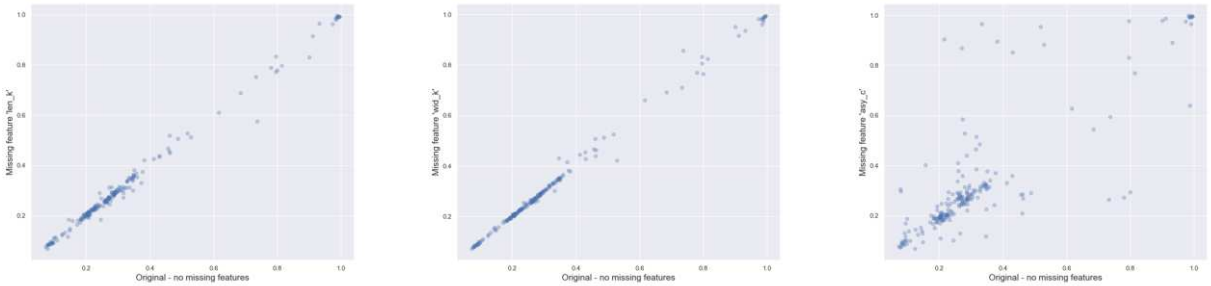
Source: Author's own elaboration

Figure 38 – Difficulty dispersion for the Wheat dataset - Iterative Imputation

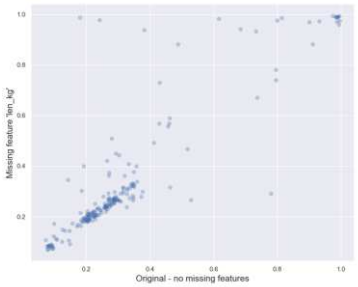
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



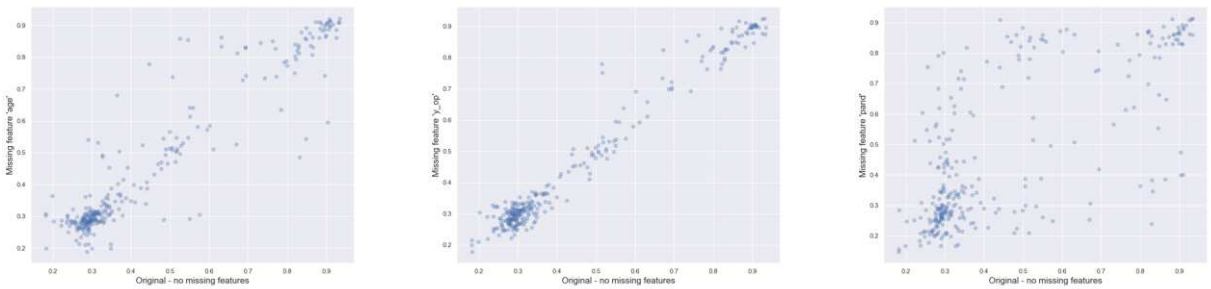
(vii) len_kg



Source: Author's own elaboration

Figure 39 – Difficulty dispersion for the Haberman dataset - Iterative Imputation

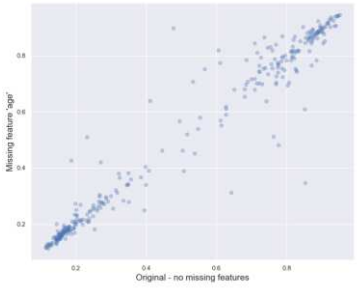
(i) age (ii) y_op (iii) pand



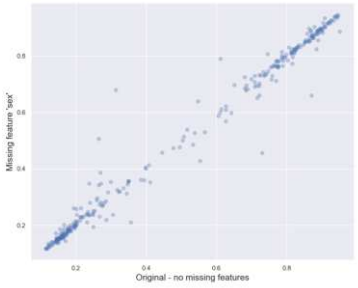
Source: Author's own elaboration

Figure 40 – Difficulty dispersion for the Cleveland dataset - Iterative Imputation

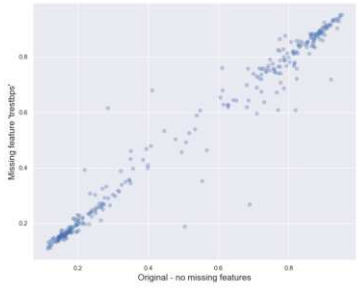
(i) age



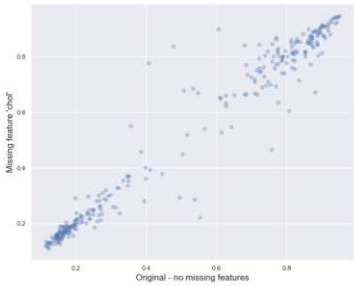
(ii) sex



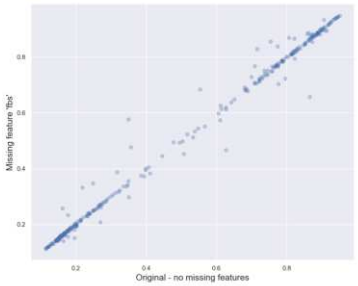
(iii) trestbps



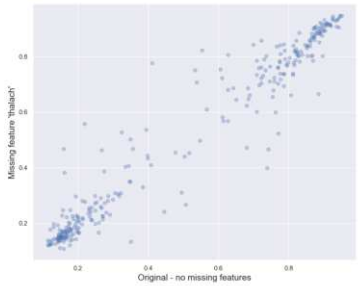
(iv) chol



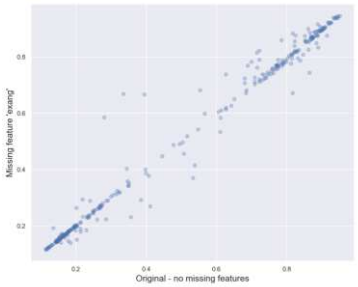
(v) fbs



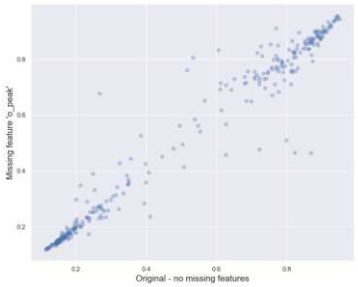
(vi) thalach



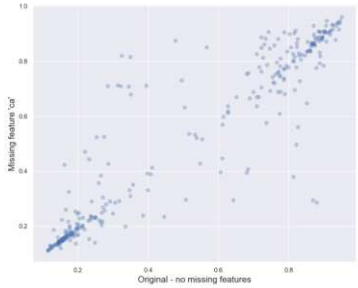
(vii) exang



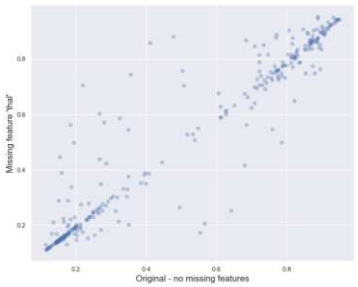
(viii) o_peak



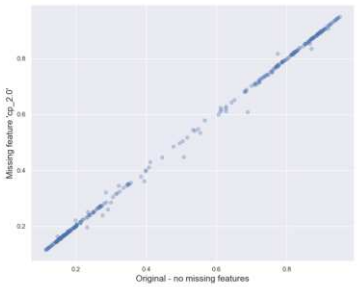
(ix) ca



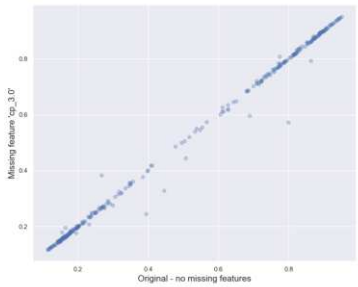
(x) thal

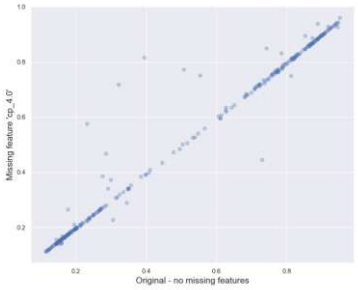


(xi) cp_2.0

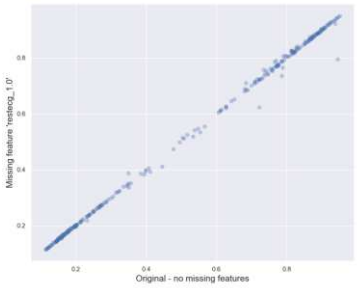


(xii) cp_3.0

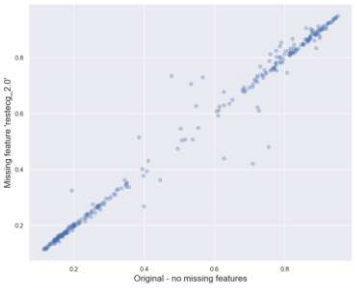




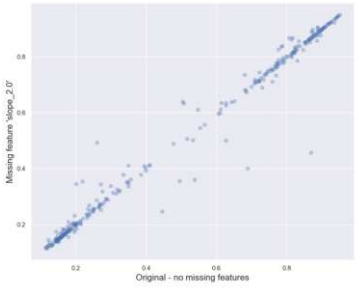
(xiii) cp_4.0



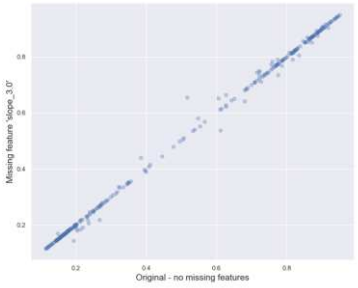
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

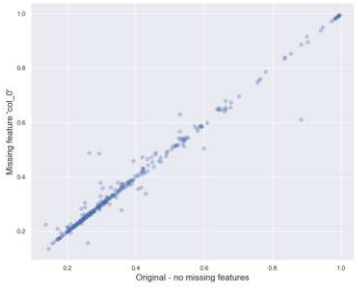


(xvii) slope_3.0

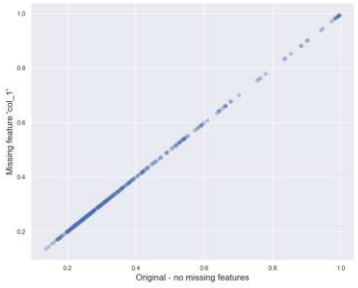
Source: Author's own elaboration

Figure 40 – Difficulty dispersion for the Ionosphere dataset - Iterative Imputation

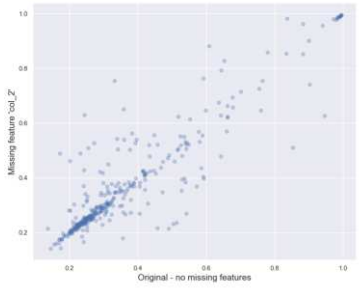
(i) col_0



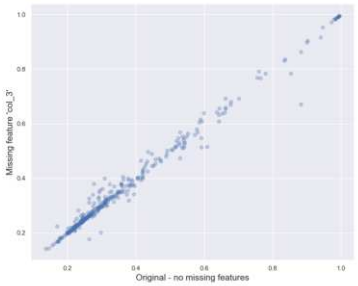
(ii) col_1



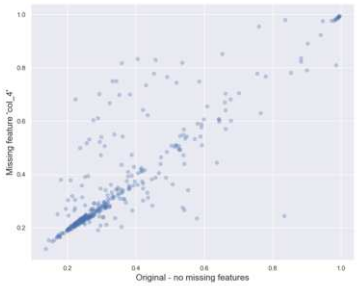
(iii) col_2



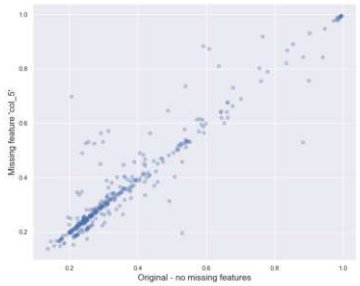
(iv) col_3



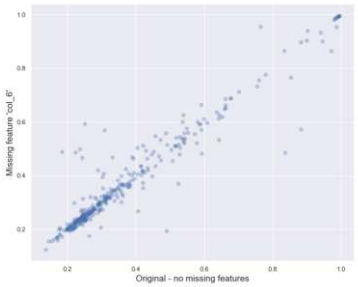
(v) col_4



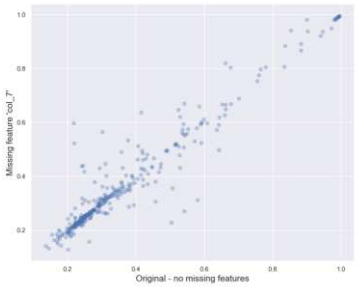
(vi) col_5



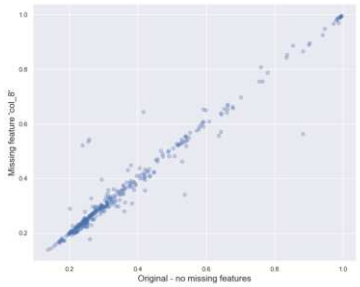
(vii) col_6



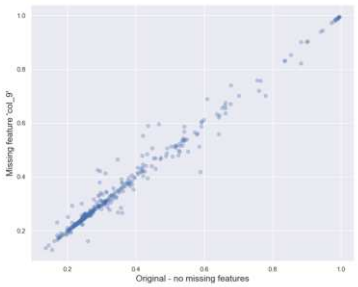
(viii) col_7



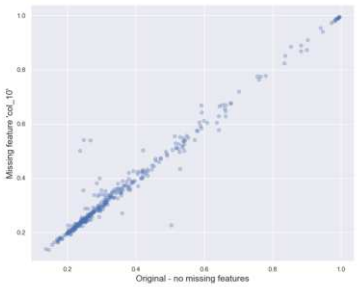
(ix) col_8



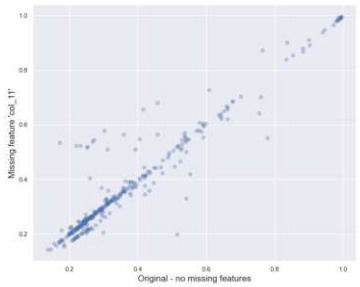
(x) col_9

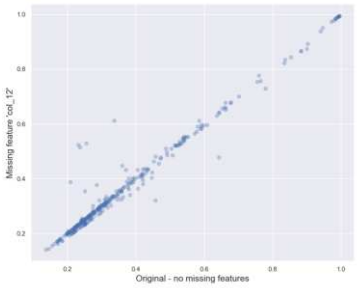


(xi) col_10

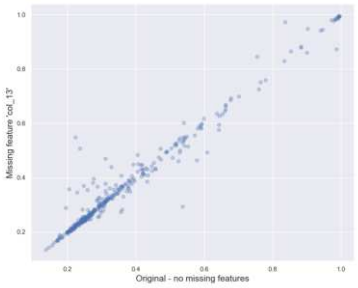


(xii) col_11

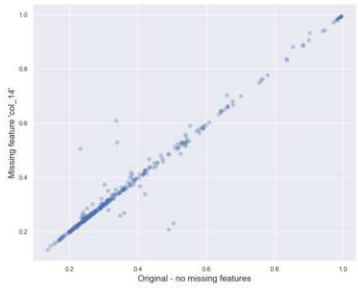




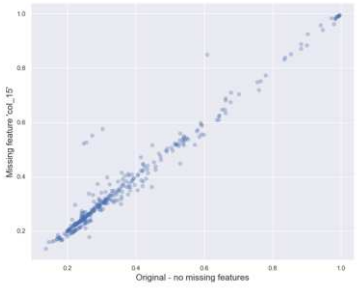
(xiii) col_12



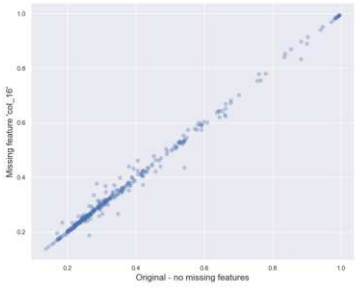
(xiv) col_13



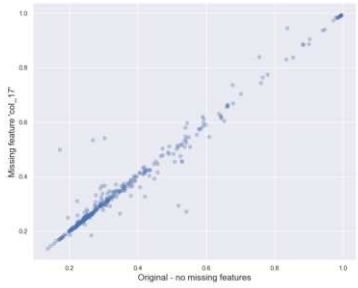
(xv) col_14



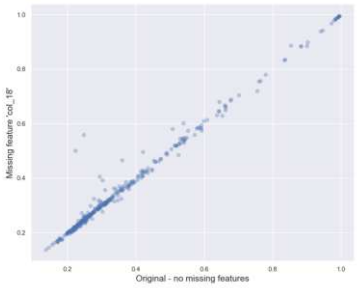
(xvi) col_15



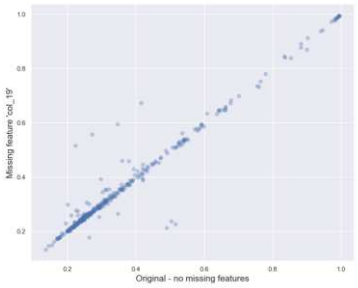
(xvii) col_16



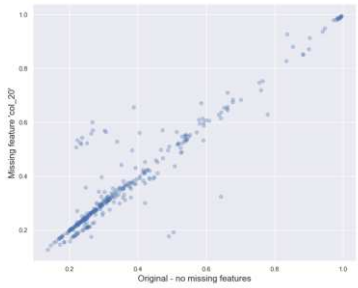
(xviii) col_17



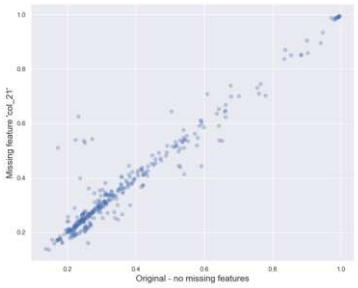
(xix) col_18



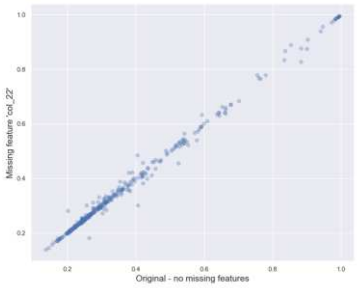
(xx) col_19



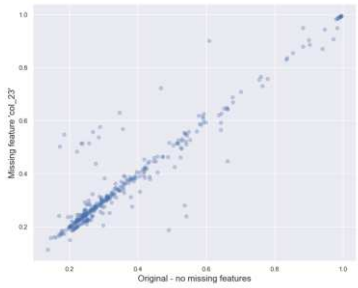
(xxi) col_20



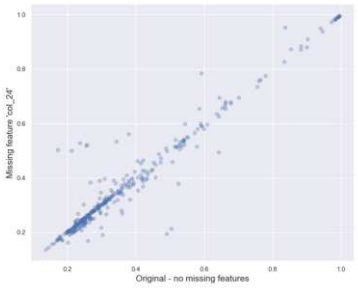
(xxii) col_21



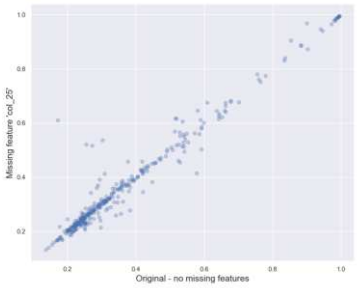
(xxiii) col_22



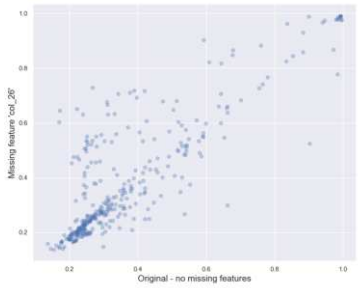
(xxiv) col_23



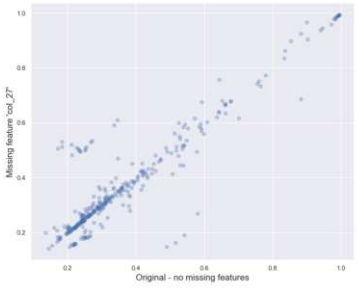
(xxv) col_24



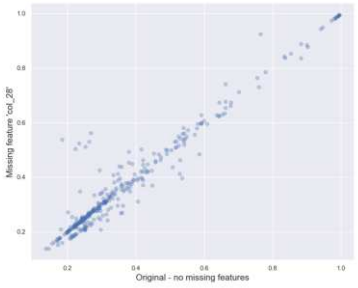
(xxvi) col_25



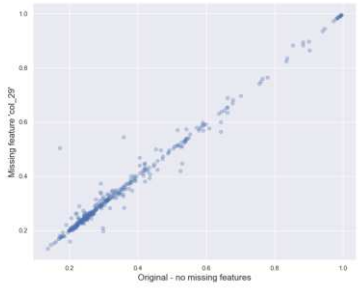
(xxvii) col_26



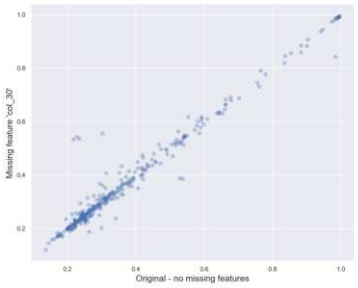
(xxviii) col_27



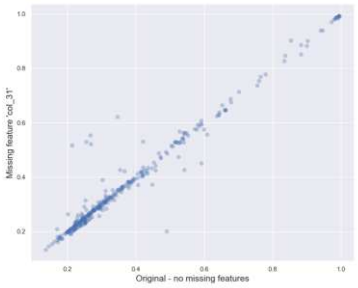
(xxix) col_28



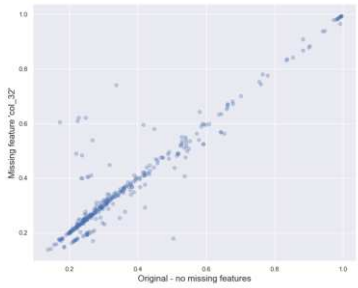
(xxx) col_29



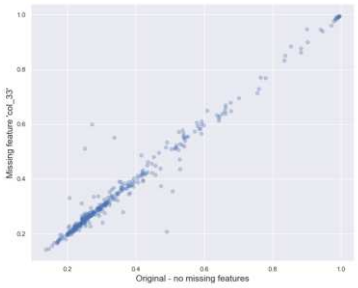
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

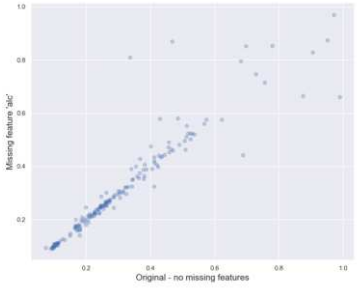


(xxxiv) col_33

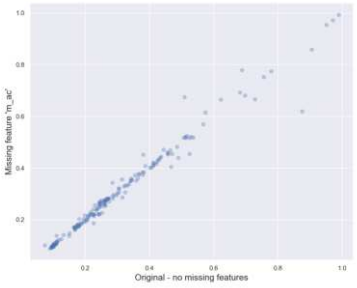
Source: Author's own elaboration

Figure 39 – Difficulty dispersion for the Wine dataset - Iterative Imputation

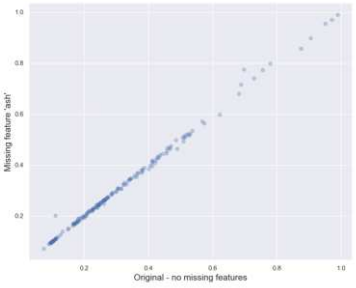
(i) alc



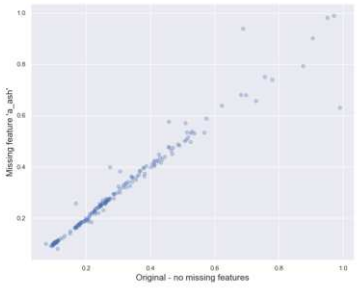
(ii) m_ac



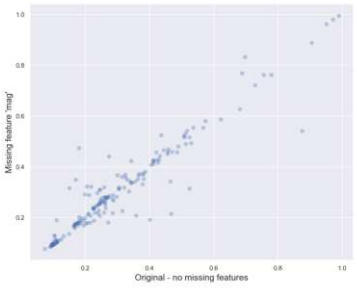
(iii) ash



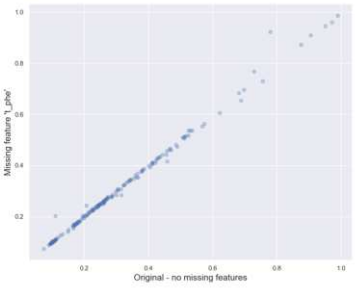
(iv) a_ash



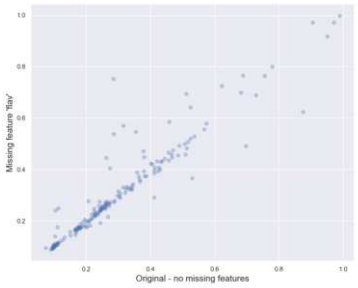
(v) mag



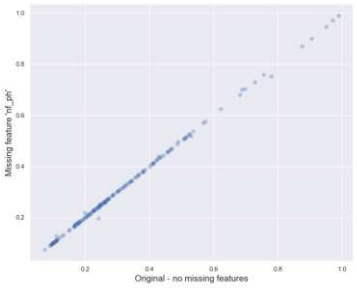
(vi) t_phe



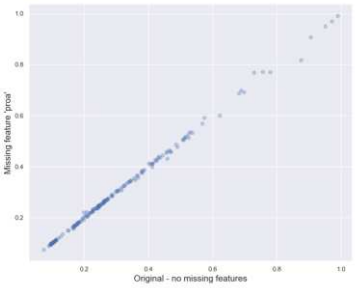
(vii) flav



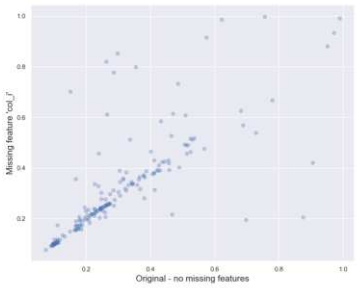
(viii) nf_ph



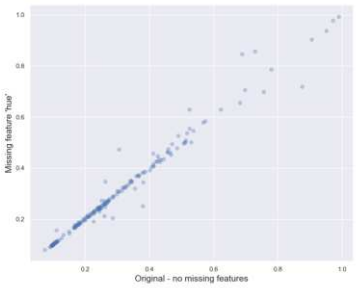
(ix) proa



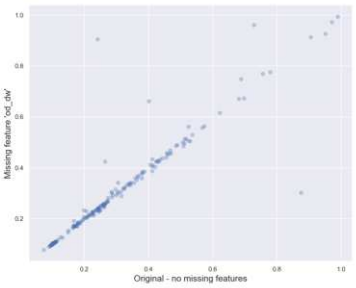
(x) col_i

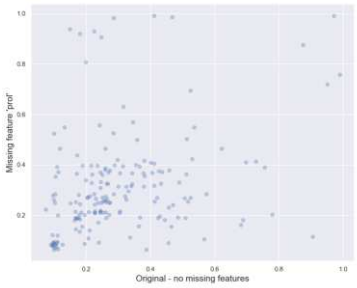


(xi) hue



(xii) od_dw



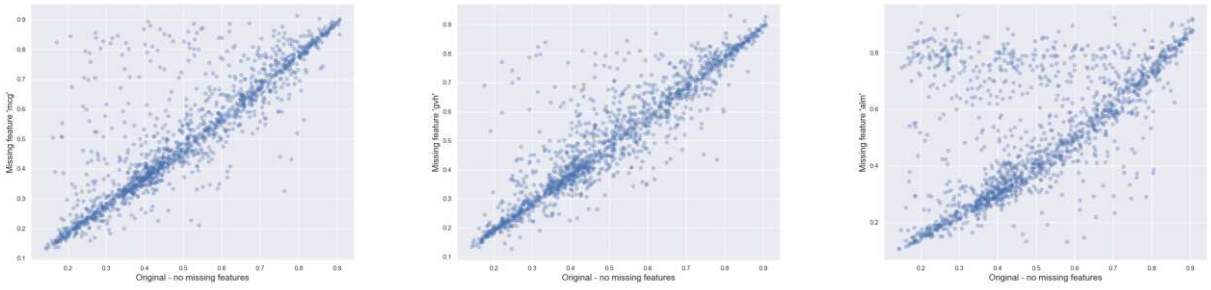


(xiii) prol

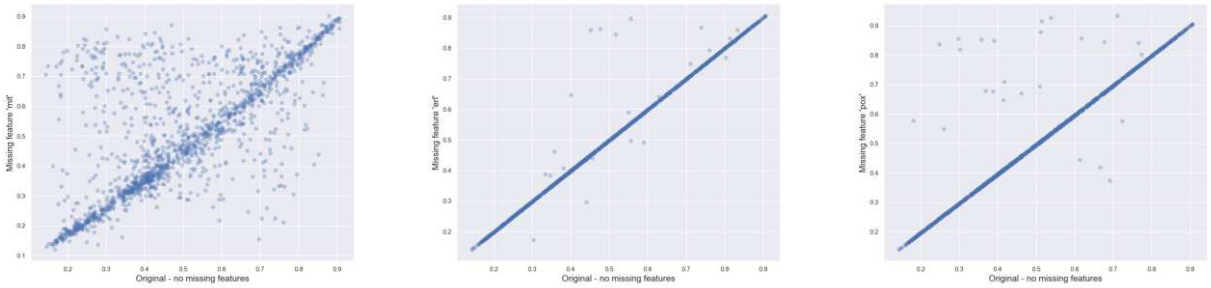
Source: Author's own elaboration

Figure 39 – Difficulty dispersion for the Yeast dataset - Iterative Imputation

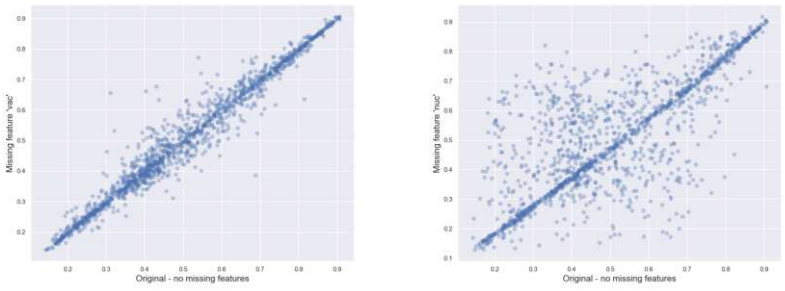
(i) mcg (ii) gvh (iii) alm



(iv) mit (v) erl (vi) pox



(vii) vac (viii) nuc



Source: Author's own elaboration

APPENDIX C – DISCRIMINATION DISPERSION PER SCENARIO

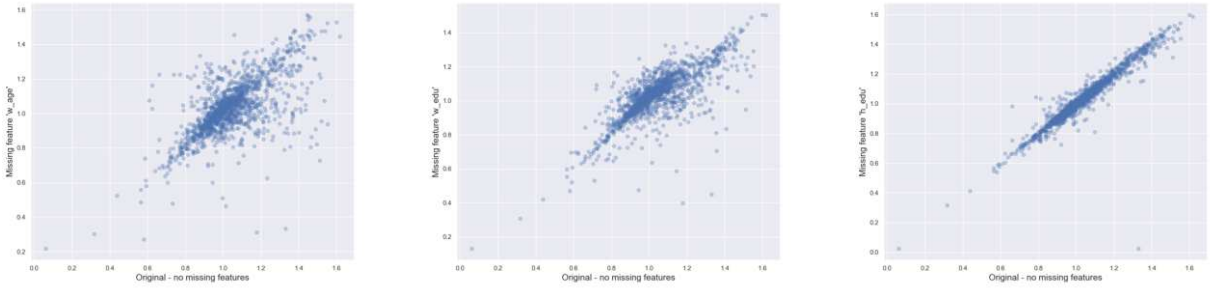
In this chapter we show the one-to-one difficulty values per scenario, for each dataset. We split the graphs for each imputation type: simple, KNN, and iterative.

C.1 SIMPLE

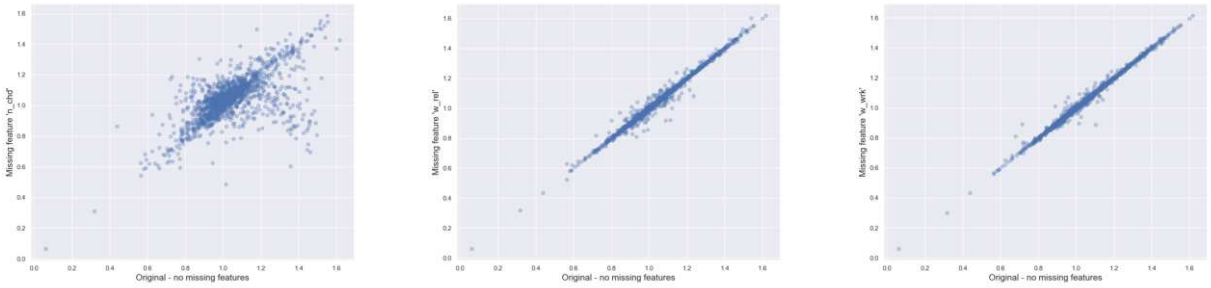
Figure 40 shows the results for the Contraceptive dataset, Figure 41 for the Pima dataset, Figure 42 for the Wheat dataset, Figure 43 for the Haberman dataset, Figure 44 for the Cleveland dataset, Figure 44 for the Ionosphere dataset, Figure 43 for the Wine dataset, and finally, Figure 43 shows the results for the Yeast dataset.

Figure 40 – Discrimination dispersion for the Contraceptive dataset - Simple Imputation

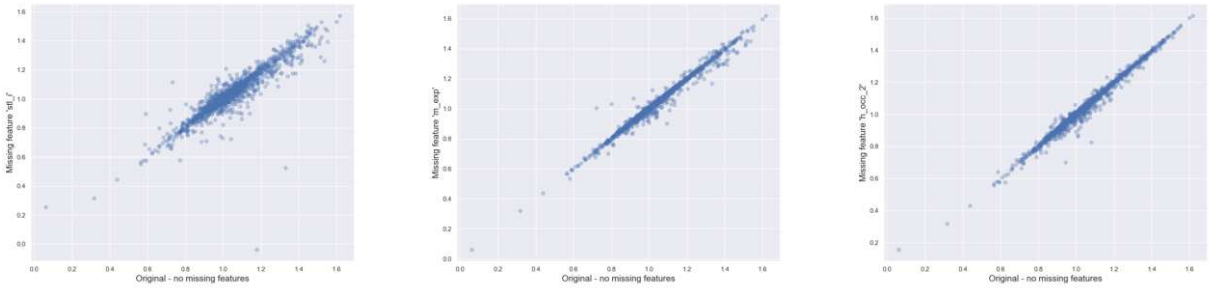
(i) w_age (ii) w_edu (iii) h_edu



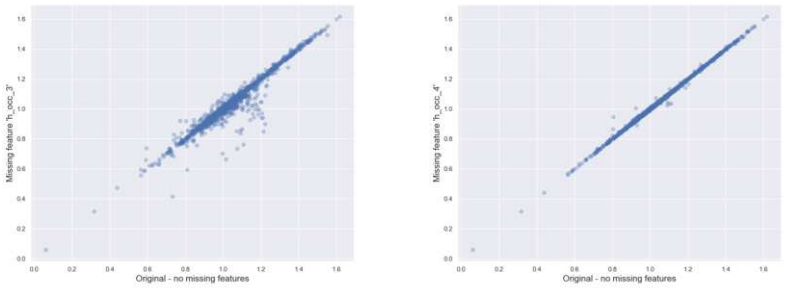
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



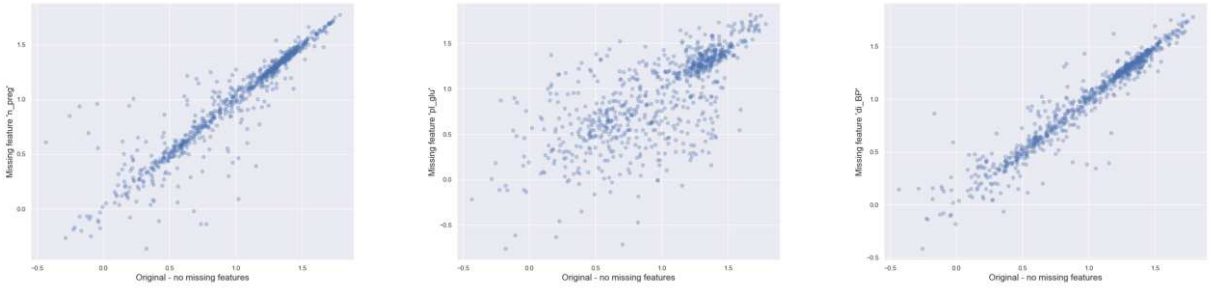
(x) h_occ_3 (xi) h_occ_4



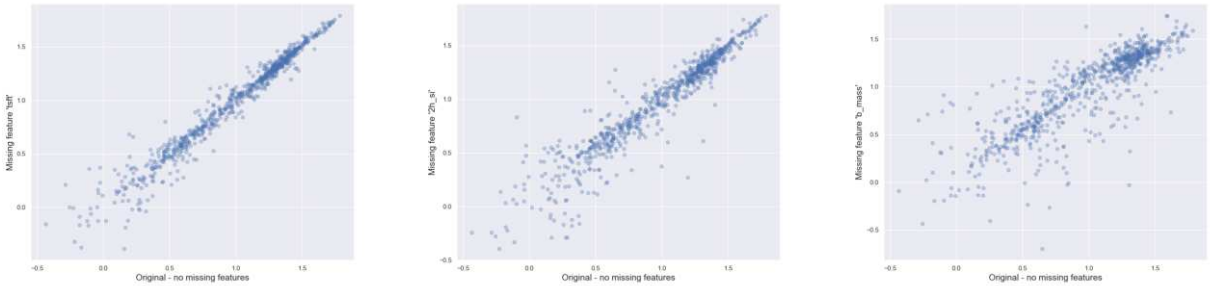
Source: Author's own elaboration

Figure 41 – Discrimination dispersion for the Pima dataset - Simple Imputation

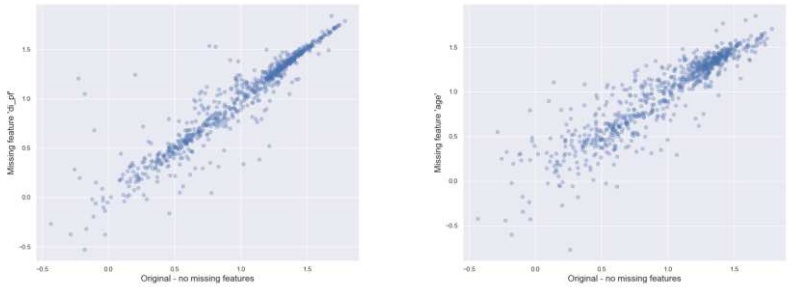
(i) n_preg (ii) pl_glu (iii) di_BP



(iv) tsft (v) 2h_si (vi) b_mass



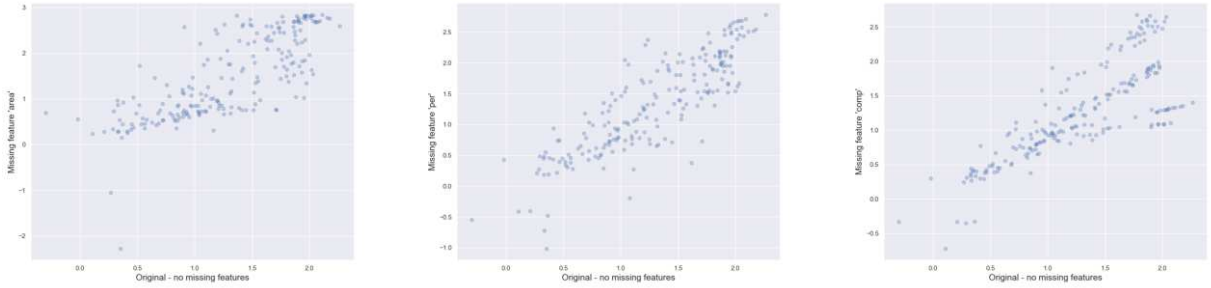
(vii) di_pf (viii) age



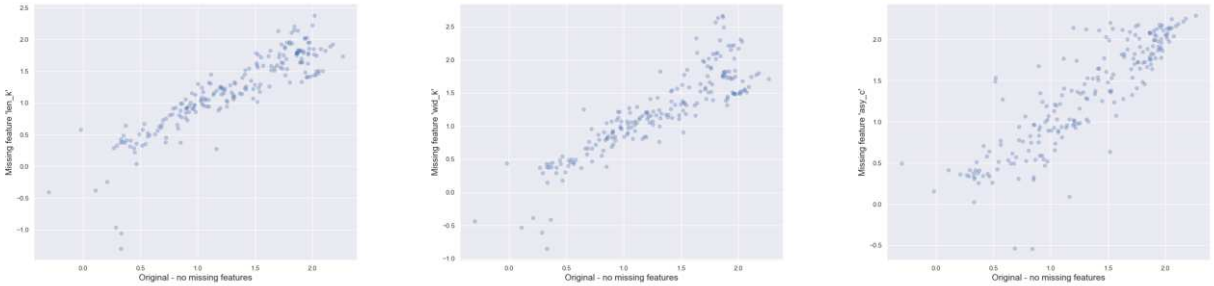
Source: Author's own elaboration

Figure 42 – Discrimination dispersion for the Wheat dataset - Simple Imputation

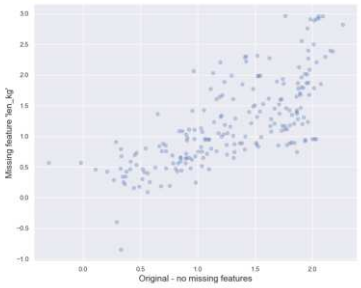
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



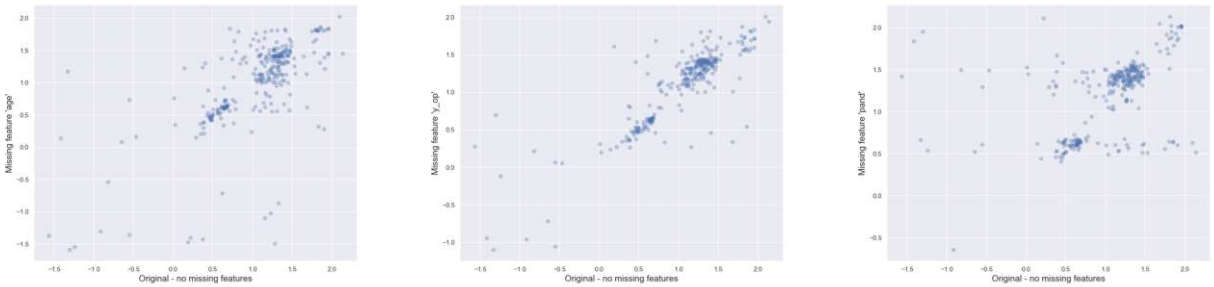
(vii) len_kg



Source: Author's own elaboration

Figure 43 – Discrimination dispersion for the Haberman dataset - Simple Imputation

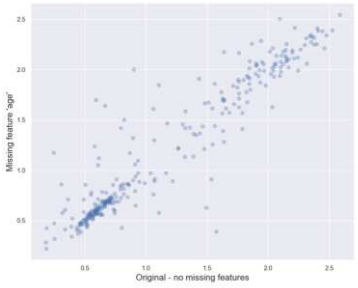
(i) age (ii) y_op (iii) pand



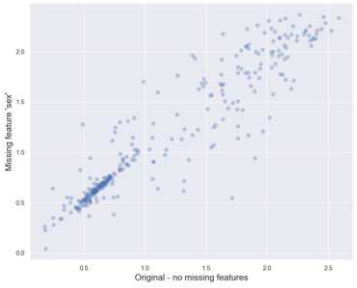
Source: Author's own elaboration

Figure 44 – Discrimination dispersion for the Cleveland dataset - Simple Imputation

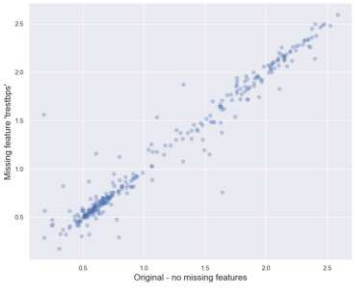
(i) age



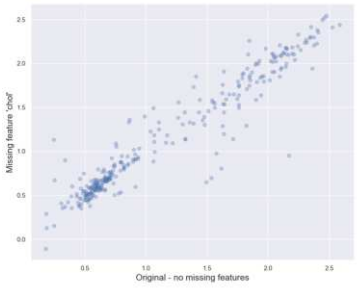
(ii) sex



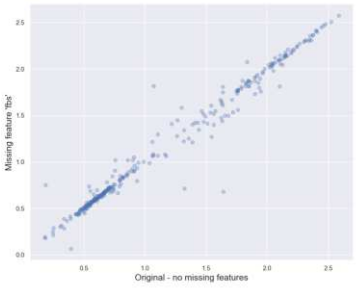
(iii) trestbps



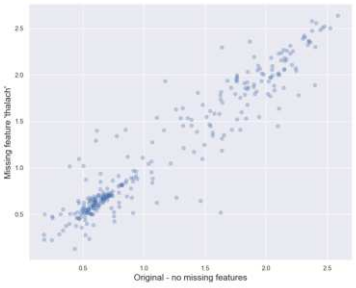
(iv) chol



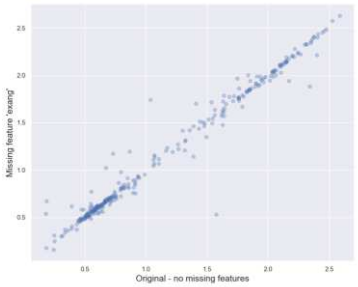
(v) fbs



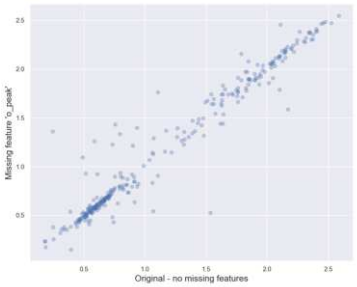
(vi) thalach



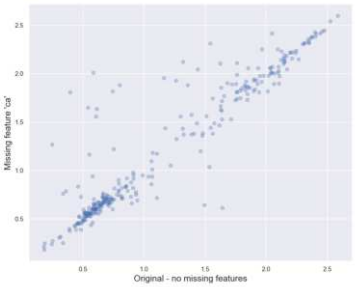
(vii) exang



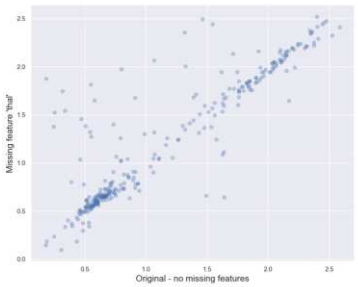
(viii) o_peak



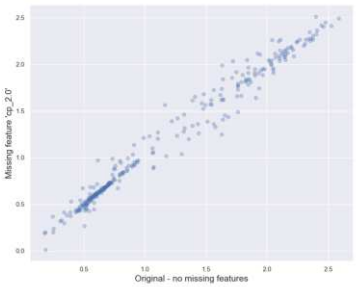
(ix) ca



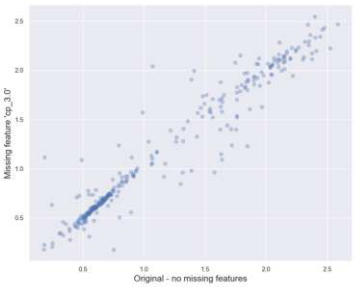
(x) thal

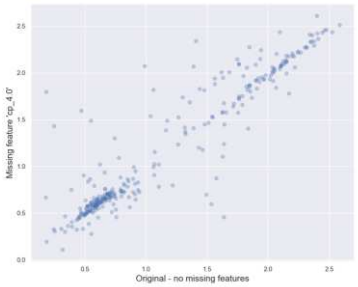


(xi) cp_2.0

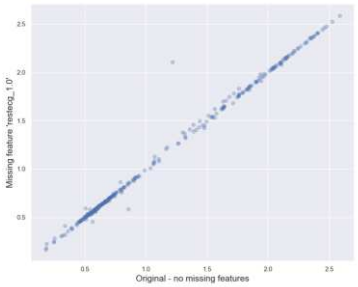


(xii) cp_3.0

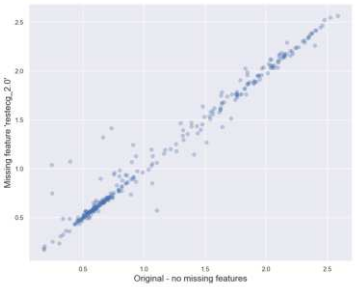




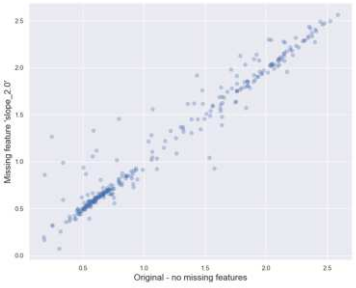
(xiii) cp_4.0



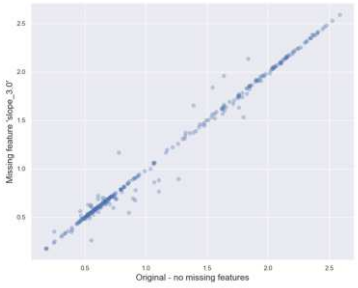
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

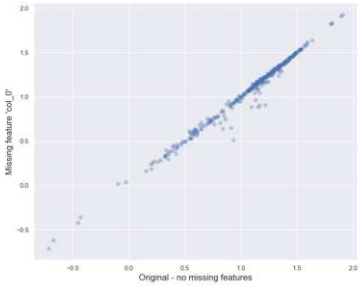


(xvii) slope_3.0

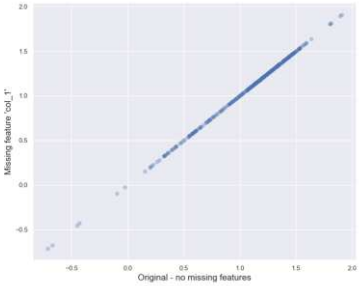
Source: Author's own elaboration

Figure 44 – Discrimination dispersion for the Ionosphere dataset - Simple Imputation

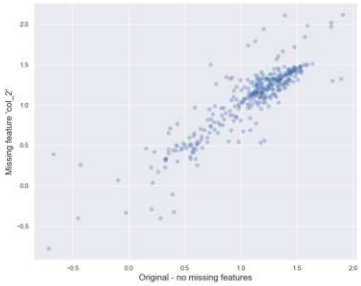
(i) col_0



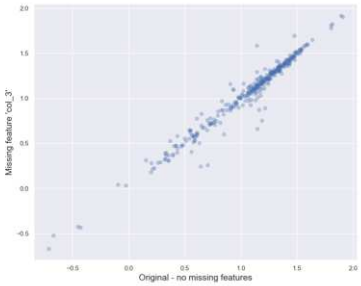
(ii) col_1



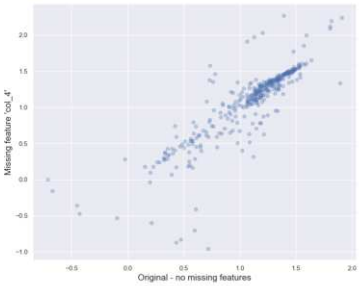
(iii) col_2



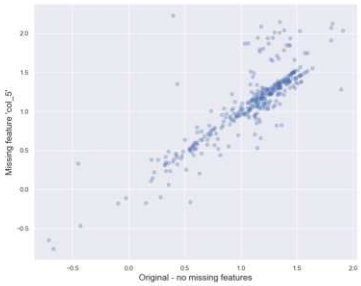
(iv) col_3



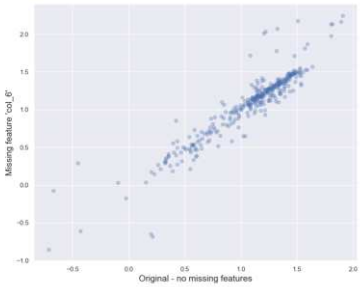
(v) col_4



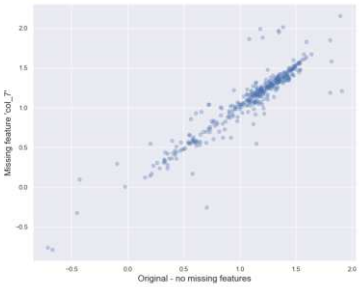
(vi) col_5



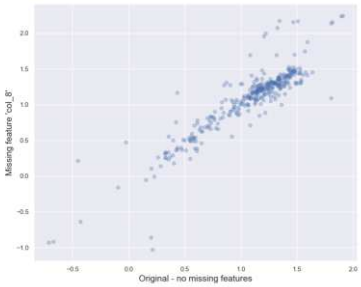
(vii) col_6



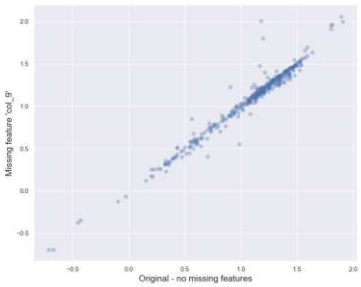
(viii) col_7



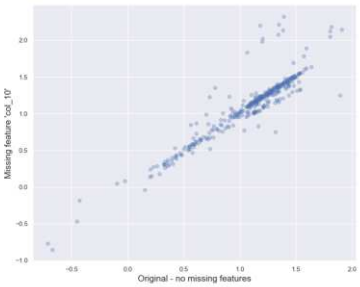
(ix) col_8



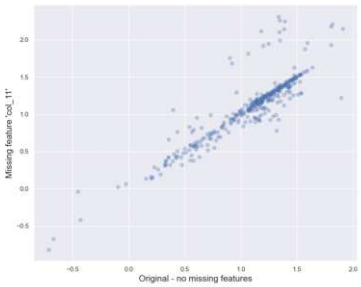
(x) col_9

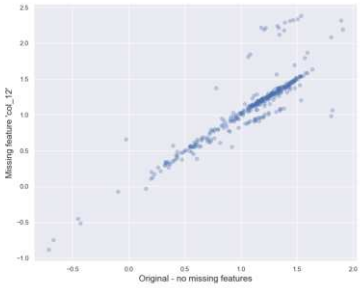


(xi) col_10

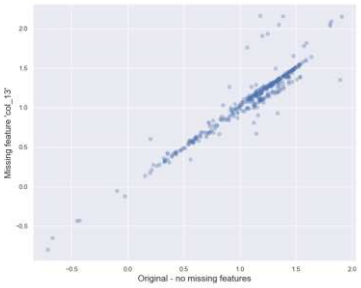


(xii) col_11

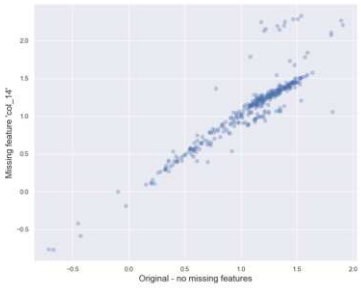




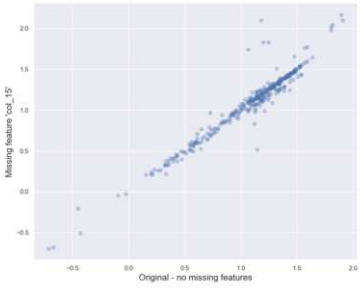
(xiii) col_12



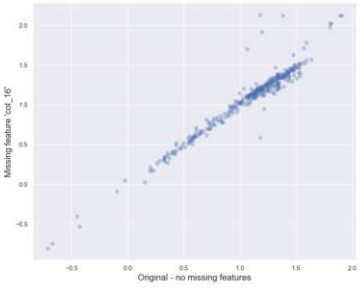
(xiv) col_13



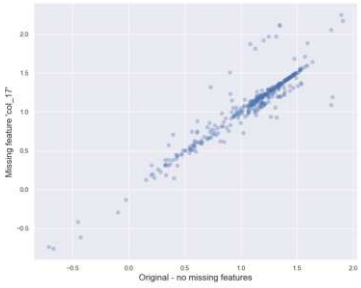
(xv) col_14



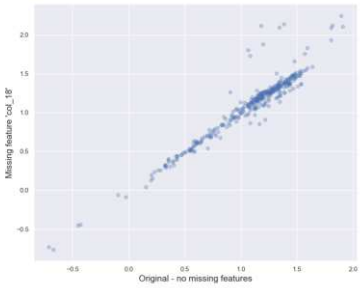
(xvi) col_15



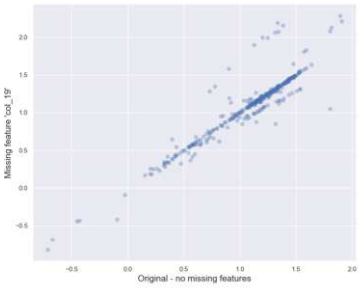
(xvii) col_16



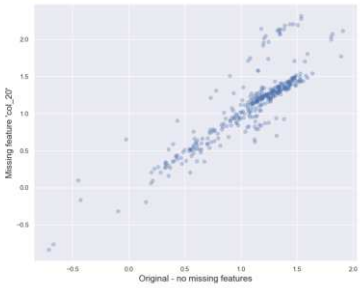
(xviii) col_17



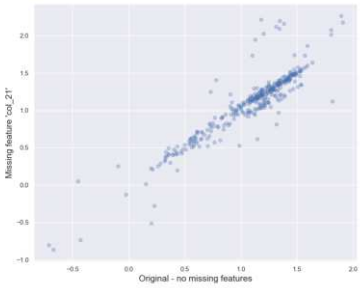
(xix) col_18



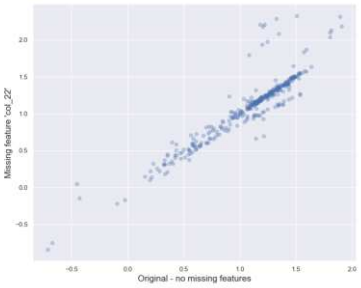
(xx) col_19



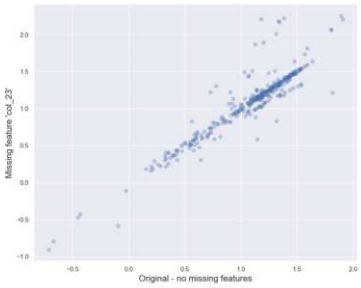
(xxi) col_20



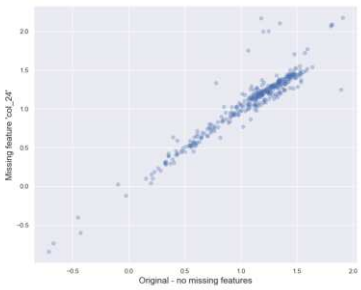
(xxii) col_21



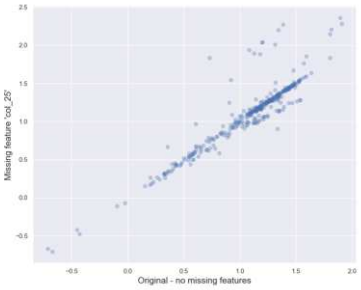
(xxiii) col_22



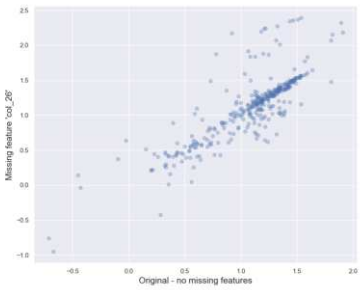
(xxiv) col_23



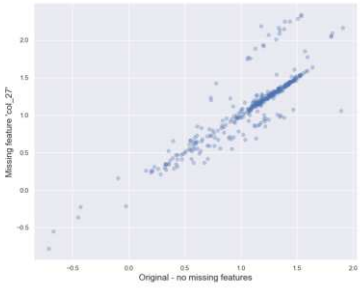
(xxv) col_24



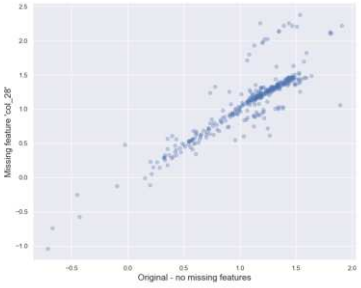
(xxvi) col_25



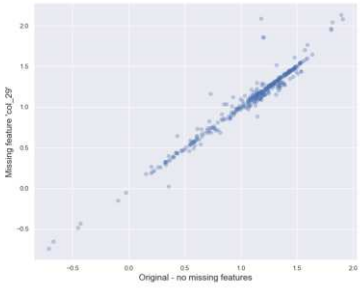
(xxvii) col_26



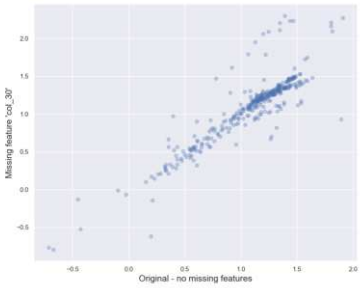
(xxviii) col_27



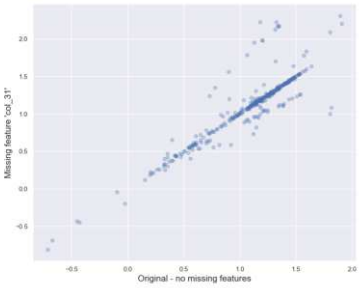
(xxix) col_28



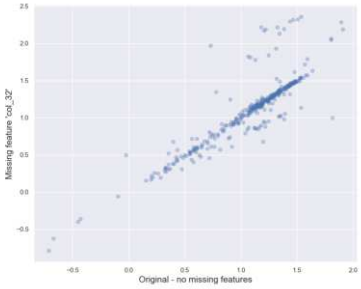
(xxx) col_29



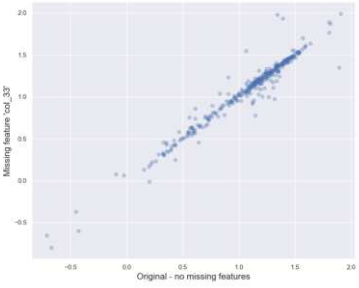
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

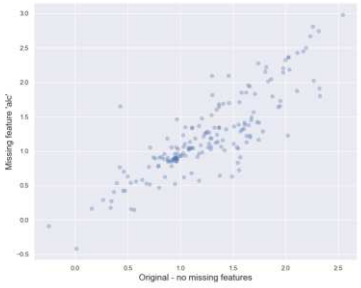


(xxxiv) col_33

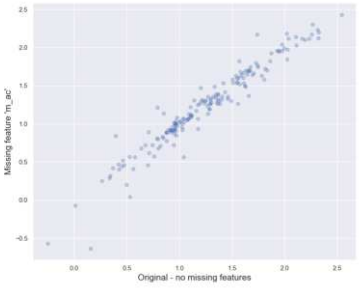
Source: Author's own elaboration

Figure 43 – Discrimination dispersion for the Wine dataset - Simple Imputation

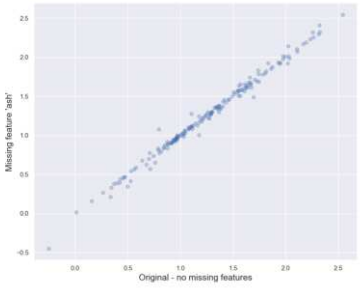
(i) alc



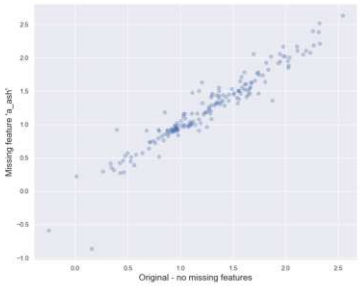
(ii) m_ac



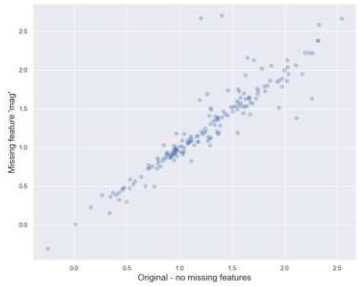
(iii) ash



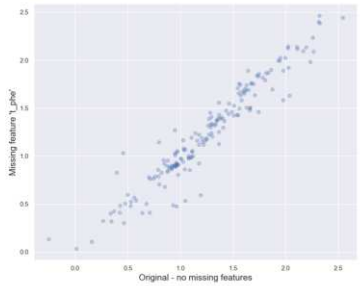
(iv) a_ash



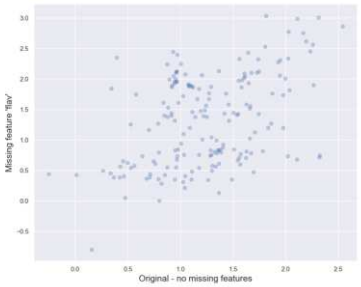
(v) mag



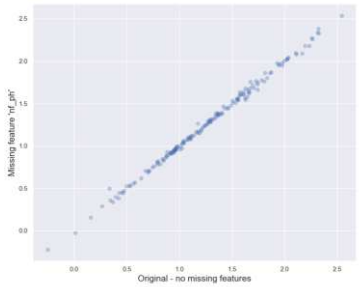
(vi) t_phe



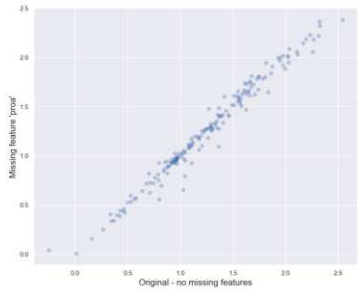
(vii) flav



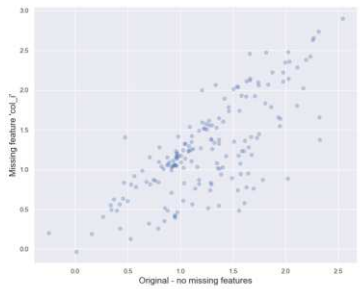
(viii) nf_ph



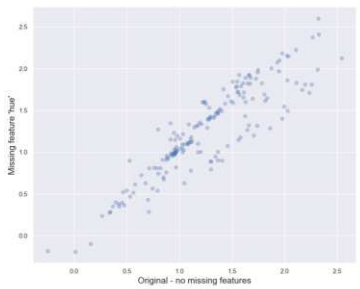
(ix) proa



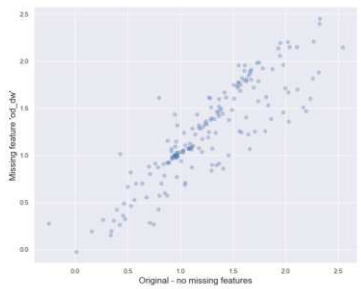
(x) col_i

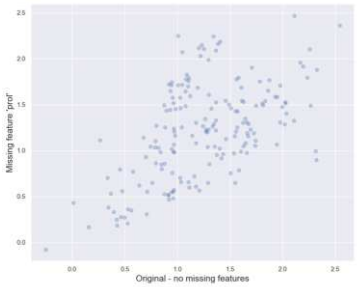


(xi) hue



(xii) od_dw



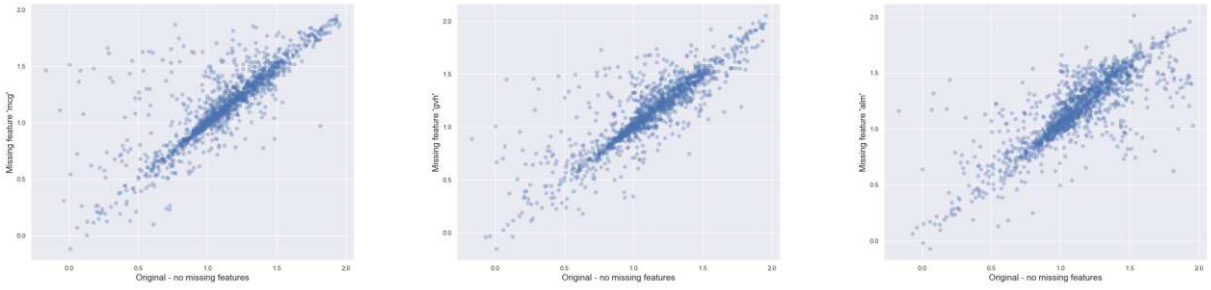


(xiii) prol

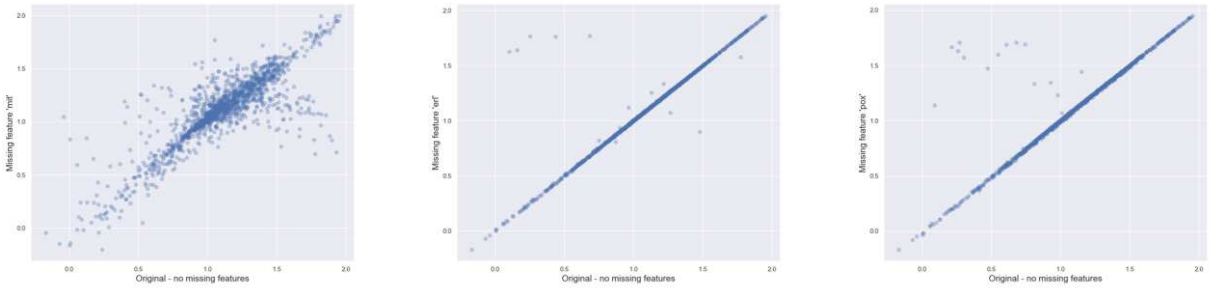
Source: Author's own elaboration

Figure 43 – Discrimination dispersion for the Yeast dataset - Simple Imputation

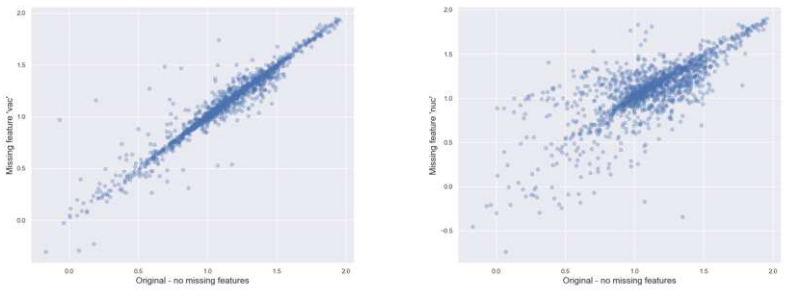
(i) mcg (ii) gvh (iii) alm



(iv) mit (v) erl (vi) pox



(vii) vac (viii) nuc



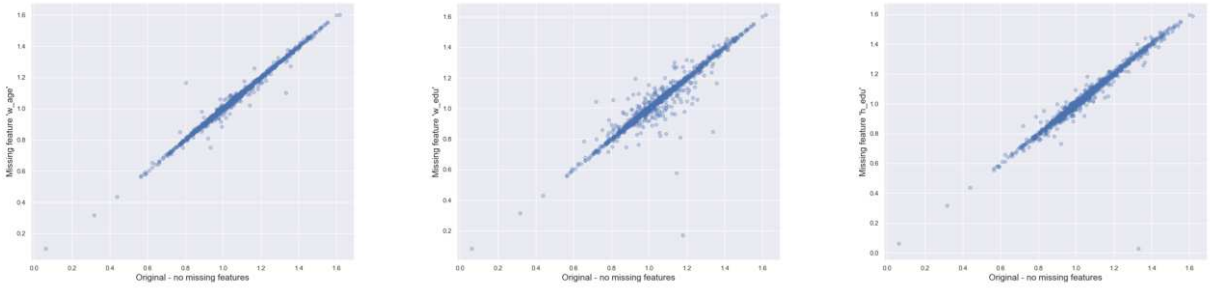
Source: Author's own elaboration

C.2 KNN

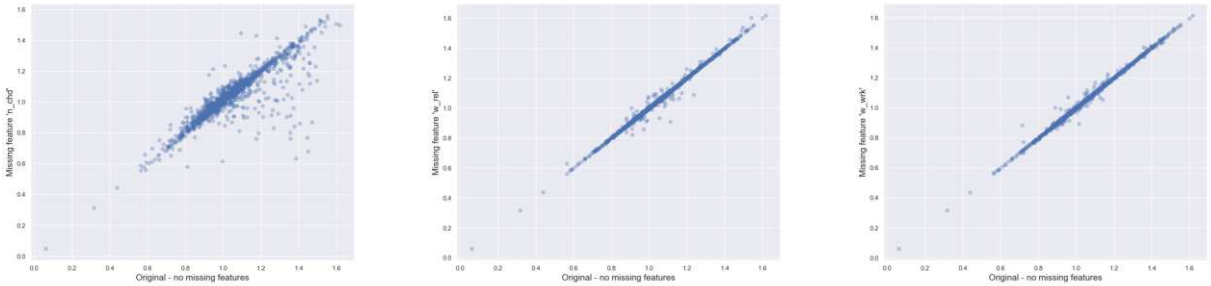
Figure 44 shows the results for the Contraceptive dataset, Figure 45 for the Pima dataset, Figure 46 for the Wheat dataset, Figure 47 for the Haberman dataset, Figure 48 for the Cleveland dataset, Figure 48 for the Ionosphere dataset, Figure 47 for the Wine dataset, and finally, Figure 47 shows the results for the Yeast dataset.

Figure 44 – Discrimination dispersion for the Contraceptive dataset - KNN Imputation

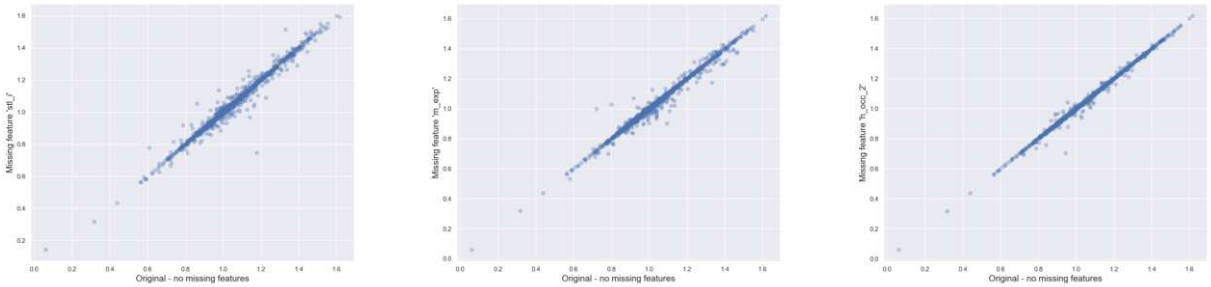
(i) w_age (ii) w_edu (iii) h_edu



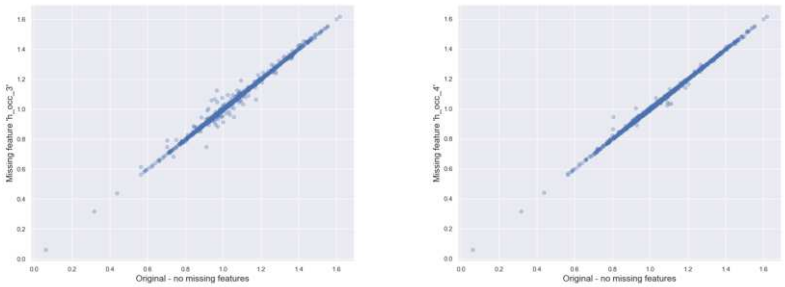
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



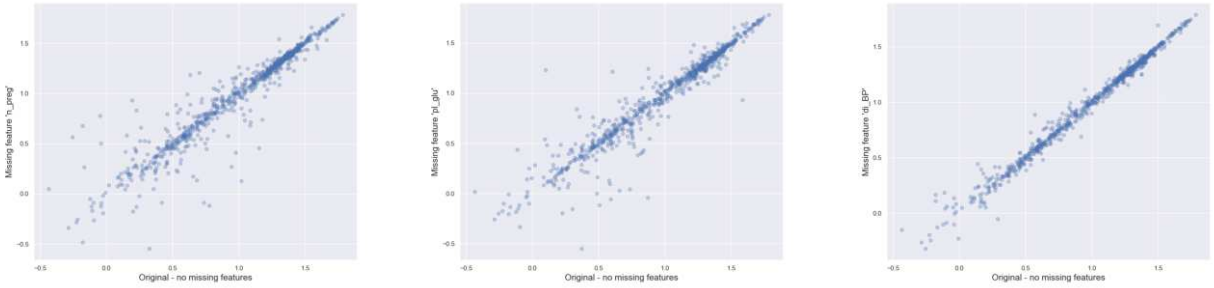
(x) h_occ_3 (xi) h_occ_4



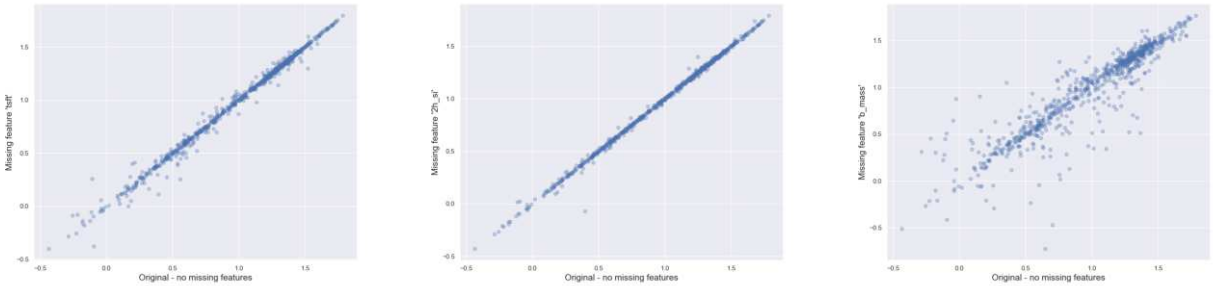
Source: Author's own elaboration

Figure 45 – Discrimination dispersion for the Pima dataset - KNN Imputation

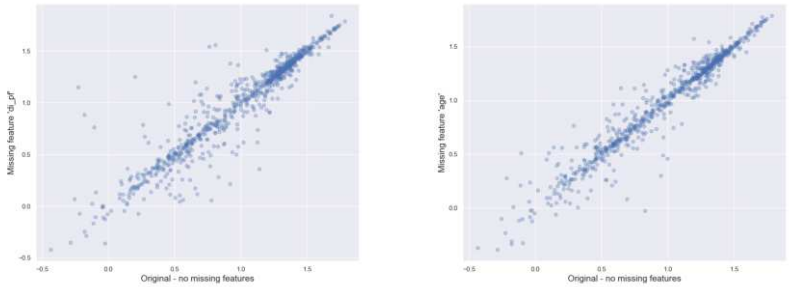
(i) n_preg (ii) pl_glu (iii) di_BP



(iv) tsft (v) 2h_si (vi) b_mass



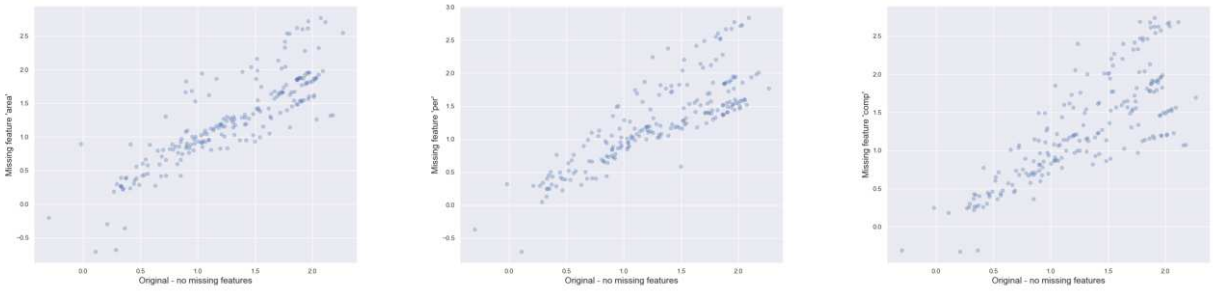
(vii) di_pf (viii) age



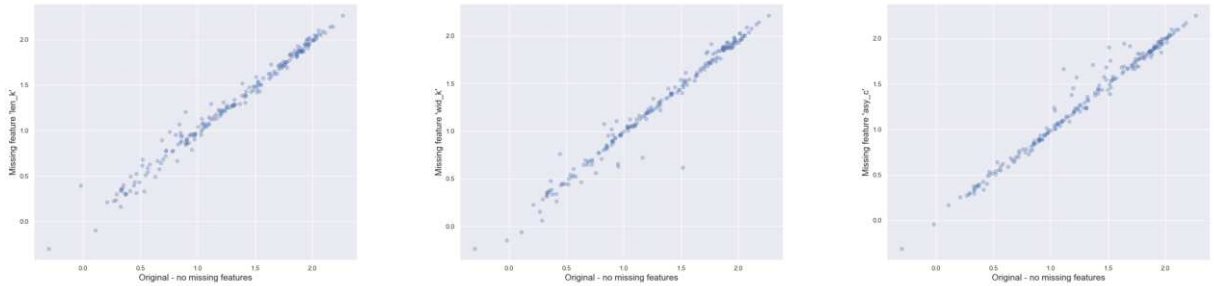
Source: Author's own elaboration

Figure 46 – Discrimination dispersion for the Wheat dataset - KNN Imputation

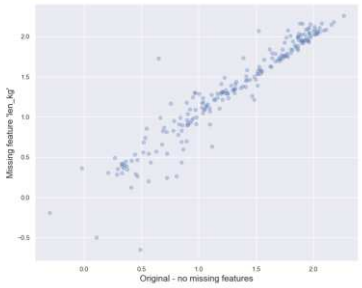
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



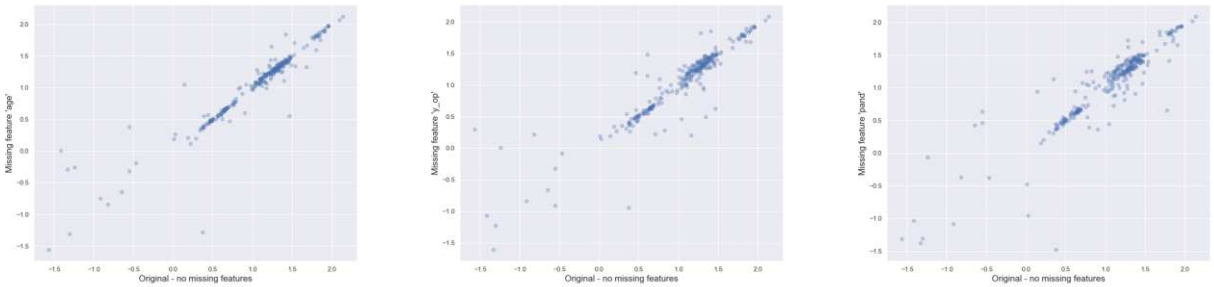
(vii) len_kg



Source: Author's own elaboration

Figure 47 – Discrimination dispersion for the Haberman dataset - KNN Imputation

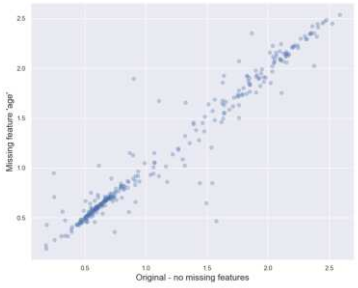
(i) age (ii) y_op (iii) pand



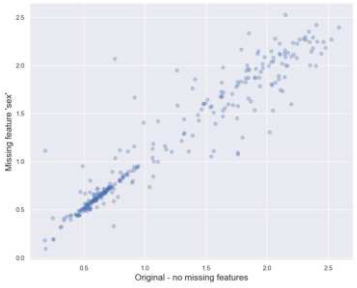
Source: Author's own elaboration

Figure 48 – Discrimination dispersion for the Cleveland dataset - KNN Imputation

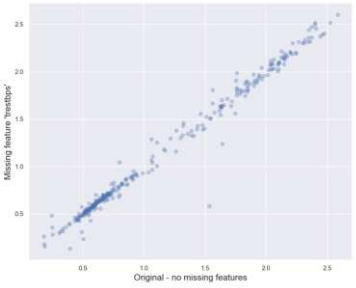
(i) age



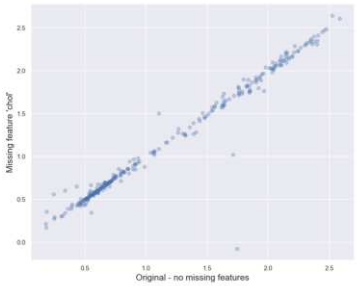
(ii) sex



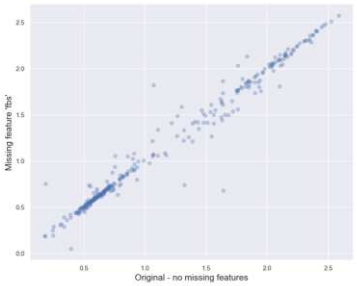
(iii) trestbps



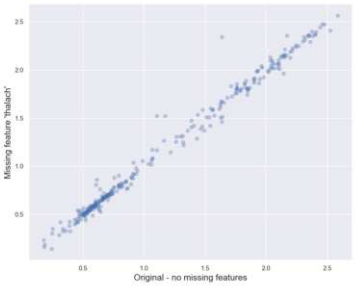
(iv) chol



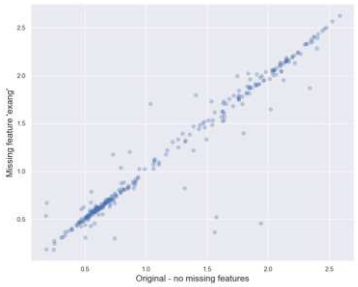
(v) fbs



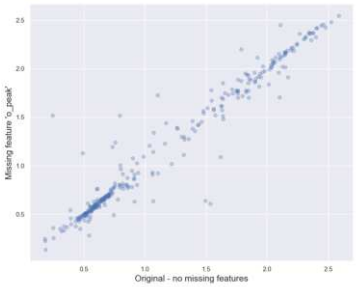
(vi) thalach



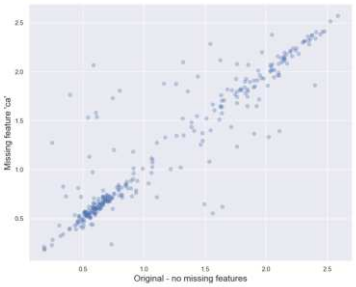
(vii) exang



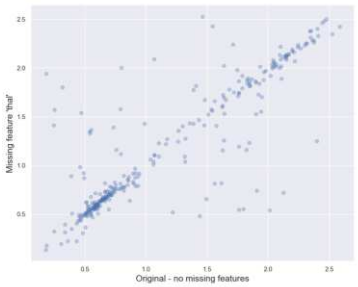
(viii) o_peak



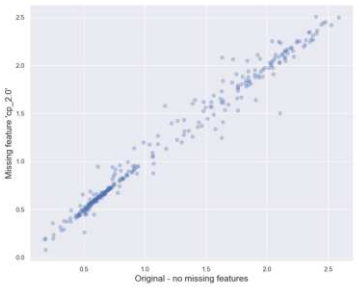
(ix) ca



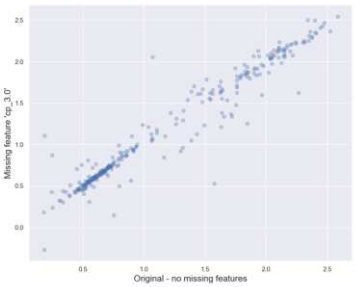
(x) thal

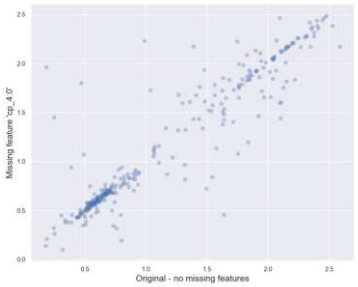


(xi) cp_2.0

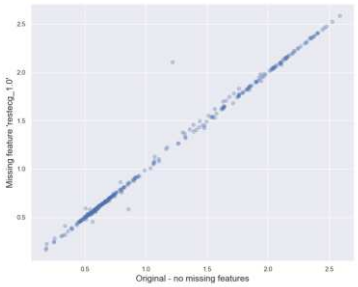


(xii) cp_3.0

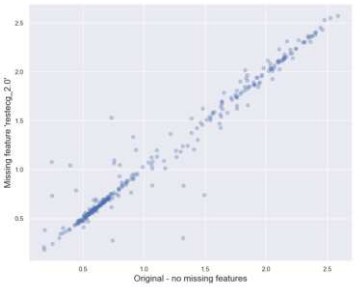




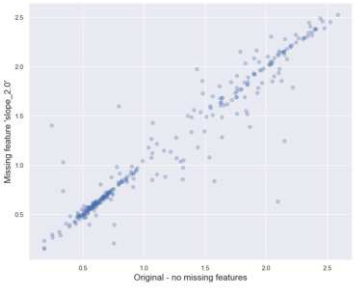
(xiii) cp_4.0



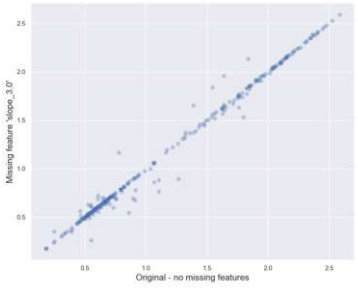
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

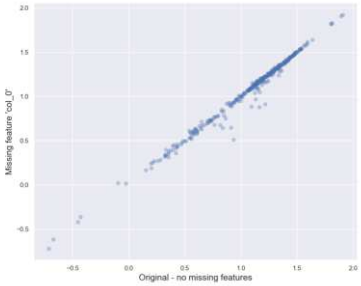


(xvii) slope_3.0

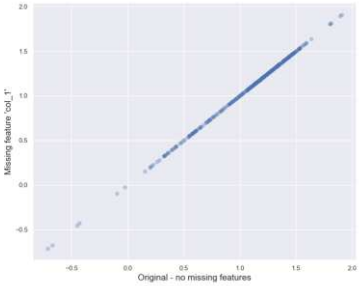
Source: Author's own elaboration

Figure 48 – Discrimination dispersion for the Ionosphere dataset - KNN Imputation

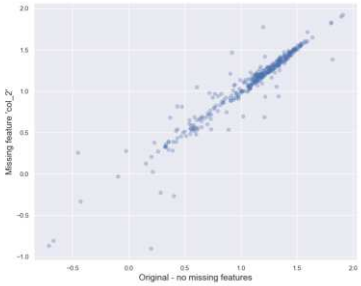
(i) col_0



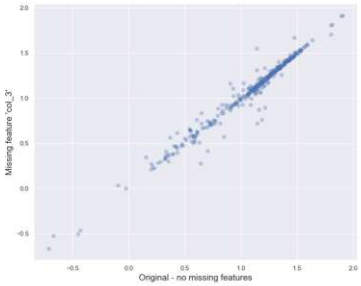
(ii) col_1



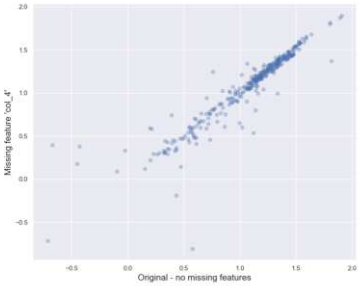
(iii) col_2



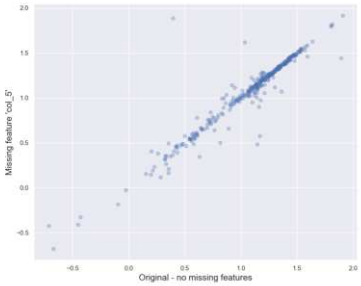
(iv) col_3



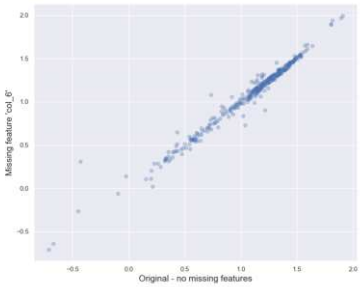
(v) col_4



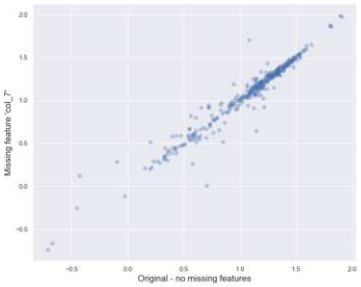
(vi) col_5



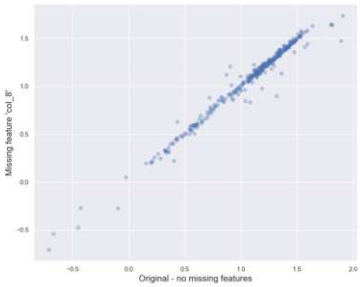
(vii) col_6



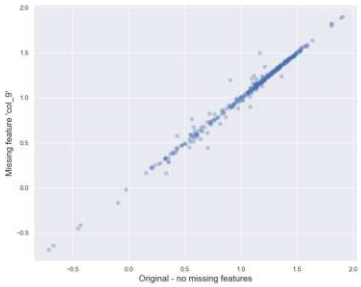
(viii) col_7



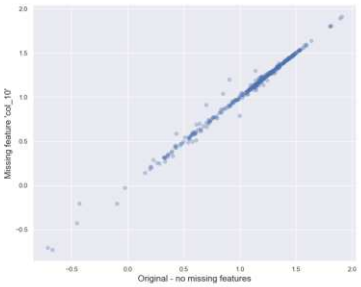
(ix) col_8



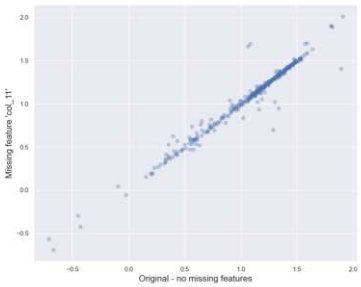
(x) col_9

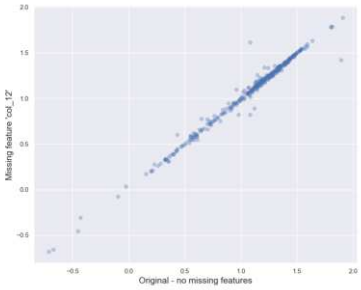


(xi) col_10

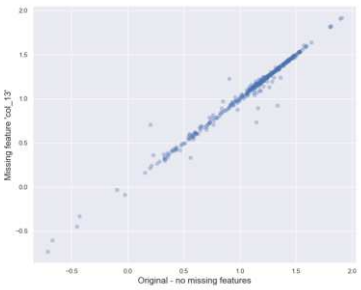


(xii) col_11

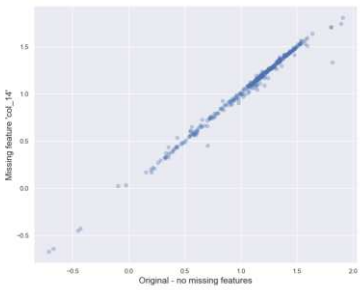




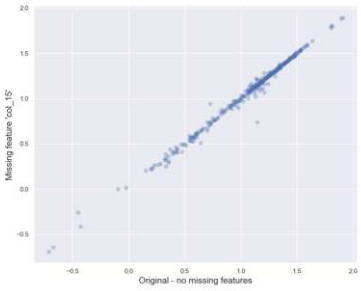
(xiii) col_12



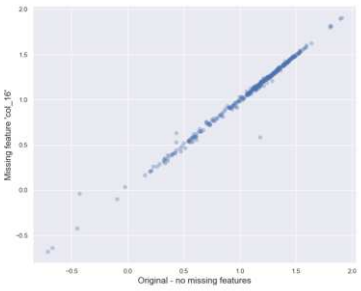
(xiv) col_13



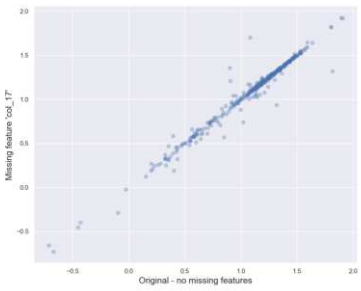
(xv) col_14



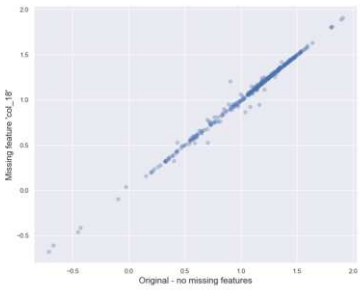
(xvi) col_15



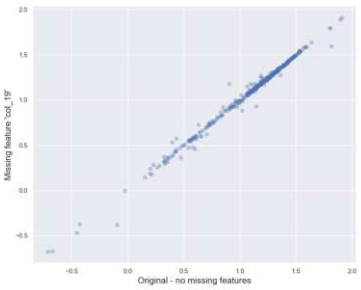
(xvii) col_16



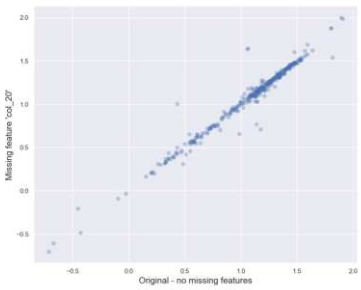
(xviii) col_17



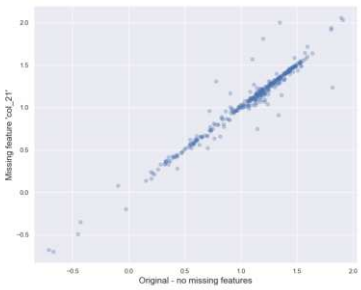
(xix) col_18



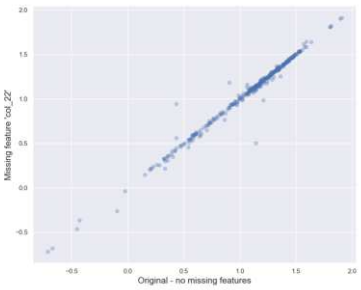
(xx) col_19



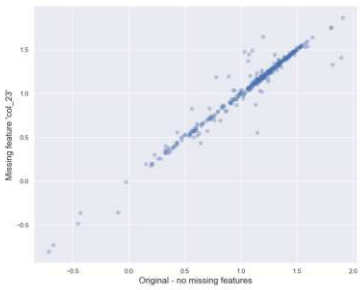
(xxi) col_20



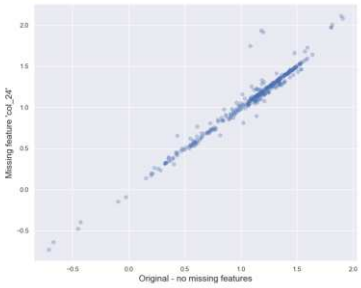
(xxii) col_21



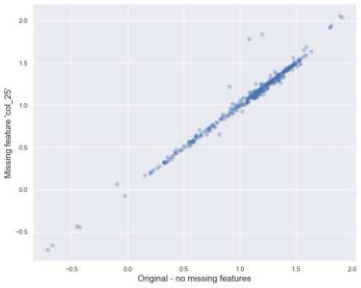
(xxiii) col_22



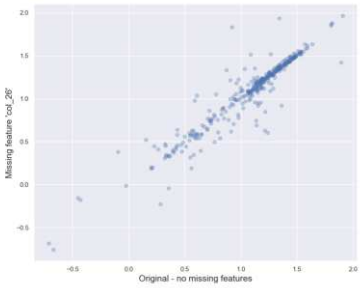
(xxiv) col_23



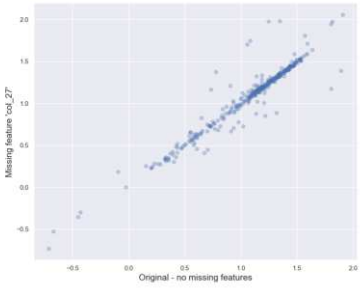
(xxv) col_24



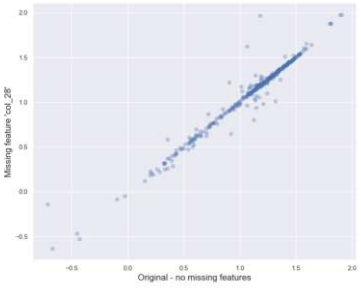
(xxvi) col_25



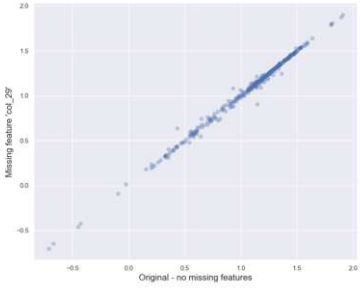
(xxvii) col_26



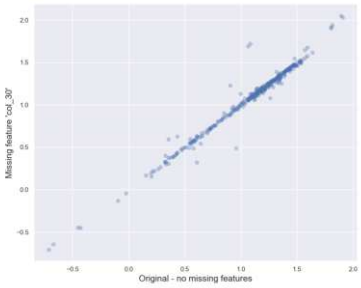
(xxviii) col_27



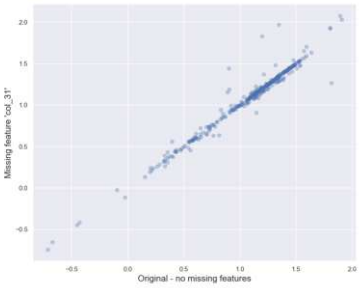
(xxix) col_28



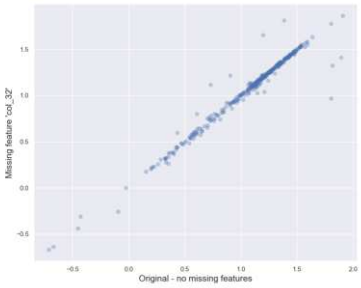
(xxx) col_29



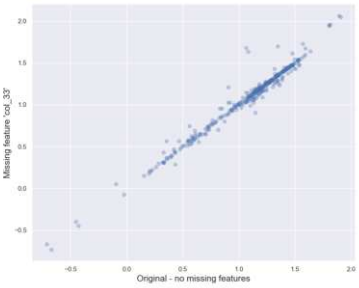
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

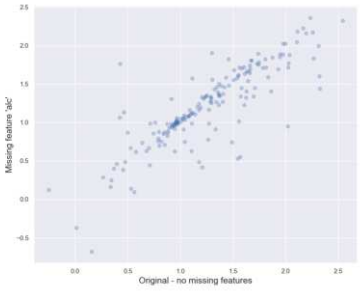


(xxxiv) col_33

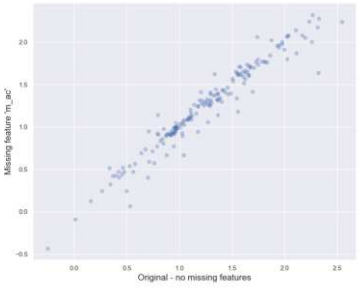
Source: Author's own elaboration

Figure 47 – Discrimination dispersion for the Wine dataset - KNN Imputation

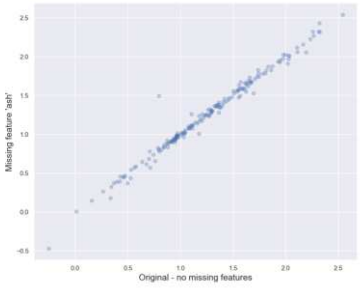
(i) alc



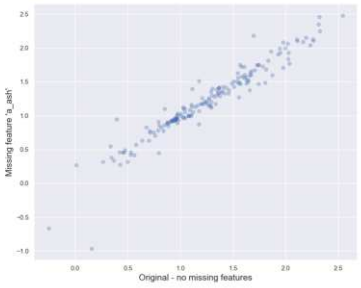
(ii) m_ac



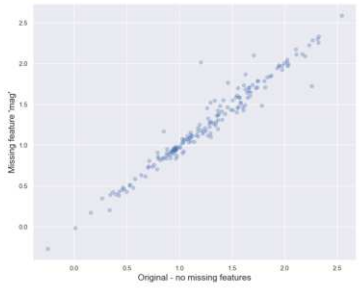
(iii) ash



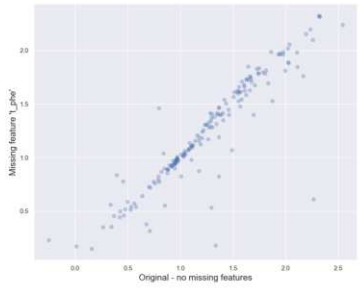
(iv) a_ash



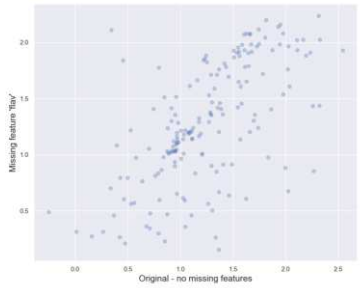
(v) mag



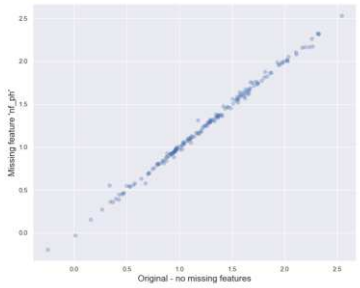
(vi) t_phe



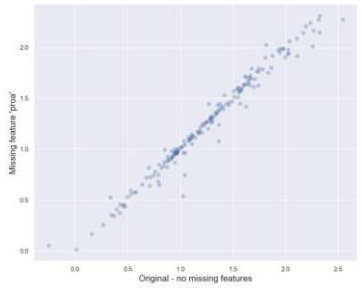
(vii) flav



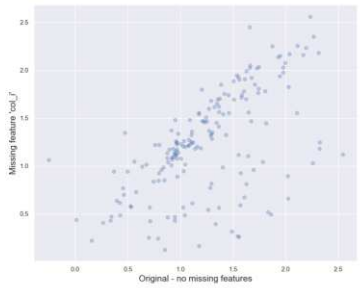
(viii) nf_ph



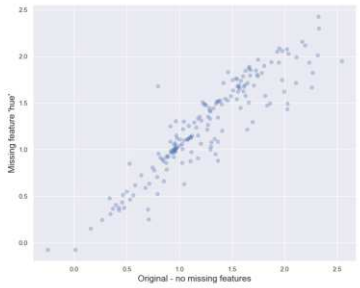
(ix) proa



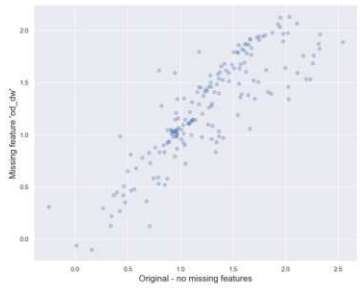
(x) col_i

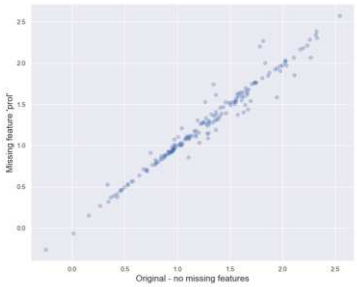


(xi) hue



(xii) od_dw



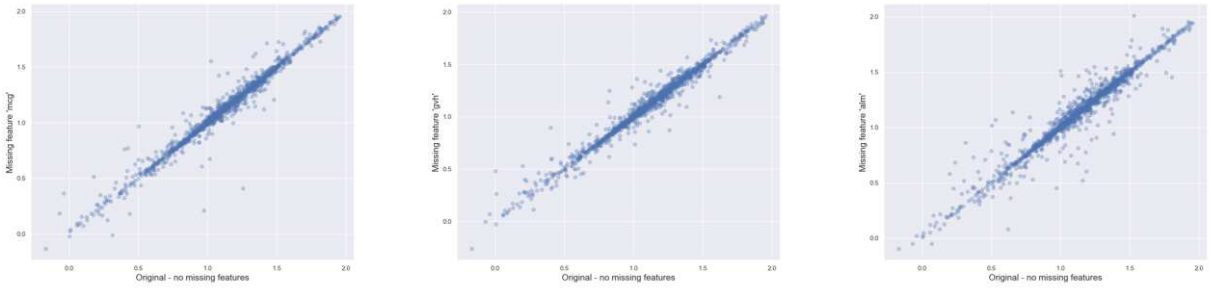


(xiii) prol

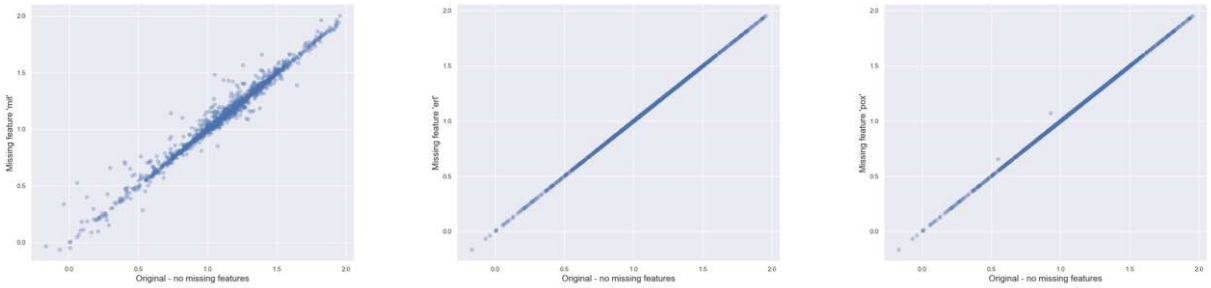
Source: Author's own elaboration

Figure 47 – Discrimination dispersion for the Yeast dataset - KNN Imputation

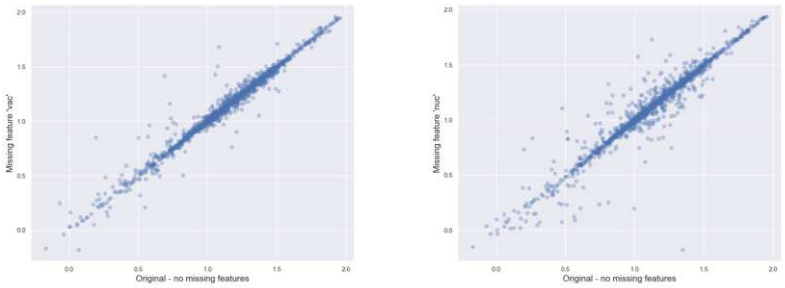
(i) mcg (ii) gvh (iii) alm



(iv) mit (v) erl (vi) pox



(vii) vac (viii) nuc



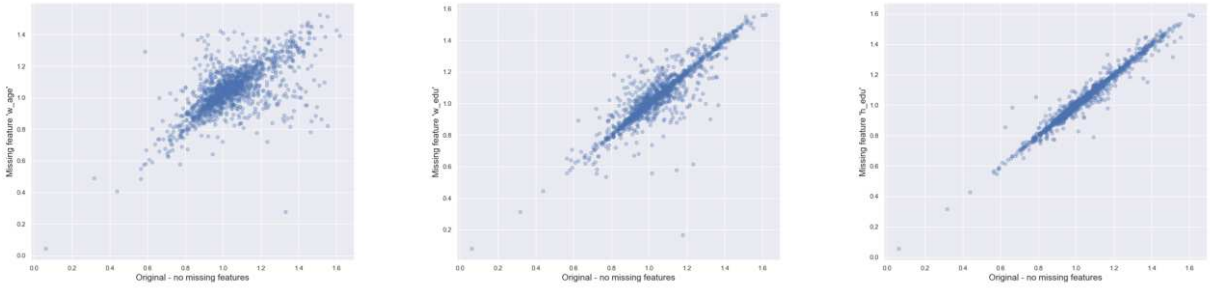
Source: Author's own elaboration

C.3 ITERATIVE

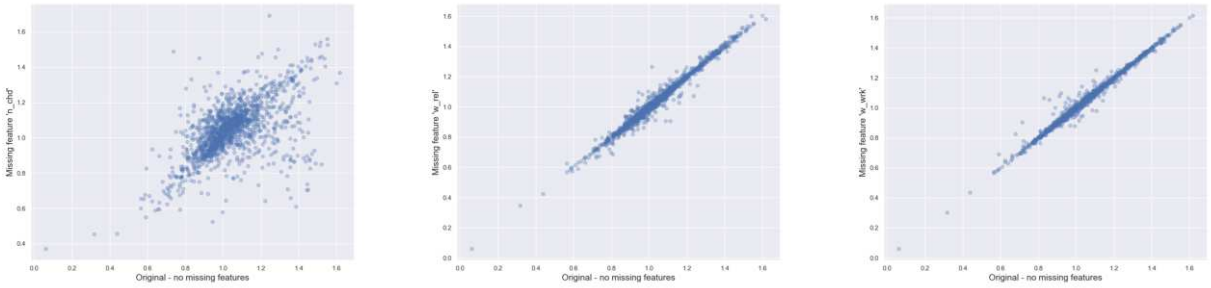
Figure 48 shows the results for the Contraceptive dataset, Figure 49 for the Pima dataset, Figure 50 for the Wheat dataset, Figure 51 for the Haberman dataset, Figure 52 for the Cleveland dataset, Figure 52 for the Ionosphere dataset, Figure 51 for the Wine dataset, and finally, Figure 51 shows the results for the Yeast dataset.

Figure 48 – Discrimination dispersion for the Contraceptive dataset - Iterative Imputation

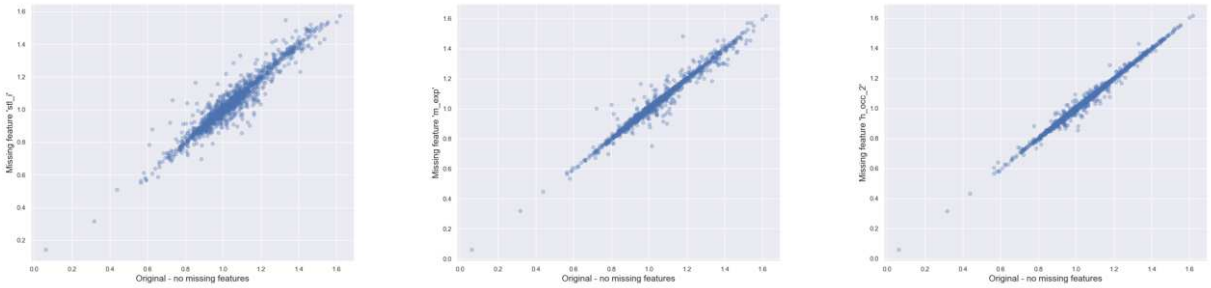
(i) w_age (ii) w_edu (iii) h_edu



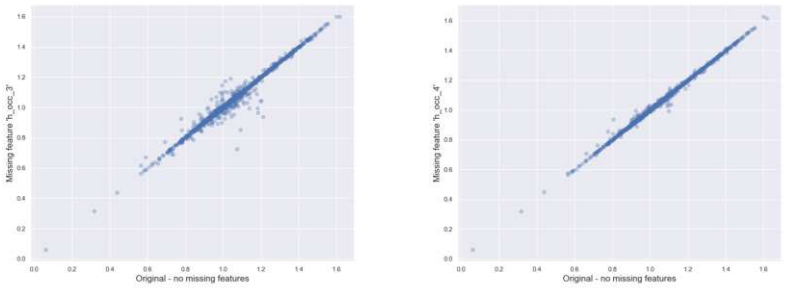
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



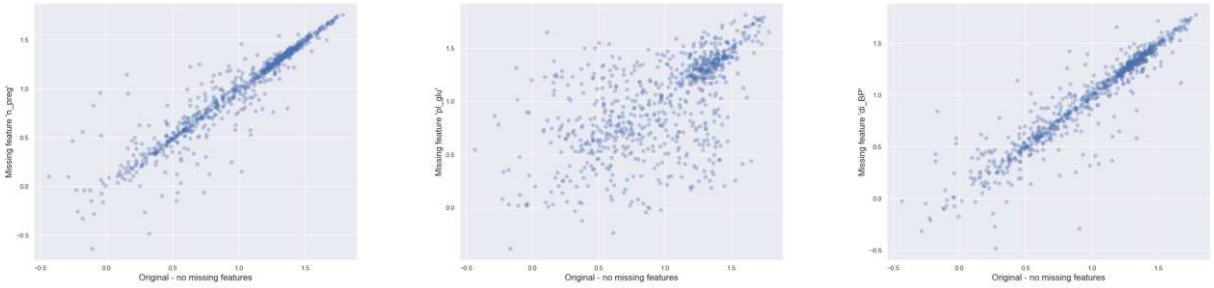
(x) h_occ_3 (xi) h_occ_4



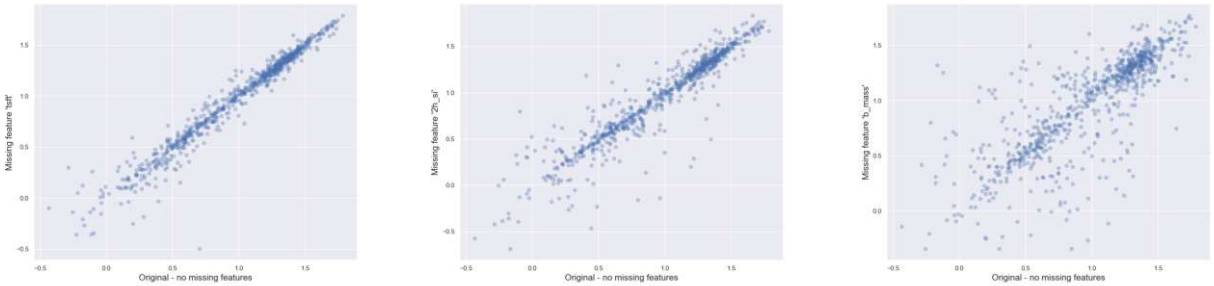
Source: Author's own elaboration

Figure 49 – Discrimination dispersion for the Pima dataset - Iterative Imputation

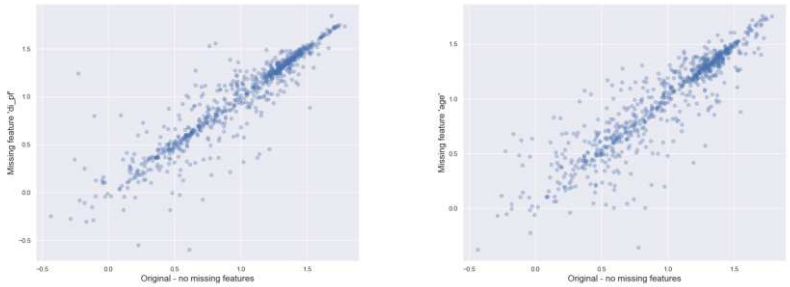
(i) n_preg (ii) pl_glu (iii) di_BP



(iv) tsft (v) 2h_si (vi) b_mass



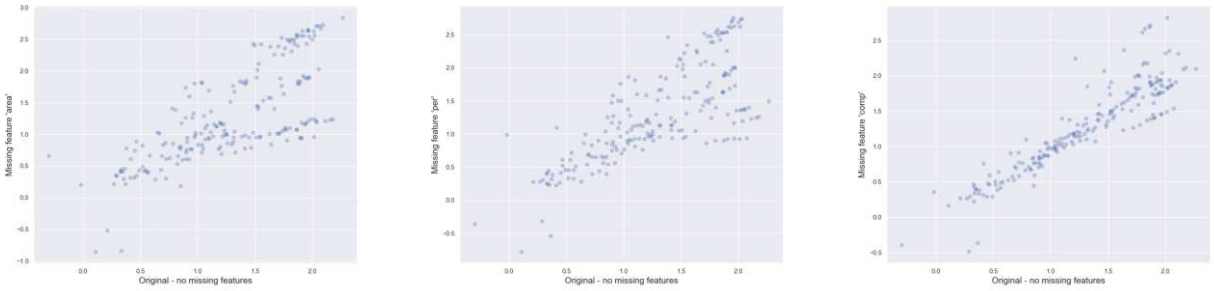
(vii) di_pf (viii) age



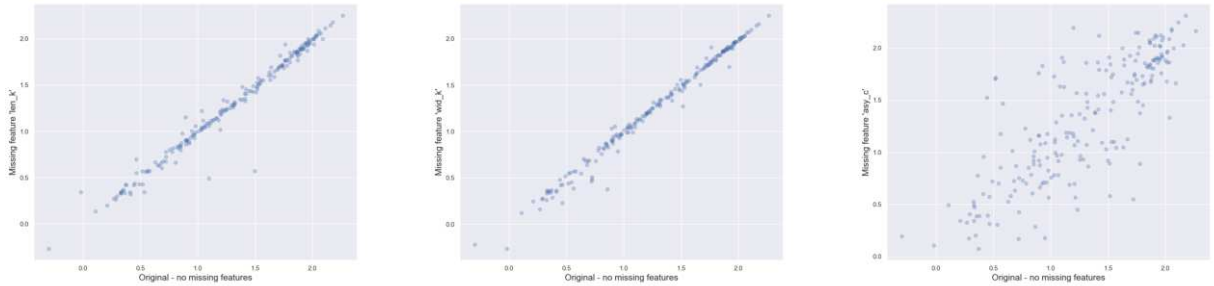
Source: Author's own elaboration

Figure 50 – Discrimination dispersion for the Wheat dataset - Iterative Imputation

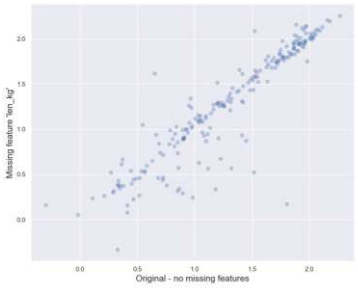
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



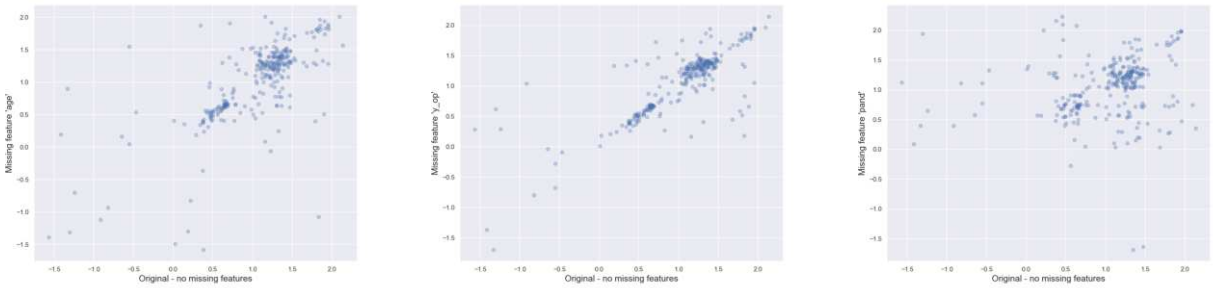
(vii) len_kg



Source: Author's own elaboration

Figure 51 – Discrimination dispersion for the Haberman dataset - Iterative Imputation

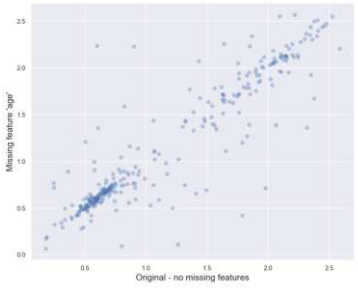
(i) age (ii) y_op (iii) pand



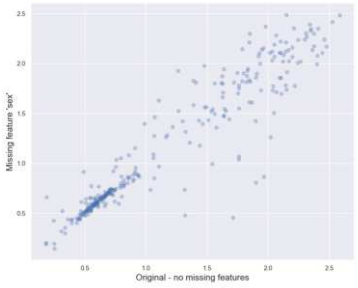
Source: Author's own elaboration

Figure 52 – Discrimination dispersion for the Cleveland dataset - Iterative Imputation

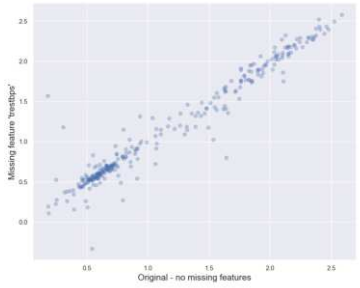
(i) age



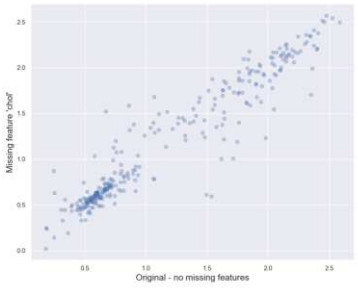
(ii) sex



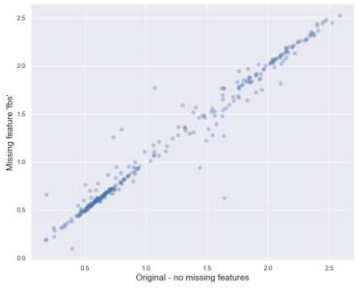
(iii) trestbps



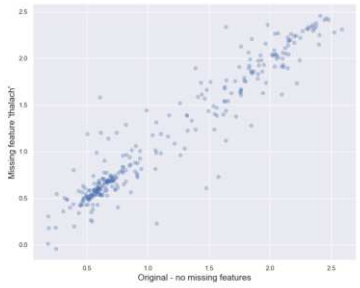
(iv) chol



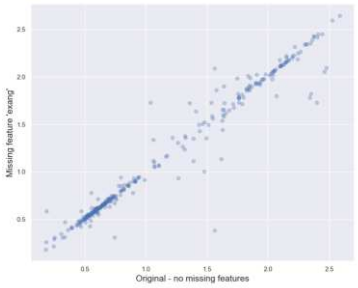
(v) fbs



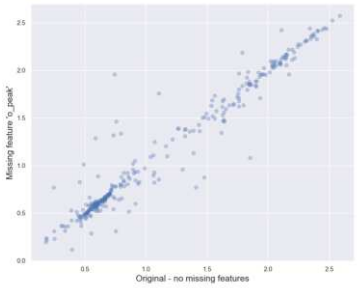
(vi) thalach



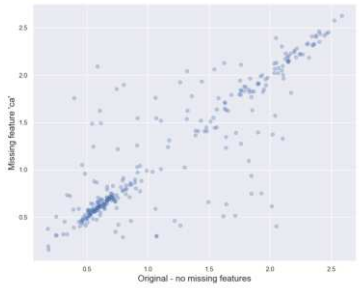
(vii) exang



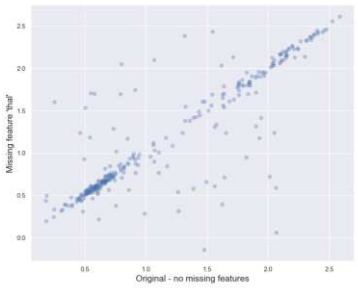
(viii) o_peak



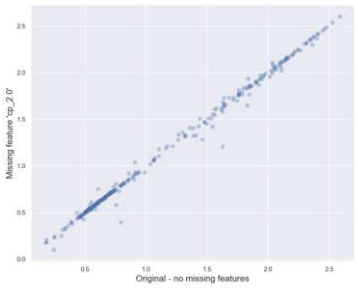
(ix) ca



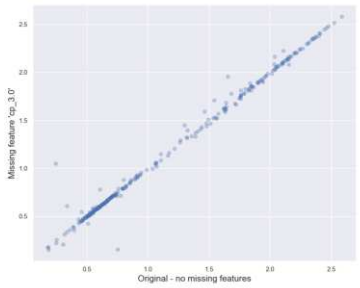
(x) thal

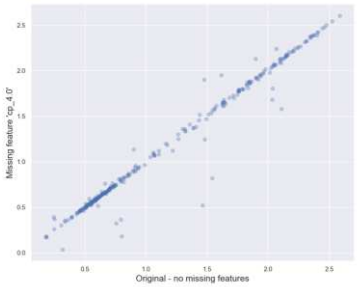


(xi) cp_2.0

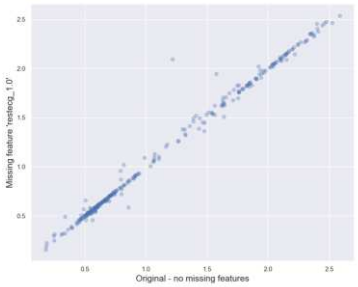


(xii) cp_3.0

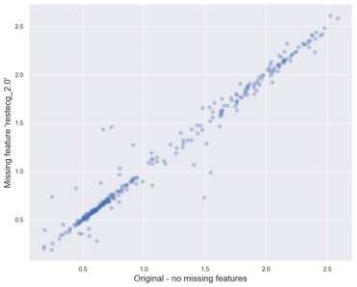




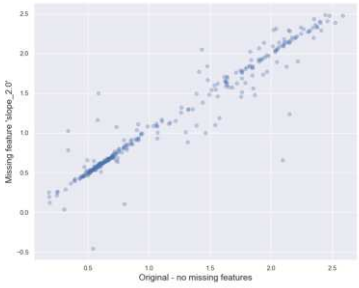
(xiii) cp_4.0



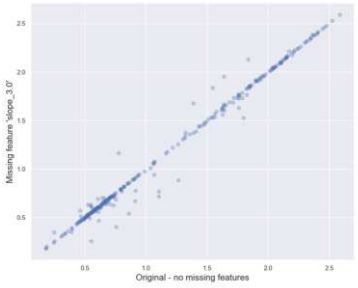
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

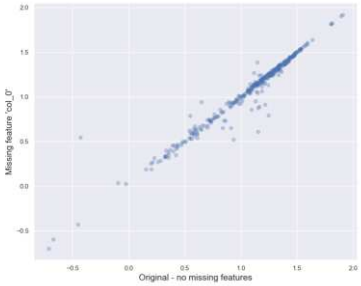


(xvii) slope_3.0

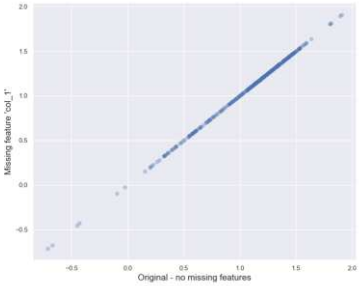
Source: Author's own elaboration

Figure 52 – Discrimination dispersion for the Ionosphere dataset - Iterative Imputation

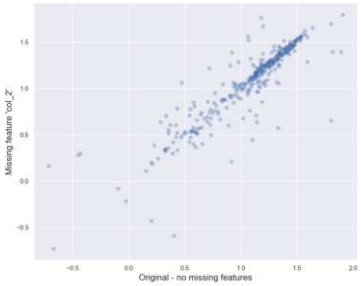
(i) col_0



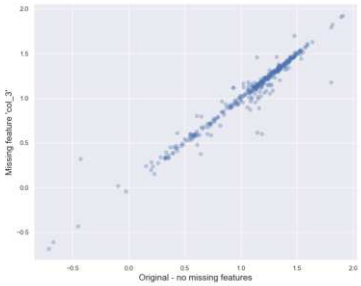
(ii) col_1



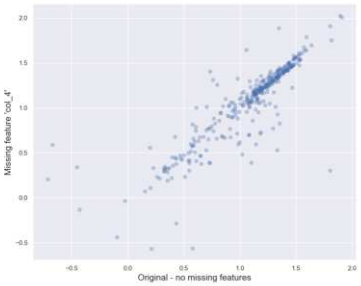
(iii) col_2



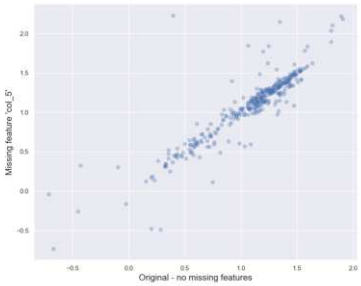
(iv) col_3



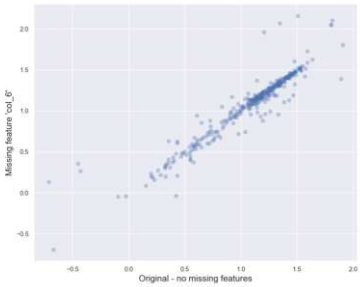
(v) col_4



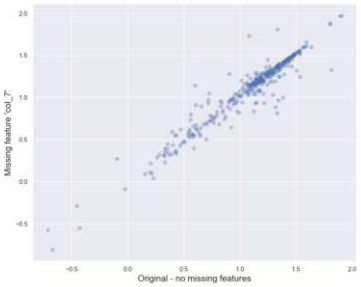
(vi) col_5



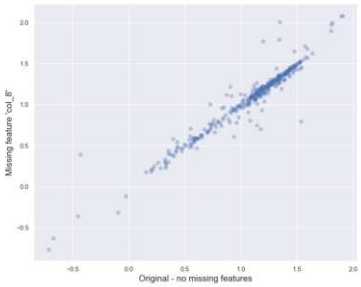
(vii) col_6



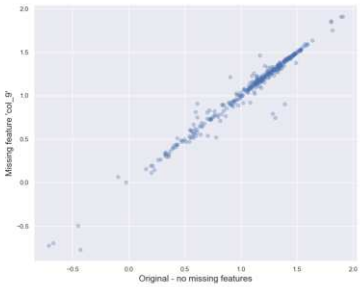
(viii) col_7



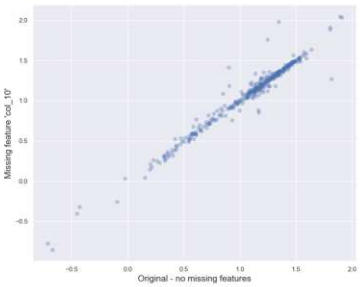
(ix) col_8



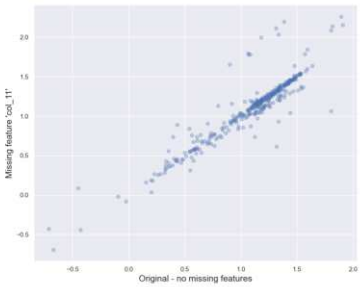
(x) col_9

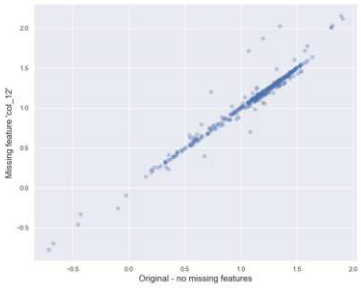


(xi) col_10

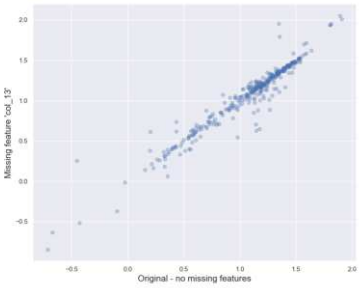


(xii) col_11

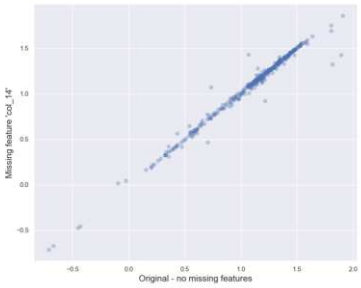




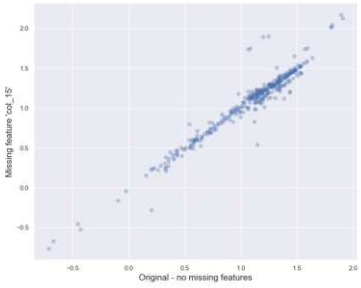
(xiii) col_12



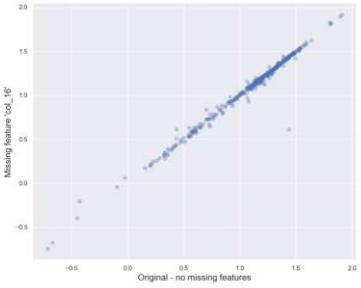
(xiv) col_13



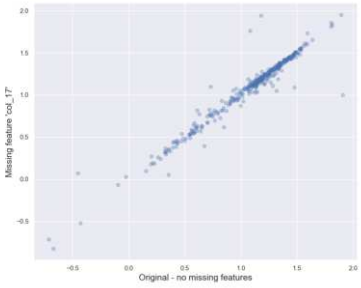
(xv) col_14



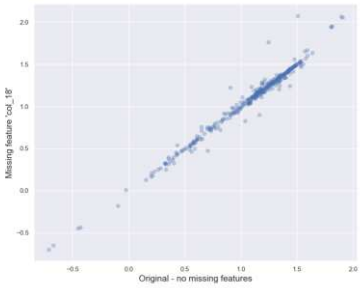
(xvi) col_15



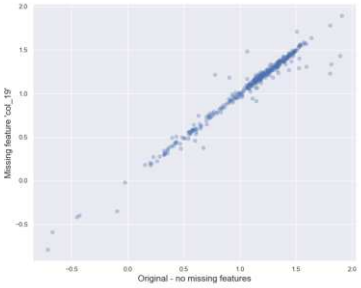
(xvii) col_16



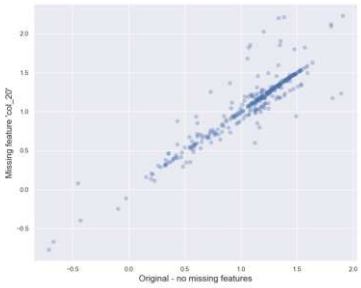
(xviii) col_17



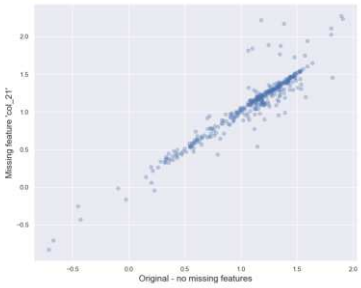
(xix) col_18



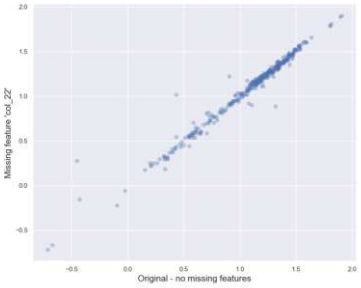
(xx) col_19



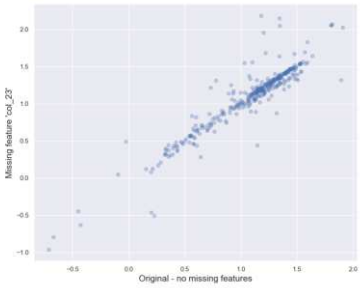
(xxi) col_20



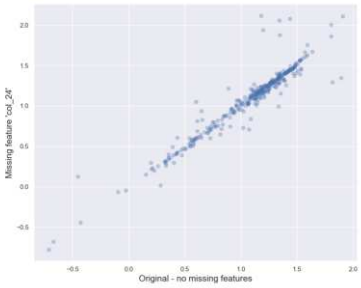
(xxii) col_21



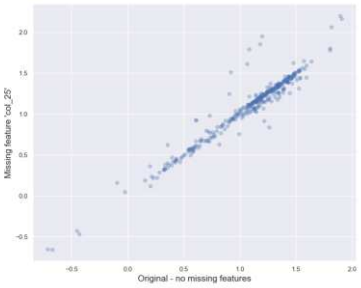
(xxiii) col_22



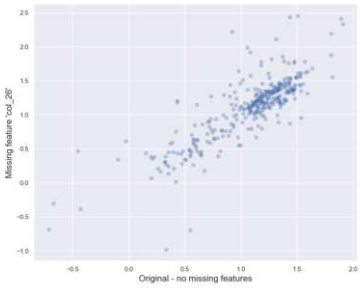
(xxiv) col_23



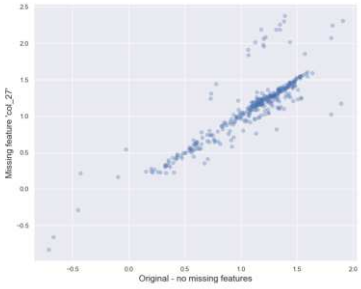
(xxv) col_24



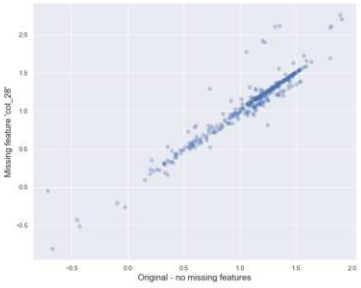
(xxvi) col_25



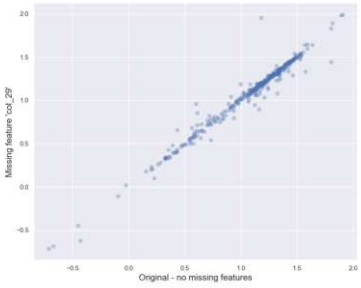
(xxvii) col_26



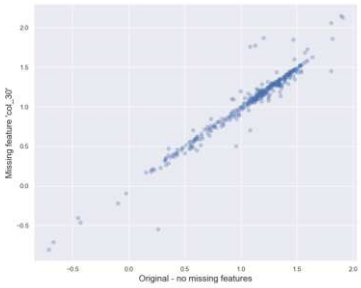
(xxviii) col_27



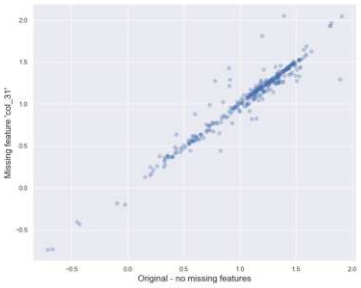
(xxix) col_28



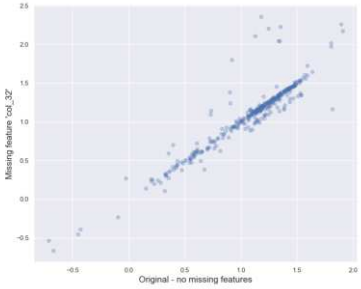
(xxx) col_29



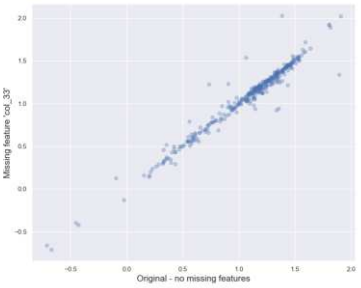
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

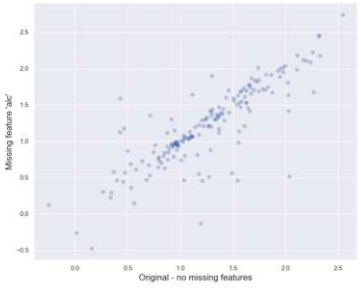


(xxxiv) col_33

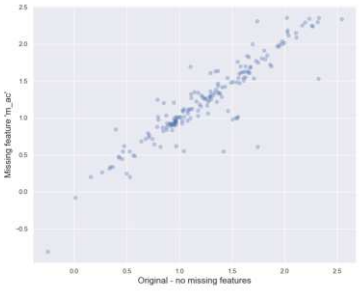
Source: Author's own elaboration

Figure 51 – Discrimination dispersion for the Wine dataset - Iterative Imputation

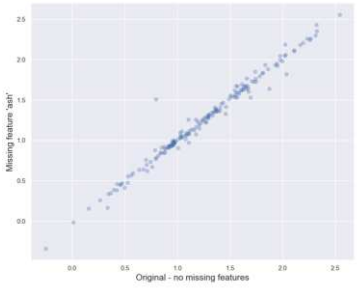
(i) alc



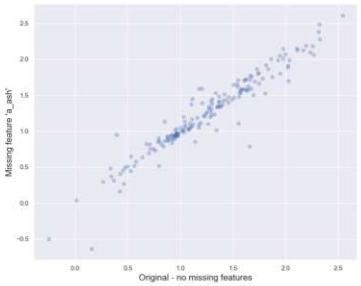
(ii) m_ac



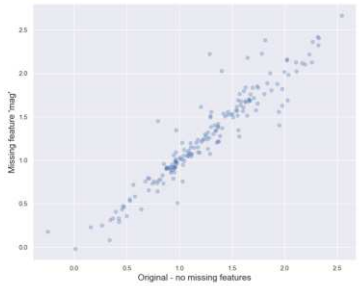
(iii) ash



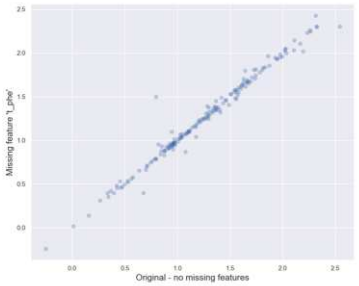
(iv) a_ash



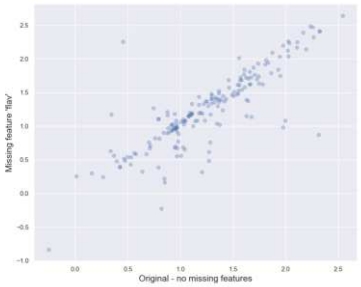
(v) mag



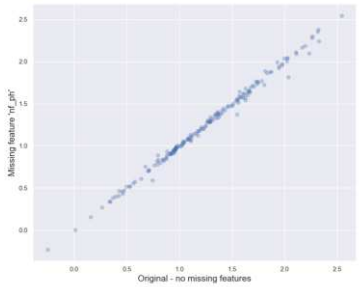
(vi) t_phe



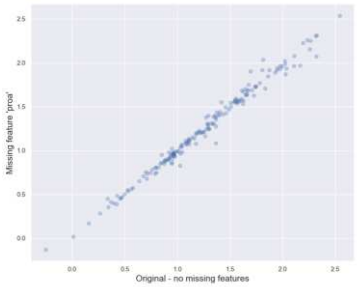
(vii) flav



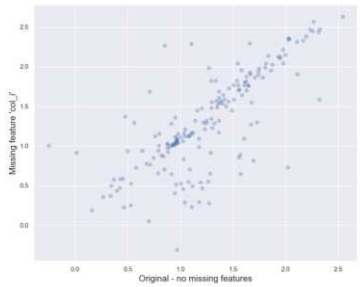
(viii) nf_ph



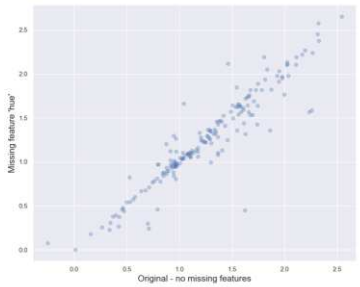
(ix) proa



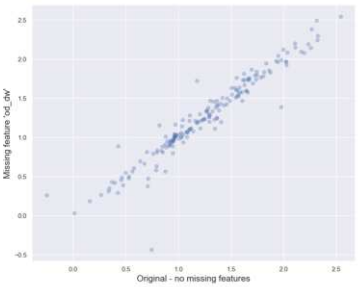
(x) col_i

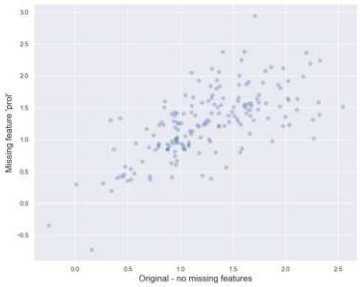


(xi) hue



(xii) od_dw



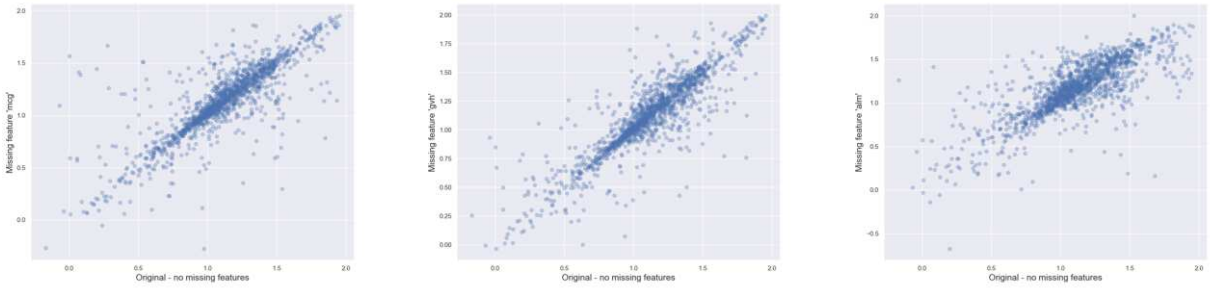


(xiii) prol

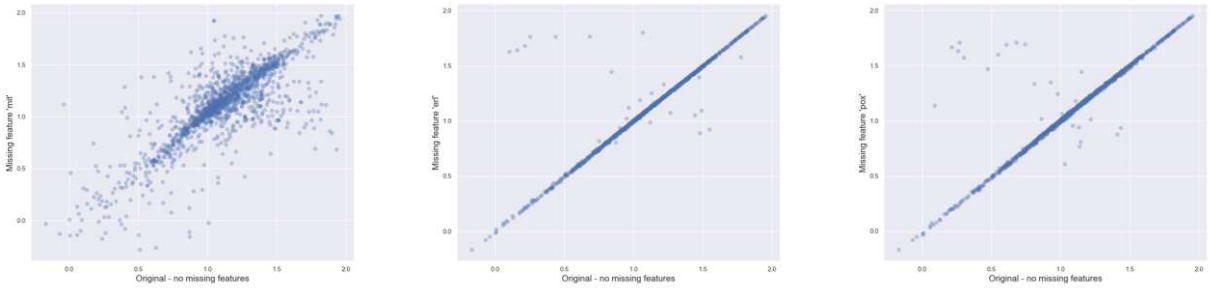
Source: Author's own elaboration

Figure 51 – Discrimination dispersion for the Yeast dataset - Iterative Imputation

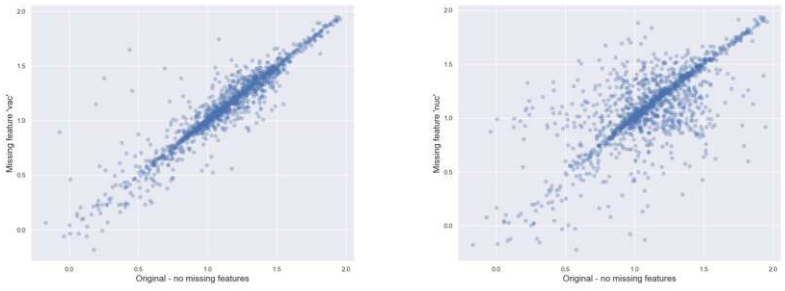
(i) mcg (ii) gvh (iii) alm



(iv) mit (v) erl (vi) pox



(vii) vac (viii) nuc



Source: Author's own elaboration

APPENDIX D – MEDIAN RANKING

D.1 SIMPLE IMPUTATION

Table 31 shows the results for the Contraceptive dataset, Table 32 for the Pima dataset, Table 33 for the Wheat dataset, Table 34 for the Haberman dataset, Table 35 for the Cleveland dataset, Table 36 for the Ionosphere dataset, Table 37 for the Wine dataset, and finally, Table 38 shows the results for the Yeast dataset.

Table 31 – Mean feature ranking for Contraceptive with Simple Imputation

	RFE_LREG	Beta-IRT_ab	RFECV_SVC	RFECV_LREG	ExtraTrees	Ridge	Beta-IR
w_age	9.0	3.0	9.0	9.0	1.0	10.0	1
w_edu	3.0	2.0	3.0	3.0	4.0	1.0	3
h_edu	8.0	6.0	8.0	8.0	5.0	11.0	9
n_chd	5.0	1.0	5.0	5.0	2.0	5.0	2
w_rel	2.0	10.0	2.0	2.0	9.0	3.0	7
w_wrk	10.0	9.0	10.0	10.0	6.0	8.0	10
stl_i	6.0	4.0	6.0	6.0	3.0	6.0	6
m_exp	2.0	7.0	2.0	2.0	10.0	2.0	11
h_occ_2	4.0	7.0	4.0	4.0	7.0	7.0	5
h_occ_3	9.0	5.0	9.0	9.0	8.0	9.0	4
h_occ_4	7.0	10.0	7.0	7.0	11.0	3.0	8

Source: Author's own calculation

Table 32 – Mean feature ranking for Pima with Simple Imputation

	RFE_LREG	Correlation	LogReg	RFECV_LREG	RandomForest	Beta-IRT_ab	Ridge
n_preg	2.0	4.0	2.0	2.0	6.0	4.0	2.0
pl_glu	4.0	1.0	4.0	4.0	1.0	1.0	4.0
di_BP	5.0	8.0	5.0	5.0	5.0	6.0	6.0
tsft	7.0	7.0	7.0	7.0	8.0	6.0	7.0
2h_si	8.0	6.0	8.0	8.0	7.0	7.0	8.0
b_mass	3.0	2.0	3.0	3.0	2.0	2.0	3.0
di_pf	1.0	5.0	1.0	1.0	4.0	5.0	1.0
age	6.0	3.0	6.0	6.0	3.0	3.0	5.0

Source: Author's own calculation

Table 33 – Mean feature ranking for wheat with Simple Imputation

	RFE_LREG	Beta-IRT_diff	Ridge	RFECV_SVC	LogReg	Correlation	ExtraTrees	Be
area	5.0	1.0	5.0	5.0	5.0	4.0	3.0	
per	4.0	2.0	3.0	4.0	4.0	5.0	2.0	
comp	6.0	5.0	7.0	6.0	6.0	2.0	7.0	
len_k	2.0	6.0	2.0	2.0	2.0	6.0	5.0	
wid_k	5.0	4.0	4.0	5.0	5.0	3.0	3.0	
asy_c	3.0	7.0	6.0	3.0	3.0	1.0	6.0	
len_kg	1.0	3.0	1.0	1.0	1.0	7.0	3.0	

Source: Author's own calculation

Table 34 – Mean feature ranking for Haberman with Simple Imputation

	ExtraTrees	Beta-IRT_diff	RFECV_SVC	RFE_LREG	LogReg	RandomForest	RFE_SV
age	2.0	2.0	2.0	2.0	2.0	1.0	2.0
y_op	3.0	3.0	3.0	3.0	3.0	3.0	3.0
pand	1.0	1.0	1.0	1.0	1.0	2.0	1.0

Source: Author's own calculation

Table 35 – Mean feature ranking for Cleveland with Simple Imputation

	RFE_SVC	Beta-IRT_ab	ExtraTrees	Beta-IRT_diff	Correlation	RFECV_LREG
age	13.0	8.0	4.0	15.0	10.0	12.0
sex	6.0	8.0	12.0	1.0	10.0	5.0
trestbps	14.0	11.0	6.0	9.0	13.0	14.0
chol	16.0	14.0	4.0	3.0	16.0	16.0
fbs	6.0	6.0	13.0	4.0	17.0	15.0
thalach	15.0	7.0	2.0	13.0	5.0	10.0
exang	4.0	13.0	9.0	10.0	6.0	6.0
o_peak	12.0	4.0	5.0	2.0	3.0	3.0
ca	3.0	2.0	1.0	14.0	1.0	2.0
thal	10.0	10.0	6.0	17.0	2.0	1.0
cp_2.0	5.0	8.0	16.0	6.0	10.0	10.0
cp_3.0	9.0	11.0	14.0	5.0	8.0	7.0
cp_4.0	1.0	4.0	8.0	11.0	4.0	4.0
restecg_1.0	17.0	6.0	17.0	8.0	15.0	17.0
restecg_2.0	8.0	9.0	11.0	12.0	12.0	10.0
slope_2.0	3.0	12.0	10.0	16.0	7.0	8.0
slope_3.0	9.0	8.0	15.0	7.0	14.0	14.0

Source: Author's own calculation

Table 36 – Mean feature ranking for Ionosphere with Simple Imputation

	Correlation	LogReg	Ridge	RFECV_SVC	RandomForest	ExtraTrees	Beta-IRT_ab
col_0	3.0	1.0	1.0	1.0	20.0	3.0	27.0
col_1	34.0	34.0	34.0	34.0	34.0	34.0	33.0
col_2	1.0	4.0	7.0	4.0	3.0	3.0	2.0
col_3	22.0	18.0	14.0	18.0	8.0	6.0	30.0
col_4	2.0	4.0	4.0	4.0	1.0	1.0	1.0
col_5	19.0	8.0	12.0	8.0	8.0	7.0	8.0
col_6	4.0	14.0	16.0	14.0	2.0	4.0	7.0
col_7	13.0	6.0	6.0	6.0	6.0	7.0	13.0
col_8	6.0	11.0	12.0	11.0	24.0	21.0	3.0
col_9	22.0	27.0	22.0	27.0	21.0	16.0	25.0
col_10	16.0	17.0	24.0	17.0	32.0	30.0	21.0
col_11	17.0	28.0	26.0	28.0	17.0	11.0	10.0
col_12	16.0	25.0	27.0	25.0	29.0	20.0	13.0
col_13	12.0	18.0	25.0	18.0	8.0	7.0	19.0
col_14	11.0	17.0	28.0	17.0	20.0	14.0	7.0
col_15	20.0	23.0	24.0	23.0	11.0	13.0	27.0
col_16	26.0	31.0	27.0	31.0	29.0	23.0	15.0
col_17	23.0	17.0	23.0	17.0	14.0	20.0	29.0
col_18	22.0	20.0	17.0	20.0	29.0	29.0	16.0
col_19	32.0	28.0	19.0	28.0	18.0	22.0	29.0
col_20	10.0	30.0	27.0	30.0	24.0	19.0	5.0
col_21	20.0	2.0	3.0	2.0	12.0	26.0	18.0
col_22	12.0	13.0	8.0	13.0	28.0	21.0	24.0
col_23	30.0	21.0	23.0	21.0	14.0	14.0	23.0
col_24	14.0	18.0	22.0	18.0	31.0	28.0	11.0
col_25	31.0	16.0	14.0	16.0	24.0	31.0	24.0
col_26	21.0	5.0	5.0	5.0	3.0	18.0	4.0
col_27	29.0	26.0	21.0	26.0	10.0	16.0	19.0
col_28	8.0	21.0	18.0	21.0	15.0	19.0	15.0
col_29	31.0	10.0	16.0	10.0	26.0	31.0	26.0
col_30	5.0	12.0	12.0	12.0	10.0	12.0	9.0
col_31	28.0	23.0	30.0	23.0	18.0	30.0	25.0
col_32	7.0	26.0	23.0	26.0	11.0	24.0	20.0
col_33	26.0	8.0	7.0	8.0	22.0	24.0	26.0

Source: Author's own calculation

Table 37 – Mean feature ranking for Wine with Simple Imputation

	ExtraTrees	Correlation	RFE_LREG	Beta-IRT_diff	Ridge	RFECV_SVC	LogReg	RF
alc	4.0	10.0	8.0	4.0	3.0	8.0	8.0	
m_ac	8.0	9.0	3.0	8.0	10.0	3.0	3.0	
ash	13.0	13.0	7.0	11.0	1.0	7.0	7.0	
a_ash	9.0	6.0	3.0	10.0	8.0	3.0	3.0	
mag	10.0	12.0	11.0	9.0	13.0	11.0	11.0	
t_phe	7.0	3.0	5.0	7.0	6.0	5.0	5.0	
flav	3.0	1.0	1.0	1.0	3.0	1.0	1.0	
nf_ph	12.0	7.0	11.0	13.0	7.0	11.0	11.0	
proa	11.0	8.0	8.0	12.0	7.0	8.0	8.0	
col_i	4.0	11.0	9.0	3.0	10.0	9.0	9.0	
hue	6.0	5.0	12.0	6.0	6.0	12.0	12.0	
od_dw	4.0	2.0	2.0	5.0	3.0	2.0	2.0	
prol	1.0	4.0	12.0	2.0	12.0	12.0	12.0	

Source: Author's own calculation

Table 38 – Mean feature ranking for Yeast with Simple Imputation

	ExtraTrees	Ridge	RFE_SVC	RFECV_LREG	RFE_LREG	Correlation	RFECV_SVC	R
mcg	3.0	2.0	2.0	1.0	1.0	3.0	1.0	
gvh	4.0	3.0	2.0	2.0	2.0	7.0	2.0	
alm	1.0	1.0	4.0	3.0	3.0	2.0	3.0	
mit	2.0	8.0	6.0	4.0	4.0	1.0	4.0	
erl	8.0	4.0	8.0	5.0	5.0	5.0	5.0	
pox	7.0	7.0	1.0	8.0	8.0	4.0	8.0	
vac	5.0	6.0	5.0	7.0	7.0	7.0	7.0	
nuc	6.0	5.0	7.0	6.0	6.0	6.0	6.0	

Source: Author's own calculation

D.2 KNN IMPUTATION

Table 39 shows the results for the Contraceptive dataset, Table 40 for the Pima dataset, Table 41 for the Wheat dataset, Table 42 for the Haberman dataset, Table 43 for the Cleveland dataset, Table 44 for the Ionosphere dataset, Table 45 for the Wine dataset, and finally, Table 46 shows the results for the Yeast dataset.

Table 39 – Mean feature ranking for Contraceptive with KNN Imputation

	RFECV_LREG	Correlation	RFECV_SVC	ExtraTrees	RandomForest	LogReg	RFE_
w_age	9.0	1.0	9.0	1.0	1.0	9.0	10
w_edu	3.0	2.0	3.0	4.0	4.0	3.0	1
h_edu	8.0	4.0	8.0	5.0	5.0	8.0	9
n_chd	5.0	6.0	5.0	2.0	2.0	5.0	5
w_rel	2.0	10.0	2.0	9.0	9.0	2.0	2
w_wrk	10.0	7.0	10.0	6.0	6.0	10.0	9
stl_i	6.0	5.0	6.0	3.0	3.0	6.0	6
m_exp	2.0	3.0	2.0	10.0	10.0	2.0	6
h_occ_2	4.0	9.0	4.0	7.0	7.0	4.0	3
h_occ_3	9.0	8.0	9.0	8.0	8.0	9.0	7
h_occ_4	7.0	11.0	7.0	11.0	11.0	7.0	4

Source: Author's own calculation

Table 40 – Mean feature ranking for Pima with KNN Imputation

	Beta-IRT_diff	RFE_LREG	Beta-IRT_ab	Ridge	Correlation	RFECV_SVC	RFECV_
n_preg	8.0	2.0	6.0	2.0	4.0	2.0	2.0
pl_glu	2.0	4.0	6.0	4.0	1.0	4.0	4.0
di_BP	6.0	5.0	7.0	6.0	8.0	5.0	5.0
tsft	4.0	7.0	5.0	7.0	7.0	7.0	7.0
2h_si	3.0	8.0	6.0	8.0	6.0	8.0	8.0
b_mass	1.0	3.0	1.0	3.0	2.0	3.0	3.0
di_pf	7.0	1.0	2.0	1.0	5.0	1.0	1.0
age	5.0	6.0	3.0	5.0	3.0	6.0	6.0

Source: Author's own calculation

Table 41 – Mean feature ranking for wheat with KNN Imputation

	RFECV_SVC	LogReg	RFECV_LREG	Ridge	RandomForest	Correlation	RFE_LREG
area	5.0	5.0	5.0	5.0	2.0	4.0	5.0
per	4.0	4.0	4.0	3.0	3.0	5.0	4.0
comp	6.0	6.0	6.0	7.0	7.0	2.0	6.0
len_k	2.0	2.0	2.0	2.0	4.0	6.0	2.0
wid_k	5.0	5.0	5.0	4.0	5.0	3.0	5.0
asy_c	3.0	3.0	3.0	6.0	6.0	1.0	3.0
len_kg	1.0	1.0	1.0	1.0	2.0	7.0	1.0

Source: Author's own calculation

Table 42 – Mean feature ranking for Haberman with KNN Imputation

	ExtraTrees	RandomForest	Correlation	RFE_SVC	RFECV_SVC	RFE_LREG	RFECV_
age	1.0	1.0	2.0	2.0	2.0	2.0	2.0
y_op	3.0	3.0	3.0	3.0	3.0	3.0	3.0
pand	2.0	2.0	1.0	1.0	1.0	1.0	1.0

Source: Author's own calculation

Table 43 – Mean feature ranking for Cleveland with KNN Imputation

	Beta-IRT_ab	Ridge	Correlation	RFECV_SVC	ExtraTrees	LogReg	RFECV_LRE
age	9.0	16.0	10.0	12.0	5.0	12.0	12.0
sex	10.0	2.0	10.0	5.0	12.0	5.0	5.0
trestbps	13.0	15.0	13.0	14.0	7.0	14.0	14.0
chol	11.0	17.0	16.0	16.0	5.0	16.0	16.0
fbs	8.0	10.0	17.0	15.0	13.0	15.0	15.0
thalach	12.0	15.0	5.0	10.0	2.0	10.0	10.0
exang	6.0	5.0	6.0	6.0	9.0	6.0	6.0
o_peak	3.0	11.0	3.0	3.0	3.0	3.0	3.0
ca	2.0	3.0	1.0	2.0	2.0	2.0	2.0
thal	8.0	10.0	2.0	1.0	4.0	1.0	1.0
cp_2.0	9.0	7.0	10.0	10.0	16.0	10.0	10.0
cp_3.0	8.0	9.0	8.0	7.0	14.0	7.0	7.0
cp_4.0	3.0	1.0	4.0	4.0	8.0	4.0	4.0
restecg_1.0	7.0	8.0	15.0	17.0	17.0	17.0	17.0
restecg_2.0	12.0	10.0	12.0	10.0	11.0	10.0	10.0
slope_2.0	12.0	4.0	7.0	8.0	10.0	8.0	8.0
slope_3.0	10.0	11.0	14.0	14.0	15.0	14.0	14.0

Source: Author's own calculation

Table 44 – Mean feature ranking for Ionosphere with KNN Imputation

	Beta-IRT_ab	RandomForest	RFECV_LREG	LogReg	Beta-IRT_diff	ExtraTrees	Ridg
col_0	9.0	9.0	1.0	1.0	3.0	3.0	1.0
col_1	26.0	34.0	34.0	34.0	22.0	34.0	34.0
col_2	8.0	3.0	4.0	4.0	34.0	2.0	7.0
col_3	20.0	8.0	18.0	18.0	15.0	8.0	14.0
col_4	2.0	1.0	4.0	4.0	4.0	1.0	4.0
col_5	8.0	10.0	8.0	8.0	6.0	4.0	12.0
col_6	7.0	3.0	14.0	14.0	5.0	5.0	16.0
col_7	12.0	6.0	6.0	6.0	7.0	5.0	6.0
col_8	31.0	19.0	11.0	11.0	33.0	16.0	12.0
col_9	19.0	18.0	27.0	27.0	23.0	14.0	22.0
col_10	16.0	30.0	17.0	17.0	28.0	31.0	24.0
col_11	23.0	14.0	28.0	28.0	9.0	12.0	26.0
col_12	15.0	26.0	25.0	25.0	14.0	25.0	27.0
col_13	11.0	8.0	18.0	18.0	16.0	7.0	25.0
col_14	20.0	23.0	17.0	17.0	32.0	21.0	28.0
col_15	17.0	14.0	23.0	23.0	31.0	11.0	24.0
col_16	26.0	23.0	31.0	31.0	29.0	27.0	27.0
col_17	25.0	12.0	17.0	17.0	20.0	19.0	23.0
col_18	18.0	30.0	20.0	20.0	17.0	28.0	17.0
col_19	23.0	24.0	28.0	28.0	19.0	23.0	19.0
col_20	17.0	24.0	30.0	30.0	11.0	10.0	27.0
col_21	21.0	17.0	2.0	2.0	18.0	23.0	3.0
col_22	29.0	27.0	13.0	13.0	26.0	18.0	8.0
col_23	11.0	15.0	21.0	21.0	21.0	19.0	23.0
col_24	15.0	31.0	18.0	18.0	10.0	21.0	22.0
col_25	11.0	22.0	16.0	16.0	12.0	32.0	14.0
col_26	1.0	3.0	5.0	5.0	1.0	17.0	5.0
col_27	5.0	12.0	26.0	26.0	2.0	16.0	21.0
col_28	16.0	20.0	21.0	21.0	27.0	15.0	18.0
col_29	22.0	28.0	10.0	10.0	30.0	32.0	16.0
col_30	23.0	16.0	12.0	12.0	8.0	14.0	12.0
col_31	15.0	21.0	23.0	23.0	13.0	28.0	30.0
col_32	24.0	15.0	26.0	26.0	25.0	19.0	23.0
col_33	21.0	20.0	8.0	8.0	24.0	28.0	7.0

Source: Author's own calculation

Table 45 – Mean feature ranking for Wine with KNN Imputation

	Beta-IRT_diff	RandomForest	ExtraTrees	Correlation	RFE_LREG	Beta-IRT_ab	RFE
alc	7.0	4.0	4.0	10.0	8.0	6.0	
m_ac	5.0	9.0	8.0	9.0	3.0	7.0	
ash	8.0	12.0	13.0	13.0	7.0	9.0	
a_ash	12.0	9.0	10.0	6.0	3.0	13.0	
mag	13.0	8.0	10.0	12.0	11.0	11.0	
t_phe	6.0	7.0	7.0	3.0	5.0	5.0	
flav	1.0	3.0	2.0	1.0	1.0	1.0	
nf_ph	11.0	13.0	11.0	7.0	11.0	8.0	
proa	9.0	11.0	12.0	8.0	8.0	8.0	
col_i	2.0	2.0	4.0	11.0	9.0	2.0	
hue	3.0	6.0	6.0	5.0	12.0	4.0	
od_dw	4.0	5.0	3.0	2.0	2.0	3.0	
prol	10.0	1.0	1.0	4.0	12.0	12.0	

Source: Author's own calculation

Table 46 – Mean feature ranking for Yeast with KNN Imputation

	Ridge	RFECV_SVC	RFE_SVC	RFECV_LREG	LogReg	Correlation	RFE_LREG	Beta
mcg	2.0	1.0	2.0	1.0	1.0	3.0	1.0	
gvh	3.0	2.0	2.0	2.0	2.0	7.0	2.0	
alm	1.0	3.0	4.0	3.0	3.0	2.0	3.0	
mit	8.0	4.0	6.0	4.0	4.0	1.0	4.0	
erl	4.0	5.0	8.0	5.0	5.0	5.0	5.0	
pox	7.0	8.0	1.0	8.0	8.0	4.0	8.0	
vac	6.0	7.0	5.0	7.0	7.0	7.0	7.0	
nuc	5.0	6.0	7.0	6.0	6.0	6.0	6.0	

Source: Author's own calculation

D.3 ITERATIVE IMPUTATION

Table 47 shows the results for the Contraceptive dataset, Table 48 for the Pima dataset, Table 49 for the Wheat dataset, Table 50 for the Haberman dataset, Table 51 for the Cleveland dataset, Table 52 for the Ionosphere dataset, Table 53 for the Wine dataset, and finally, Table 54 shows the results for the Yeast dataset.

Table 47 – Mean feature ranking for Contraceptive with Iterative Imputation

	RFE_SVC	Beta-IRT_ab	RFECV_LREG	RandomForest	LogReg	Ridge	RFE_LREG
w_age	10.0	1.0	9.0	1.0	9.0	10.0	9.0
w_edu	1.0	3.0	3.0	4.0	3.0	1.0	3.0
h_edu	9.0	6.0	8.0	5.0	8.0	11.0	8.0
n_chd	5.0	2.0	5.0	2.0	5.0	5.0	5.0
w_rel	2.0	8.0	2.0	9.0	2.0	3.0	2.0
w_wrk	9.0	9.0	10.0	6.0	10.0	8.0	10.0
stl_i	6.0	7.0	6.0	3.0	6.0	6.0	6.0
m_exp	6.0	9.0	2.0	10.0	2.0	2.0	2.0
h_occ_2	3.0	9.0	4.0	8.0	4.0	7.0	4.0
h_occ_3	7.0	6.0	9.0	7.0	9.0	9.0	9.0
h_occ_4	4.0	8.0	7.0	11.0	7.0	3.0	7.0

Source: Author's own calculation

Table 48 – Mean feature ranking for Pima with Iterative Imputation

	LogReg	Beta-IRT_diff	RFE_SVC	RandomForest	RFECV_SVC	Correlation	ExtraTr
n_preg	2.0	7.0	2.0	6.0	2.0	4.0	5.0
pl_glu	4.0	1.0	4.0	1.0	4.0	1.0	1.0
di_BP	5.0	4.0	5.0	5.0	5.0	8.0	6.0
tsft	7.0	6.0	7.0	8.0	7.0	7.0	7.0
2h_si	8.0	3.0	8.0	7.0	8.0	6.0	8.0
b_mass	3.0	2.0	3.0	2.0	3.0	2.0	3.0
di_pf	1.0	5.0	1.0	4.0	1.0	5.0	4.0
age	6.0	8.0	6.0	3.0	6.0	3.0	2.0

Source: Author's own calculation

Table 49 – Mean feature ranking for wheat with Iterative Imputation

	Correlation	Beta-IRT_diff	RFECV_LREG	Ridge	LogReg	RFE_LREG	ExtraTrees	E
area	4.0	1.0	5.0	5.0	5.0	5.0	1.0	
per	5.0	2.0	4.0	3.0	4.0	4.0	2.0	
comp	2.0	4.0	6.0	7.0	6.0	6.0	7.0	
len_k	6.0	7.0	2.0	2.0	2.0	2.0	5.0	
wid_k	3.0	6.0	5.0	4.0	5.0	5.0	3.0	
asy_c	1.0	5.0	3.0	6.0	3.0	3.0	6.0	
len_kg	7.0	3.0	1.0	1.0	1.0	1.0	3.0	

Source: Author's own calculation

Table 50 – Mean feature ranking for Haberman with Iterative Imputation

	ExtraTrees	RFE_LREG	Correlation	RandomForest	RFECV_SVC	Beta-IRT_diff	Ridge
age	1.0	2.0	2.0	1.0	2.0	2.0	2.0
y_op	3.0	3.0	3.0	3.0	3.0	3.0	3.0
pand	2.0	1.0	1.0	2.0	1.0	1.0	1.0

Source: Author's own calculation

Table 51 – Mean feature ranking for Cleveland with Iterative Imputation

	Beta-IRT_diff	ExtraTrees	LogReg	RFE_LREG	RFECV_SVC	RandomForest	Ri
age	16.0	4.0	12.0	12.0	12.0	4.0	1
sex	6.0	12.0	5.0	5.0	5.0	11.0	2
trestbps	9.0	7.0	14.0	14.0	14.0	5.0	1
chol	11.0	6.0	16.0	16.0	16.0	3.0	1
fbs	7.0	13.0	15.0	15.0	15.0	14.0	1
thalach	8.0	3.0	10.0	10.0	10.0	1.0	1
exang	4.0	9.0	6.0	6.0	6.0	9.0	5
o_peak	5.0	4.0	3.0	3.0	3.0	2.0	1
ca	2.0	1.0	2.0	2.0	2.0	6.0	3
thal	1.0	4.0	1.0	1.0	1.0	7.0	1
cp_2.0	14.0	16.0	10.0	10.0	10.0	15.0	7
cp_3.0	17.0	14.0	7.0	7.0	7.0	13.0	9
cp_4.0	3.0	8.0	4.0	4.0	4.0	8.0	1
restecg_1.0	12.0	17.0	17.0	17.0	17.0	17.0	8
restecg_2.0	13.0	11.0	10.0	10.0	10.0	12.0	1
slope_2.0	15.0	10.0	8.0	8.0	8.0	10.0	4
slope_3.0	10.0	15.0	14.0	14.0	14.0	16.0	1

Source: Author's own calculation

Table 52 – Mean feature ranking for Ionosphere with Iterative Imputation

	RFE_SVC	Correlation	LogReg	Beta-IRT_ab	ExtraTrees	RFECV_LREG	RFECV_SV
col_0	1.0	3.0	1.0	26.0	2.0	1.0	1.0
col_1	34.0	34.0	34.0	29.0	34.0	34.0	34.0
col_2	9.0	1.0	4.0	5.0	3.0	4.0	4.0
col_3	28.0	22.0	18.0	25.0	7.0	18.0	18.0
col_4	6.0	2.0	4.0	2.0	1.0	4.0	4.0
col_5	9.0	19.0	8.0	4.0	8.0	8.0	8.0
col_6	12.0	4.0	14.0	19.0	5.0	14.0	14.0
col_7	4.0	13.0	6.0	4.0	4.0	6.0	6.0
col_8	19.0	6.0	11.0	29.0	16.0	11.0	11.0
col_9	29.0	22.0	27.0	24.0	10.0	27.0	27.0
col_10	17.0	16.0	17.0	17.0	30.0	17.0	17.0
col_11	18.0	17.0	28.0	14.0	17.0	28.0	28.0
col_12	19.0	16.0	25.0	23.0	18.0	25.0	25.0
col_13	18.0	12.0	18.0	18.0	8.0	18.0	18.0
col_14	12.0	11.0	17.0	22.0	16.0	17.0	17.0
col_15	25.0	20.0	23.0	17.0	11.0	23.0	23.0
col_16	27.0	26.0	31.0	27.0	26.0	31.0	31.0
col_17	24.0	23.0	17.0	13.0	18.0	17.0	17.0
col_18	21.0	22.0	20.0	17.0	27.0	20.0	20.0
col_19	17.0	32.0	28.0	24.0	23.0	28.0	28.0
col_20	29.0	10.0	30.0	11.0	20.0	30.0	30.0
col_21	2.0	20.0	2.0	8.0	24.0	2.0	2.0
col_22	19.0	12.0	13.0	17.0	21.0	13.0	13.0
col_23	23.0	30.0	21.0	9.0	15.0	21.0	21.0
col_24	22.0	14.0	18.0	20.0	25.0	18.0	18.0
col_25	20.0	31.0	16.0	13.0	30.0	16.0	16.0
col_26	5.0	21.0	5.0	1.0	22.0	5.0	5.0
col_27	28.0	29.0	26.0	7.0	13.0	26.0	26.0
col_28	15.0	8.0	21.0	21.0	23.0	21.0	21.0
col_29	8.0	31.0	10.0	21.0	32.0	10.0	10.0
col_30	12.0	5.0	12.0	29.0	13.0	12.0	12.0
col_31	24.0	28.0	23.0	18.0	29.0	23.0	23.0
col_32	24.0	7.0	26.0	18.0	18.0	26.0	26.0
col_33	7.0	26.0	8.0	16.0	25.0	8.0	8.0

Source: Author's own calculation

Table 53 – Mean feature ranking for Wine with Iterative Imputation

	RFECV_LREG	RFE_SVC	LogReg	ExtraTrees	Correlation	Ridge	Beta-IRT_diff	RF
alc	8.0	2.0	8.0	3.0	10.0	3.0	4.0	
m_ac	3.0	3.0	3.0	8.0	9.0	10.0	7.0	
ash	7.0	1.0	7.0	13.0	13.0	1.0	10.0	
a_ash	3.0	8.0	3.0	9.0	6.0	8.0	8.0	
mag	11.0	12.0	11.0	10.0	12.0	13.0	9.0	
t_phe	5.0	11.0	5.0	7.0	3.0	6.0	13.0	
flav	1.0	6.0	1.0	1.0	1.0	3.0	3.0	
nf_ph	11.0	7.0	11.0	12.0	7.0	7.0	11.0	
proa	8.0	10.0	8.0	11.0	8.0	7.0	12.0	
col_i	9.0	5.0	9.0	3.0	11.0	10.0	1.0	
hue	12.0	10.0	12.0	6.0	5.0	6.0	6.0	
od_dw	2.0	4.0	2.0	5.0	2.0	3.0	5.0	
prol	12.0	13.0	12.0	2.0	4.0	12.0	2.0	

Source: Author's own calculation

Table 54 – Mean feature ranking for Yeast with Iterative Imputation

	Beta-IRT_ab	RFECV_LREG	RFE_LREG	RFECV_SVC	Ridge	RandomForest	LogReg
mcg	3.0	1.0	1.0	1.0	2.0	3.0	1.0
gvh	5.0	2.0	2.0	2.0	3.0	4.0	2.0
alm	1.0	3.0	3.0	3.0	1.0	1.0	3.0
mit	2.0	4.0	4.0	4.0	8.0	2.0	4.0
erl	8.0	5.0	5.0	5.0	4.0	8.0	5.0
pox	6.0	8.0	8.0	8.0	7.0	7.0	8.0
vac	7.0	7.0	7.0	7.0	6.0	5.0	7.0
nuc	4.0	6.0	6.0	6.0	5.0	6.0	6.0

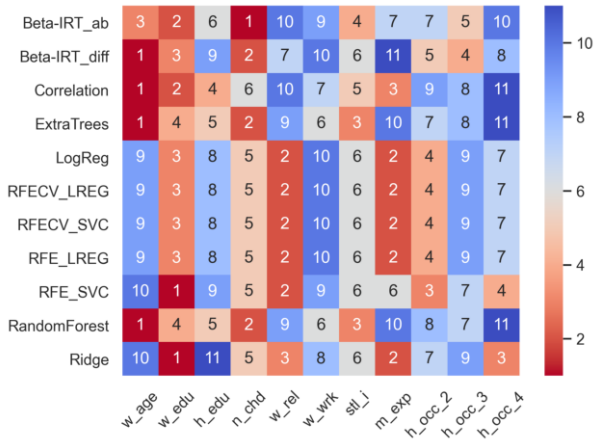
Source: Author's own calculation

APPENDIX E – MEDIAN FEATURE RANKING: HEATMAPS

E.1 SIMPLE IMPUTATION

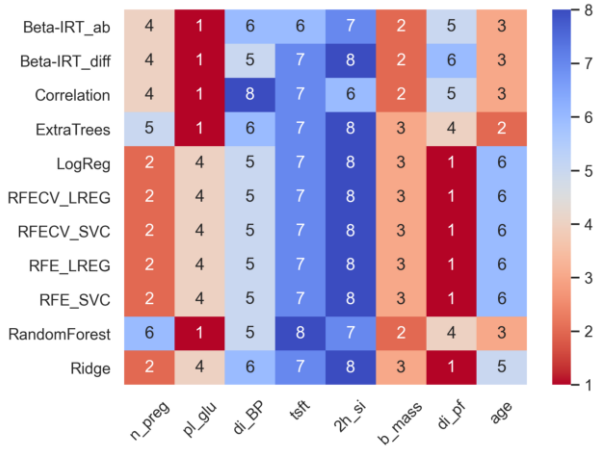
Figure 52 shows the results for the Contraceptive dataset, Figure 53 for the Pima dataset, Figure 54 for the Wheat dataset, Figure 55 for the Haberman dataset, Figure 56 for the Cleveland dataset, Figure 57 for the Ionosphere dataset, Figure 58 for the Wine dataset, and finally, Figure 59 shows the results for the Yeast dataset.

Figure 52 – Heatmap of median feature ranking for Contraceptive with Simple imputation



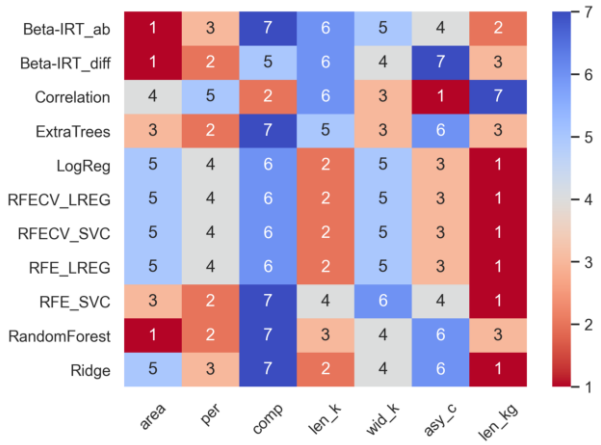
Source: Author's own elaboration

Figure 53 – Heatmap of median feature ranking for Pima with Simple imputation



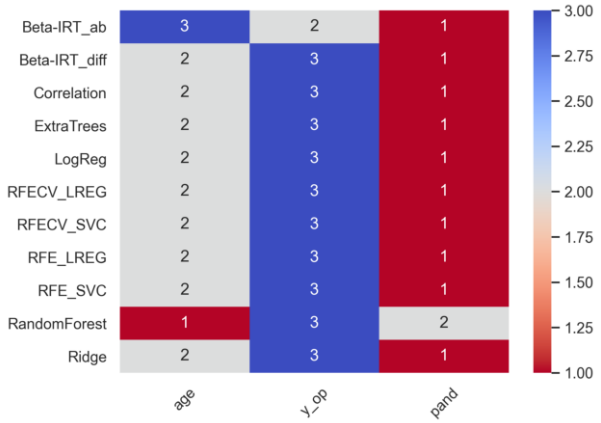
Source: Author's own elaboration

Figure 54 – Heatmap of median feature ranking for Wheat with Simple imputation



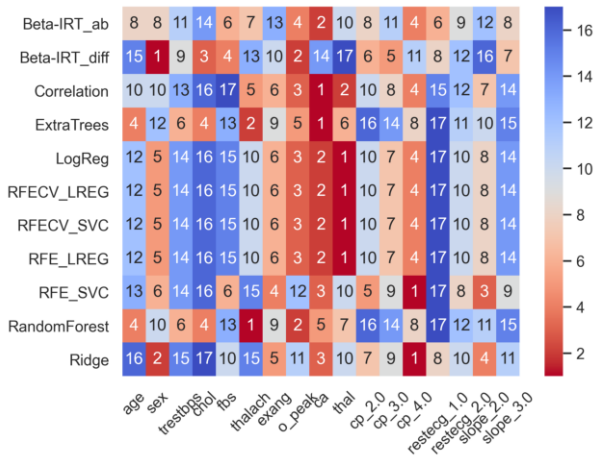
Source: Author's own elaboration

Figure 55 – Heatmap of median feature ranking for Haberman with Simple imputation



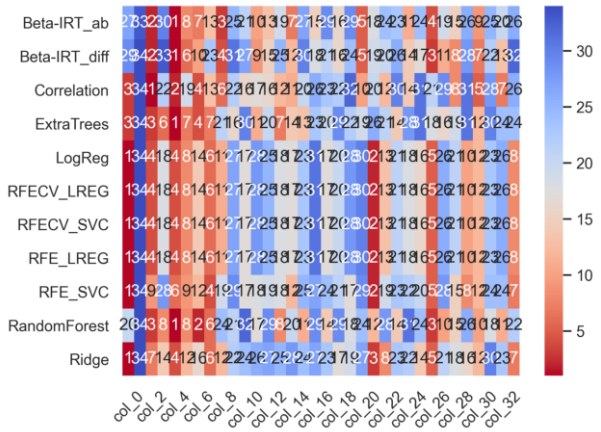
Source: Author's own elaboration

Figure 56 – Heatmap of median feature ranking for Cleveland with Simple imputation



Source: Author's own elaboration

Figure 57 – Heatmap of median feature ranking for Ionosphere with Simple imputation



Source: Author's own elaboration

Figure 58 – Heatmap of median feature ranking for Wine with Simple imputation

figures/dataset_wine/simple/rank_smpl_median_v02_annot.png

Source: Author's own elaboration

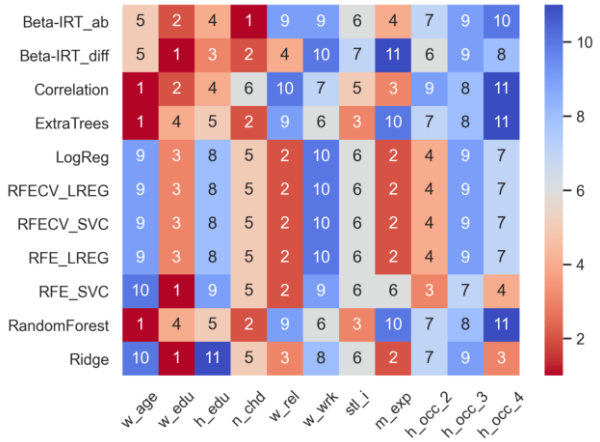
Figure 59 – Heatmap of median feature ranking for Yeast with Simple imputation

Source: Author's own elaboration

E.2 KNN IMPUTATION

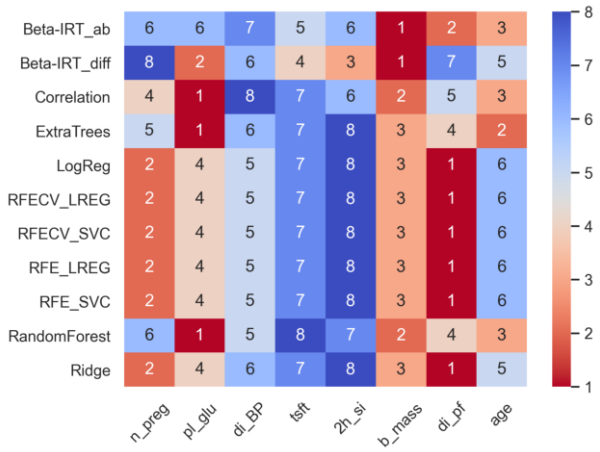
Figure 60 shows the results for the Contraceptive dataset, Figure 61 for the Pima dataset, Figure 62 for the Wheat dataset, Figure 63 for the Haberman dataset, Figure 64 for the Cleveland dataset, Figure 65 for the Ionosphere dataset, Figure 66 for the Wine dataset, and finally, Figure 67 shows the results for the Yeast dataset.

Figure 60 – Heatmap of median feature ranking for Contraceptive with Knn imputation



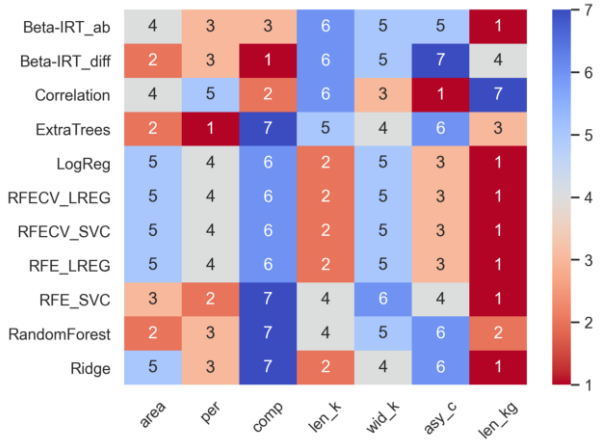
Source: Author's own elaboration

Figure 61 – Heatmap of median feature ranking for Pima with Knn imputation



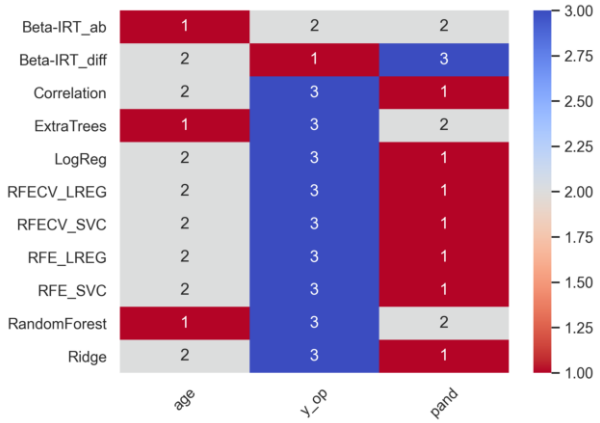
Source: Author's own elaboration

Figure 62 – Heatmap of median feature ranking for Wheat with Knn imputation



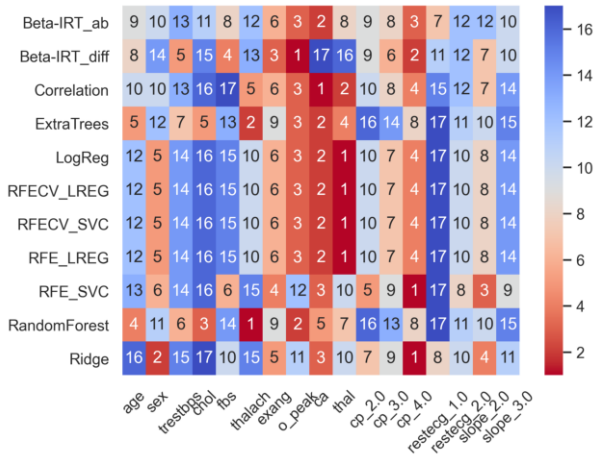
Source: Author's own elaboration

Figure 63 – Heatmap of median feature ranking for Haberman with Knn imputation



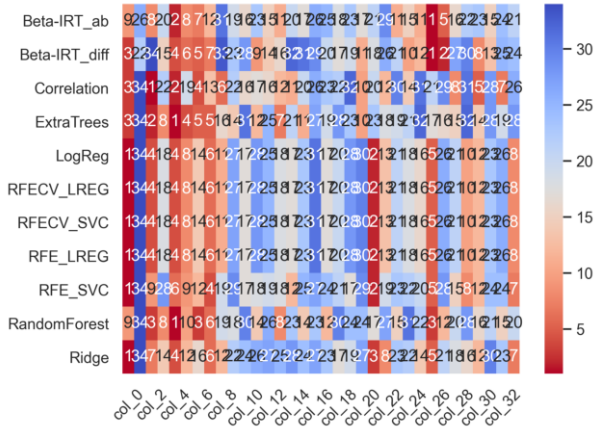
Source: Author's own elaboration

Figure 64 – Heatmap of median feature ranking for Cleveland with Knn imputation



Source: Author's own elaboration

Figure 65 – Heatmap of median feature ranking for Ionosphere with Knn imputation



Source: Author's own elaboration

Figure 66 – Heatmap of median feature ranking for Wine with Knn imputation

Source: Author's own elaboration

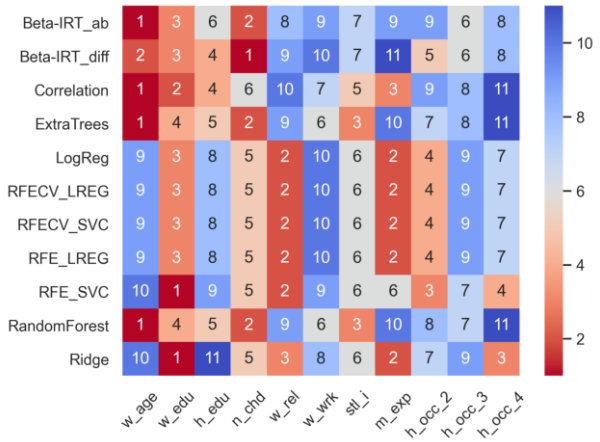
Figure 67 – Heatmap of median feature ranking for Yeast with Knn imputation

Source: Author's own elaboration

E.3 ITERATIVE IMPUTATION

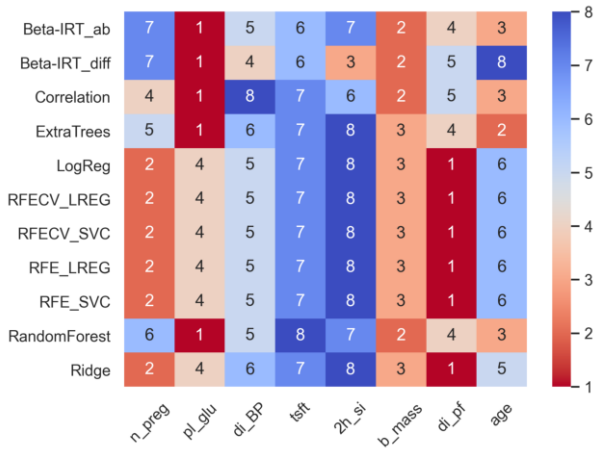
Figure 68 shows the results for the Contraceptive dataset, Figure 69 for the Pima dataset, Figure 70 for the Wheat dataset, Figure 71 for the Haberman dataset, Figure 72 for the Cleveland dataset, Figure 73 for the Ionosphere dataset, Figure 74 for the Wine dataset, and finally, Figure 75 shows the results for the Yeast dataset.

Figure 68 – Heatmap of median feature ranking for Contraceptive with Iterative imputation



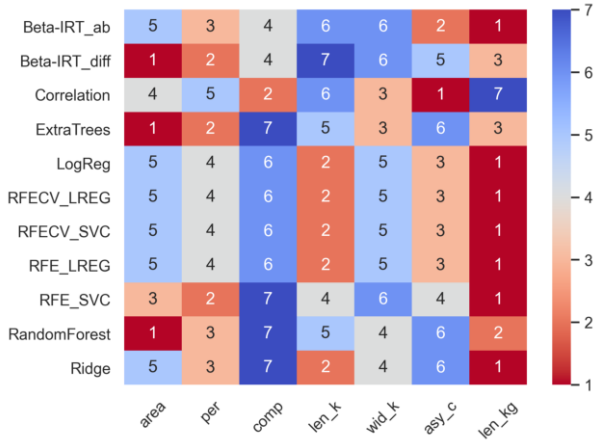
Source: Author's own elaboration

Figure 69 – Heatmap of median feature ranking for Pima with Iterative imputation



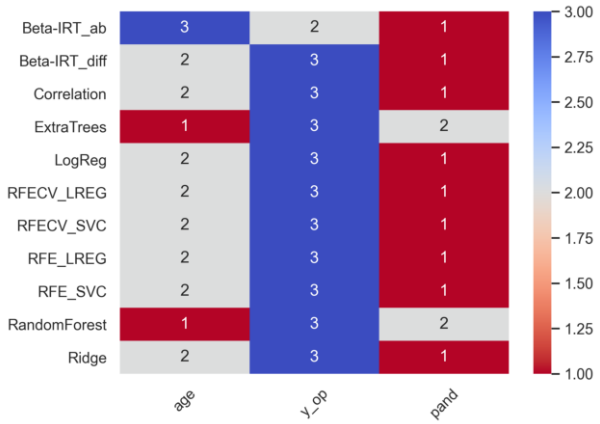
Source: Author's own elaboration

Figure 70 – Heatmap of median feature ranking for Wheat with Iterative imputation



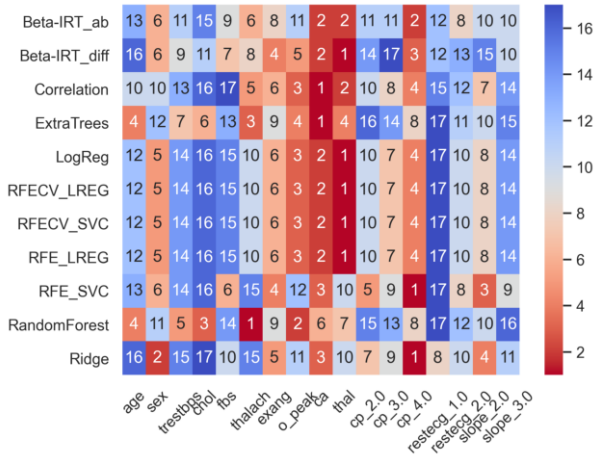
Source: Author's own elaboration

Figure 71 – Heatmap of median feature ranking for Haberman with Iterative imputation



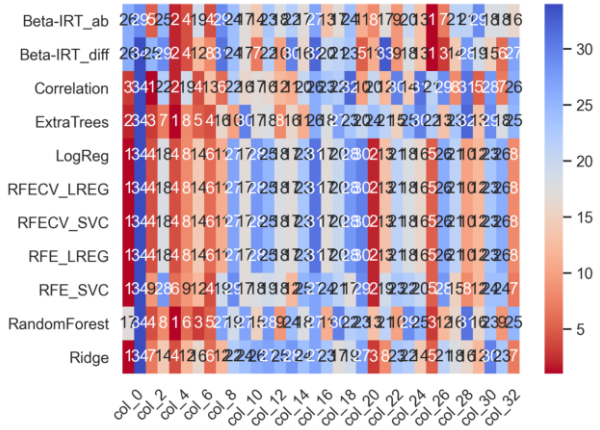
Source: Author's own elaboration

Figure 72 – Heatmap of median feature ranking for Cleveland with Iterative imputation



Source: Author's own elaboration

Figure 73 – Heatmap of median feature ranking for Ionosphere with Iterative imputation

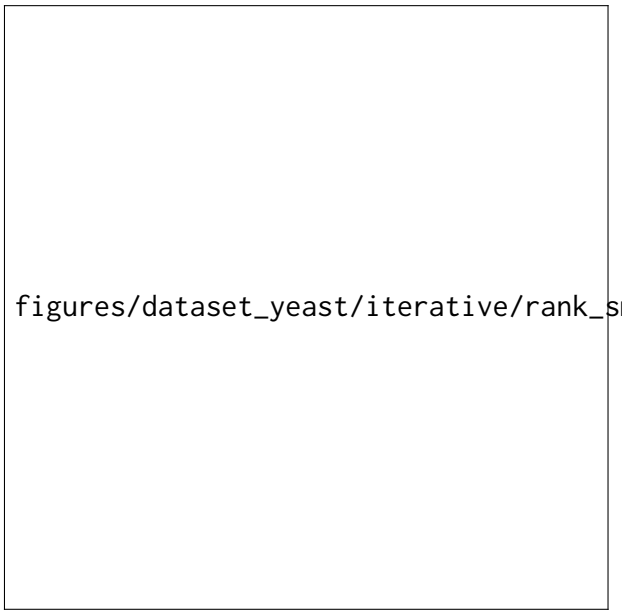


Source: Author's own elaboration

Figure 74 – Heatmap of median feature ranking for Wine with Iterative imputation

Source: Author's own elaboration

Figure 75 – Heatmap of median feature ranking for Yeast with Iterative imputation



Source: Author's own elaboration

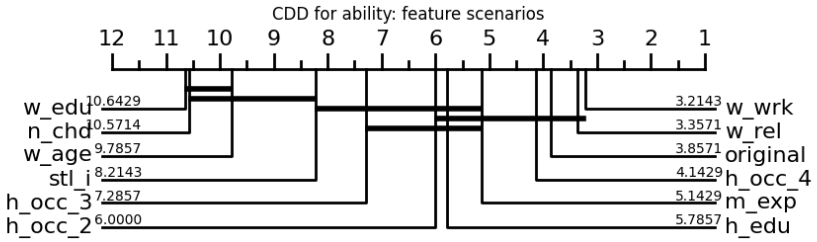
APPENDIX F – CRITICAL DIFFERENCE DIAGRAMS

In this chapter we show the critical difference diagrams for all datasets analyzed in the experimental evaluation of the current work.

F.1 SIMPLE IMPUTATION

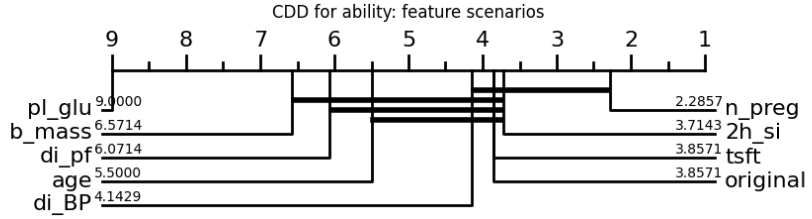
Figure 76 shows the CDD ability results for the Contraceptive dataset, Figure 77 for the Pima dataset, Figure 78 for the Wheat dataset, Figure 79 for the Haberman dataset, Figure 80 for the Cleveland dataset, Figure 81 for the Wine dataset, and Figure 82 for the Yeast dataset.

Figure 76 – CDD for Ability per Scenario: Contraceptive with Simple Imputation



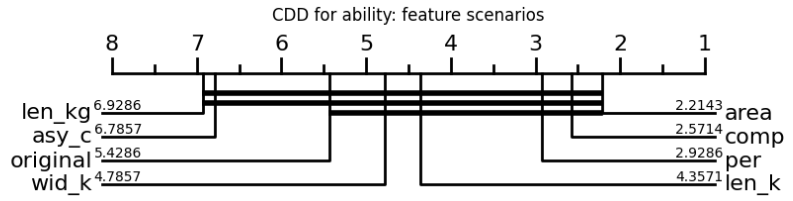
Source: Author's own elaboration

Figure 77 – CDD for Ability per Scenario: Pima with Simple Imputation



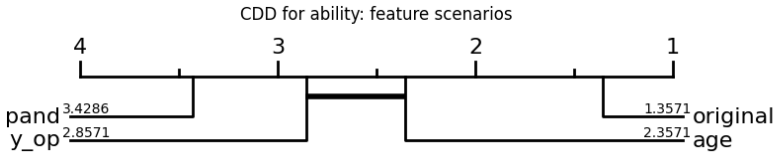
Source: Author's own elaboration

Figure 78 – CDD for Ability per Scenario: Wheat with Simple Imputation



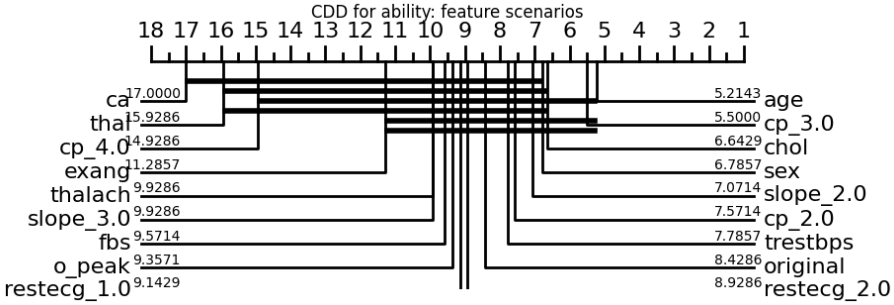
Source: Author's own elaboration

Figure 79 – CDD for Ability per Scenario: Haberman with Simple Imputation



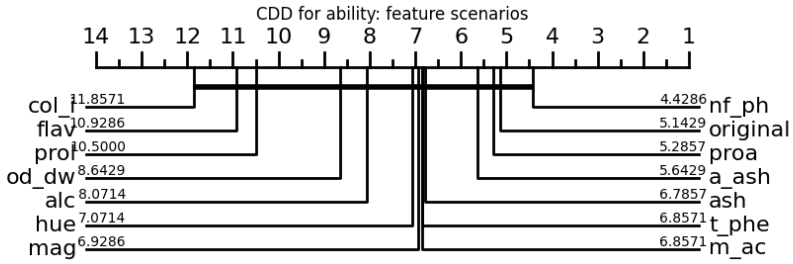
Source: Author's own elaboration

Figure 80 – CDD for Ability per Scenario: Cleveland with Simple Imputation



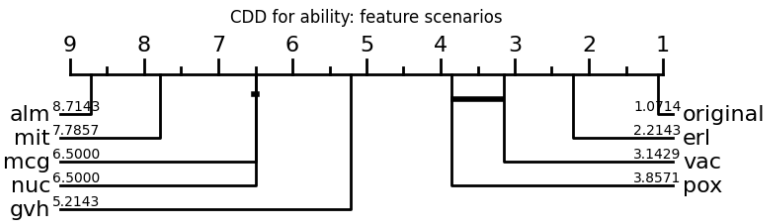
Source: Author's own elaboration

Figure 81 – CDD for Ability per Scenario: Wine with Simple Imputation



Source: Author's own elaboration

Figure 82 – CDD for Ability per Scenario: Yeast with Simple Imputation

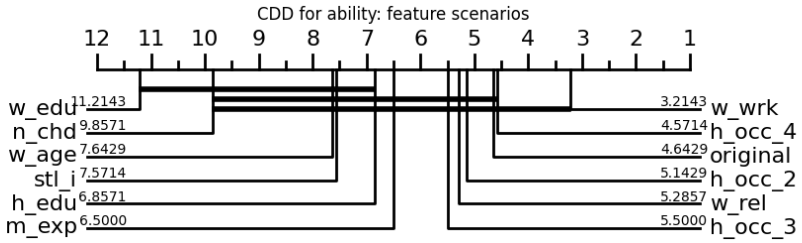


Source: Author's own elaboration

F.2 KNN IMPUTATION

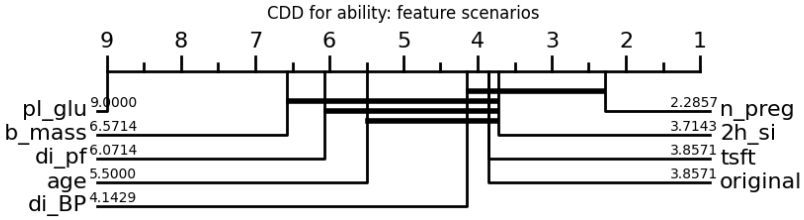
Figure 83 shows the CDD ability results for the Contraceptive dataset, Figure 84 for the Pima dataset, Figure 85 for the Wheat dataset, Figure 86 for the Haberman dataset, Figure 87 for the Cleveland dataset, Figure 88 for the Wine dataset, and Figure 89 for the Yeast dataset.

Figure 83 – CDD for Ability per Scenario: Contraceptive with KNN Imputation



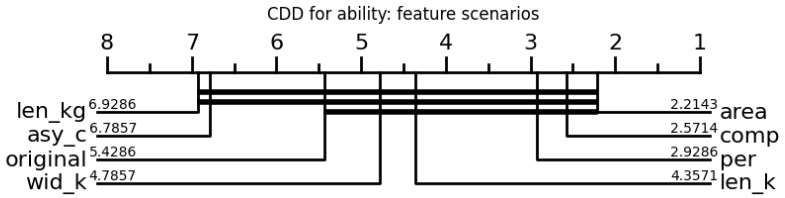
Source: Author's own elaboration

Figure 84 – CDD for Ability per Scenario: Pima with KNN Imputation



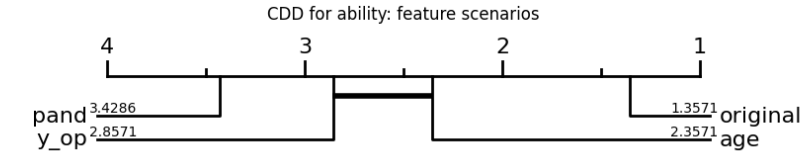
Source: Author's own elaboration

Figure 85 – CDD for Ability per Scenario: Wheat with KNN Imputation



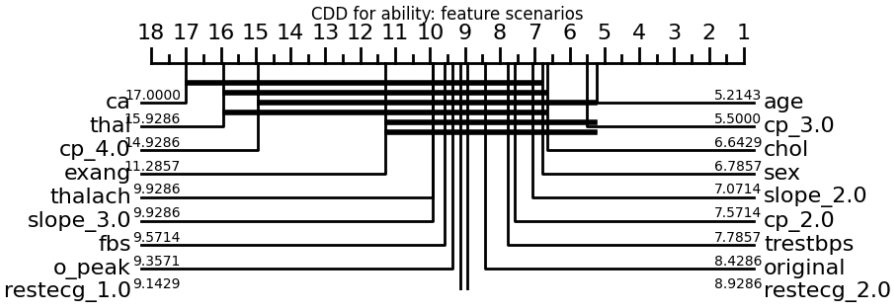
Source: Author's own elaboration

Figure 86 – CDD for Ability per Scenario: Haberman with KNN Imputation



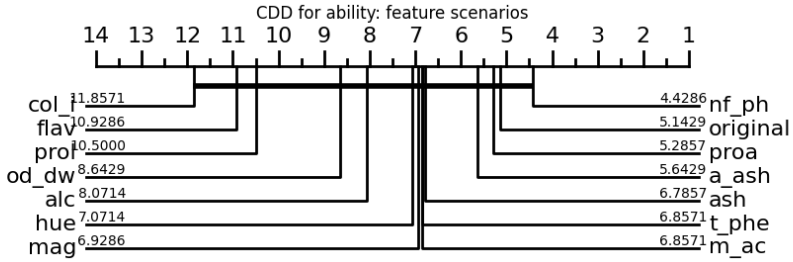
Source: Author's own elaboration

Figure 87 – CDD for Ability per Scenario: Cleveland with KNN Imputation



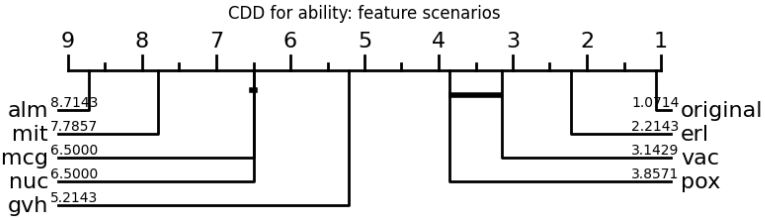
Source: Author's own elaboration

Figure 88 – CDD for Ability per Scenario: Wine with KNN Imputation



Source: Author's own elaboration

Figure 89 – CDD for Ability per Scenario: Yeast with KNN Imputation

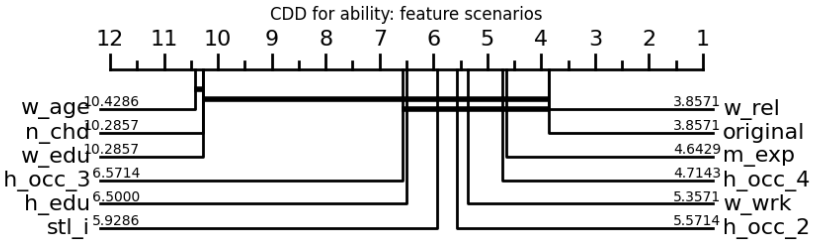


Source: Author's own elaboration

F.3 ITERATIVE IMPUTATION

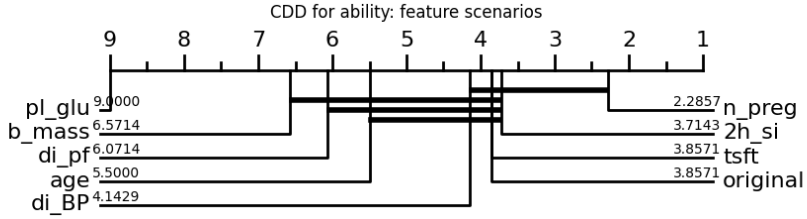
Figure 90 shows the CDD ability results for the Contraceptive dataset, Figure 91 for the Pima dataset, Figure 92 for the Wheat dataset, Figure 93 for the Haberman dataset, Figure 94 for the Cleveland dataset, Figure 95 for the Wine dataset, and Figure 96 for the Yeast dataset.

Figure 90 – CDD for Ability per Scenario: Contraceptive with Iterative Imputation



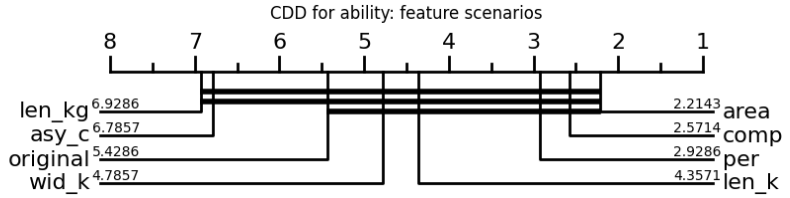
Source: Author's own elaboration

Figure 91 – CDD for Ability per Scenario: Pima with Iterative Imputation



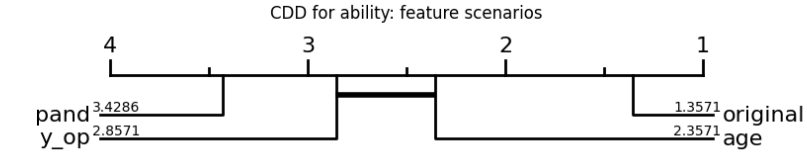
Source: Author's own elaboration

Figure 92 – CDD for Ability per Scenario: Wheat with Iterative Imputation



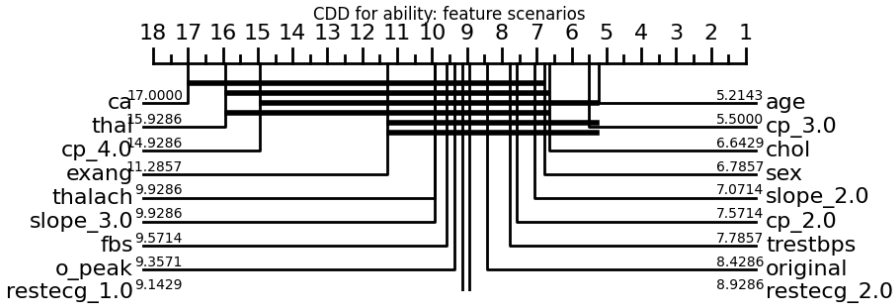
Source: Author's own elaboration

Figure 93 – CDD for Ability per Scenario: Haberman with Iterative Imputation



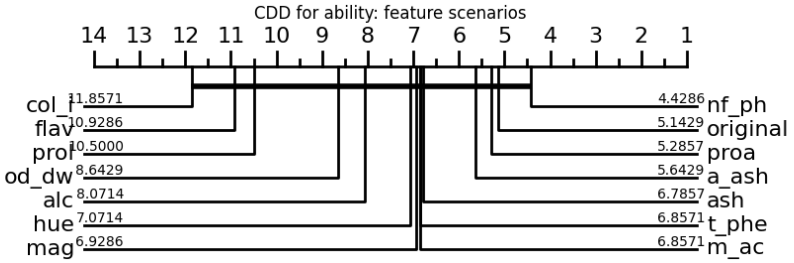
Source: Author's own elaboration

Figure 94 – CDD for Ability per Scenario: Cleveland with Iterative Imputation



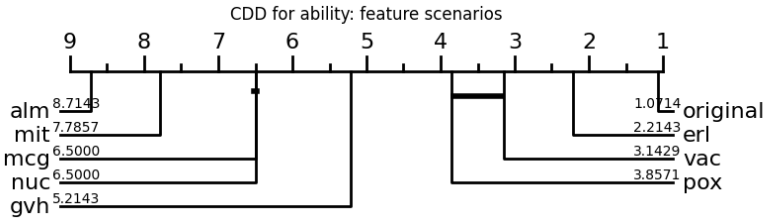
Source: Author's own elaboration

Figure 95 – CDD for Ability per Scenario: Wine with Iterative Imputation



Source: Author's own elaboration

Figure 96 – CDD for Ability per Scenario: Yeast with Iterative Imputation



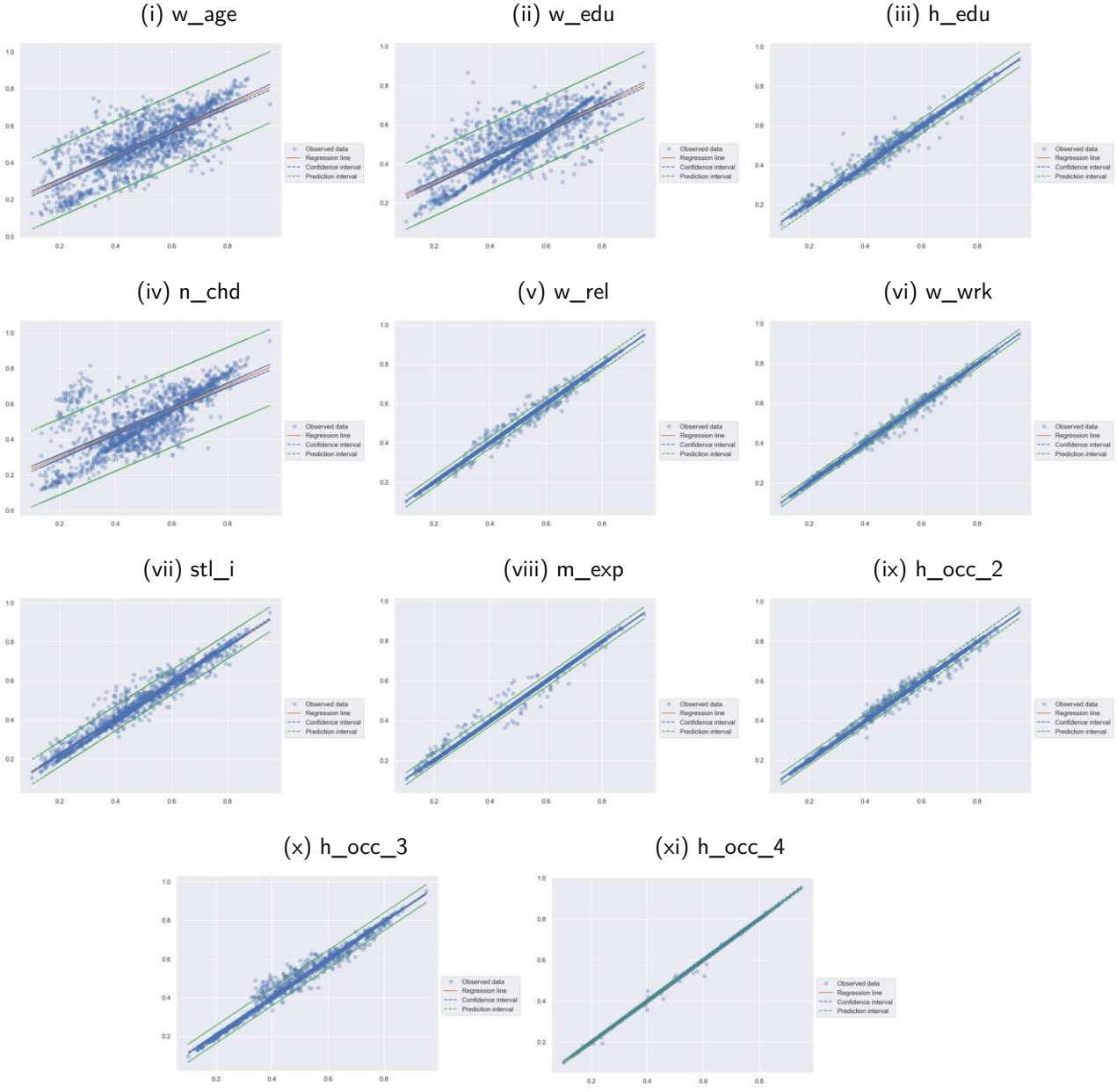
Source: Author's own elaboration

APPENDIX G – OLS REGRESSION ANALYSIS

G.1 SIMPLE IMPUTATION

Figure 97 shows the results for the Contraceptive dataset, Figure 98 for the Pima dataset, Figure 99 for the Wheat dataset, Figure 100 for the Haberman dataset, Figure 101 for the Cleveland dataset, Figure 101 for the Ionosphere dataset, Figure 100 for the Wine dataset, and finally, Figure 100 shows the results for the Yeast dataset.

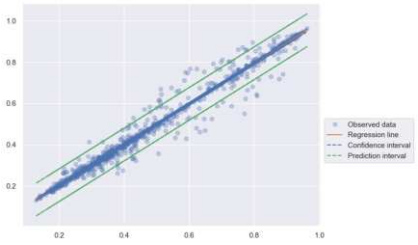
Figure 97 – OLS model fit for Contraceptive dataset - Simple Imputation



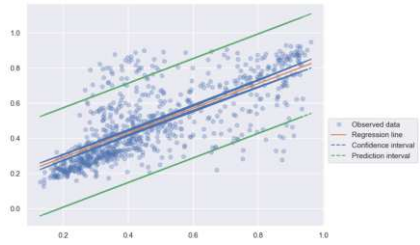
Source: Author's own elaboration

Figure 98 – OLS model fit for Pima dataset - Simple Imputation

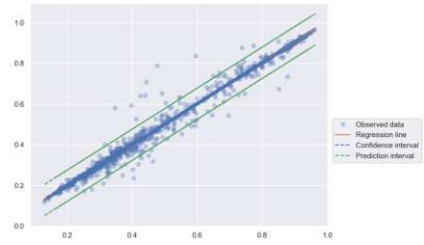
(i) n_preg



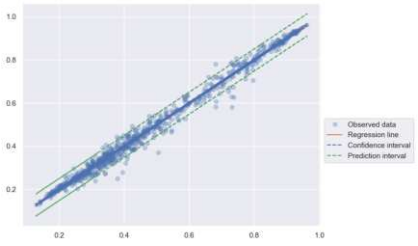
(ii) pl_glu



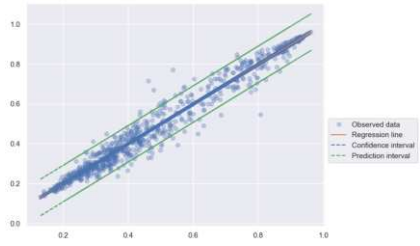
(iii) di_BP



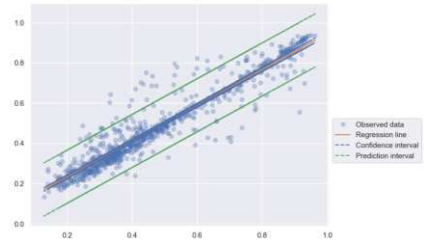
(iv) tsft



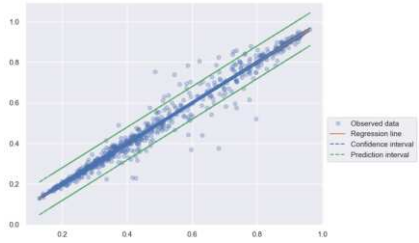
(v) 2h_si



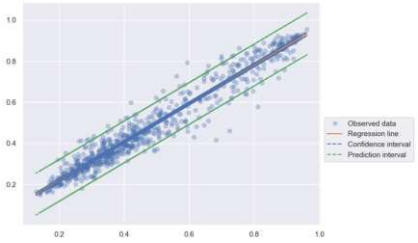
(vi) b_mass



(vii) di_pf



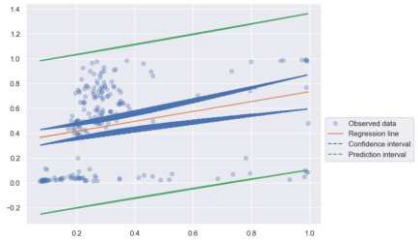
(viii) age



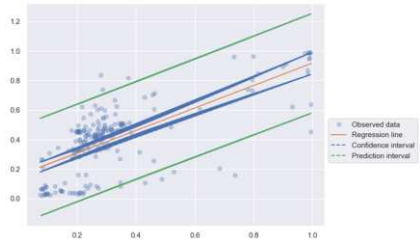
Source: Author's own elaboration

Figure 99 – OLS model fit for Wheat dataset - Simple Imputation

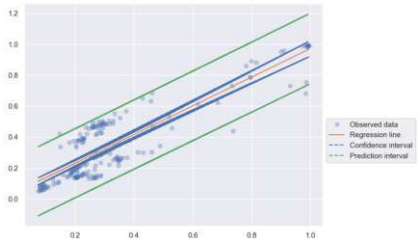
(i) area



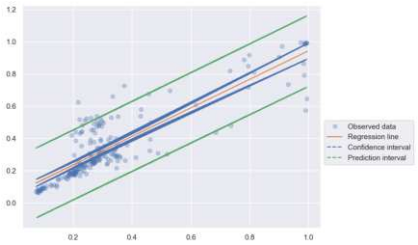
(ii) per



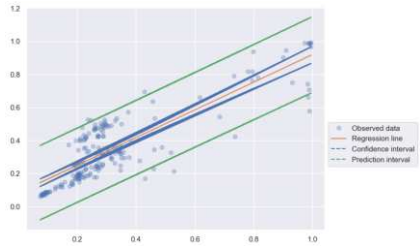
(iii) comp



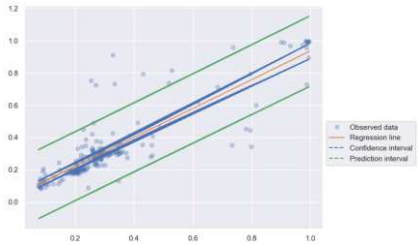
(iv) len_k



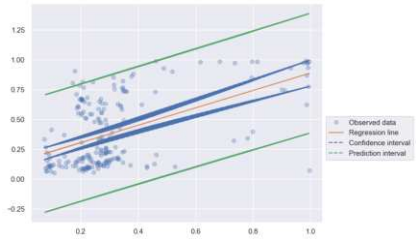
(v) wid_k



(vi) asy_c



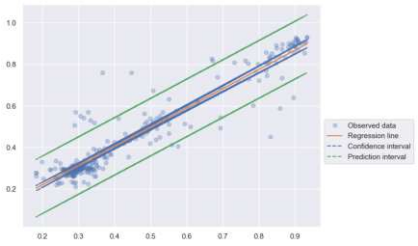
(vii) len_kg



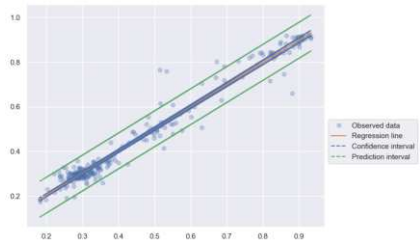
Source: Author's own elaboration

Figure 100 – OLS model fit for Haberman dataset - Simple Imputation

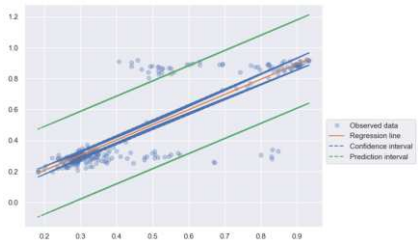
(i) age



(ii) y_op



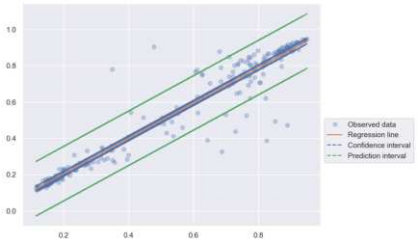
(iii) pand



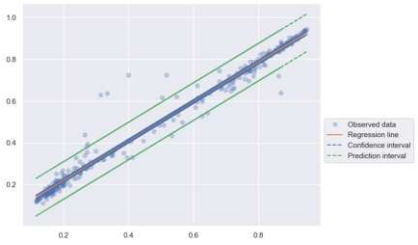
Source: Author's own elaboration

Figure 101 – OLS model fit for Cleveland dataset - Simple Imputation

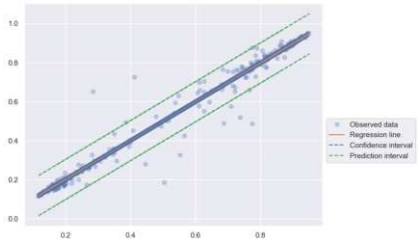
(i) age



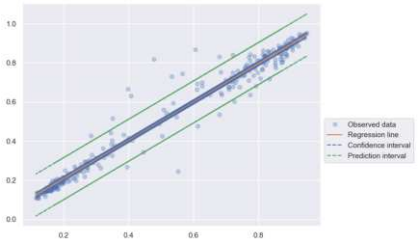
(ii) sex



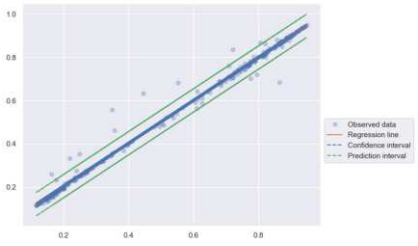
(iii) trestbps



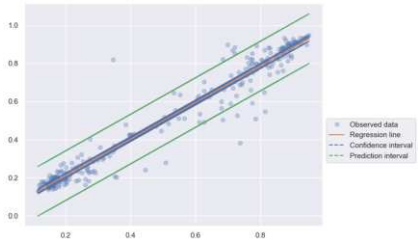
(iv) chol



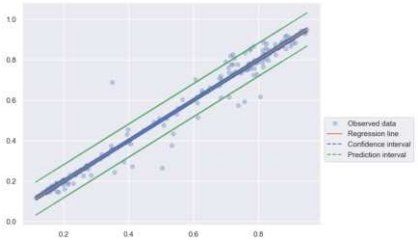
(v) fbs



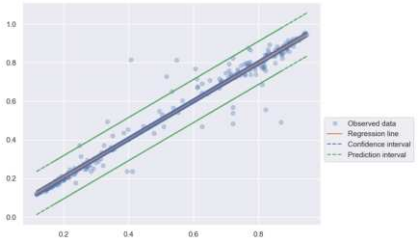
(vi) thalach



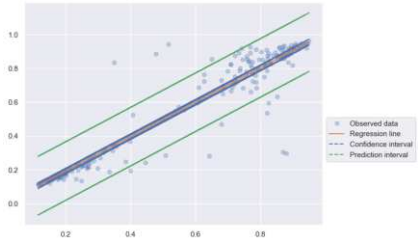
(vii) exang



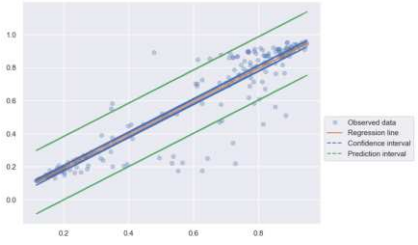
(viii) o_peak



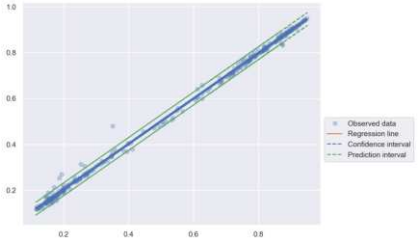
(ix) ca



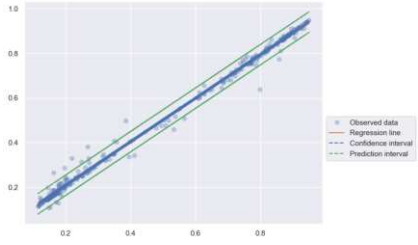
(x) thal

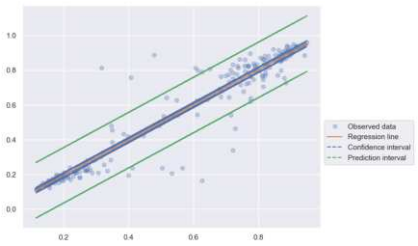


(xi) cp_2.0

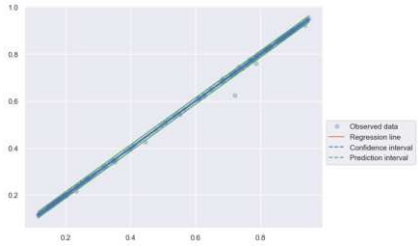


(xii) cp_3.0

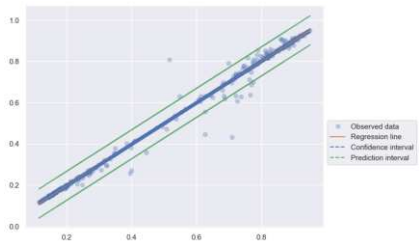




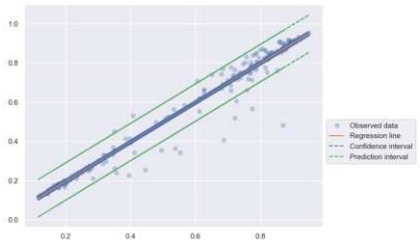
(xiii) cp_4.0



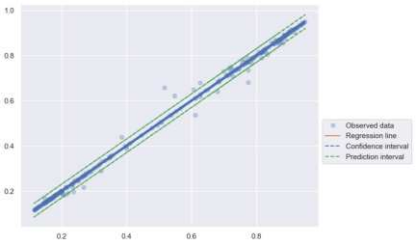
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

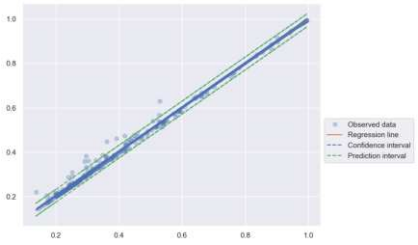


(xvii) slope_3.0

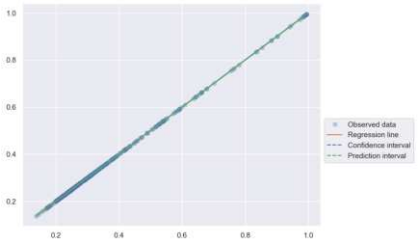
Source: Author's own elaboration

Figure 101 – OLS model fit for Ionosphere dataset - Simple Imputation

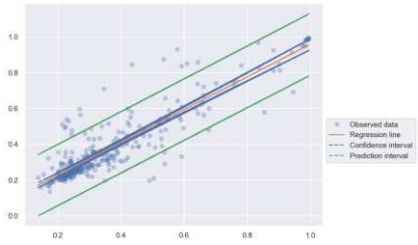
(i) col_0



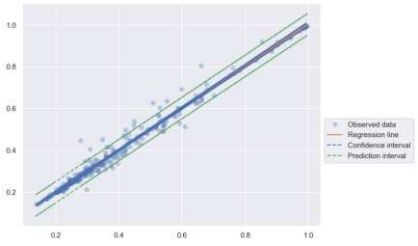
(ii) col_1



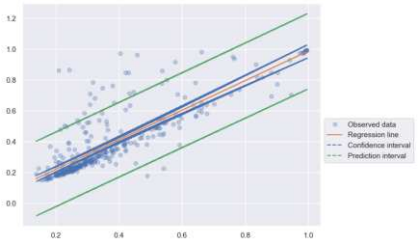
(iii) col_2



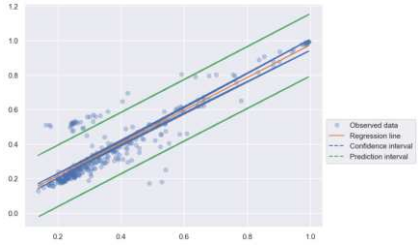
(iv) col_3



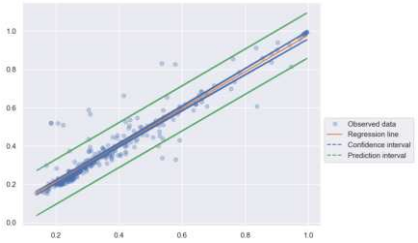
(v) col_4



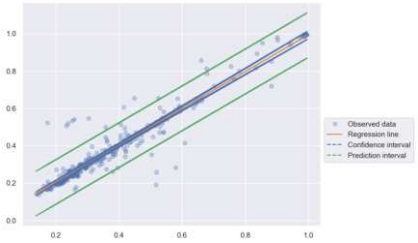
(vi) col_5



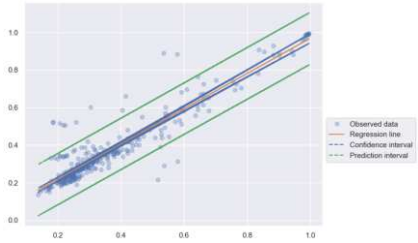
(vii) col_6



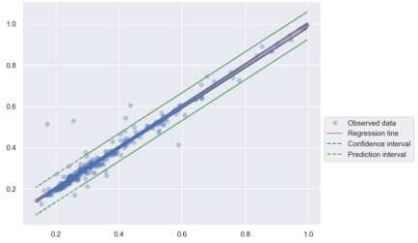
(viii) col_7



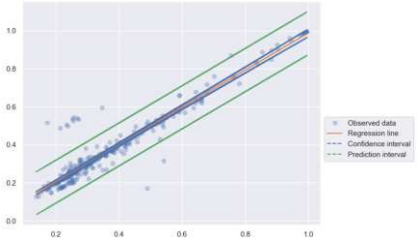
(ix) col_8



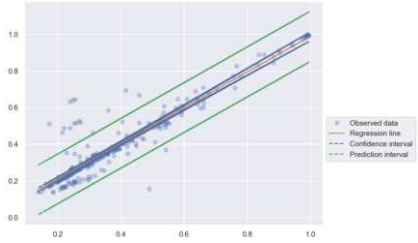
(x) col_9

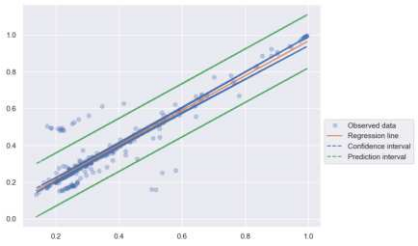


(xi) col_10

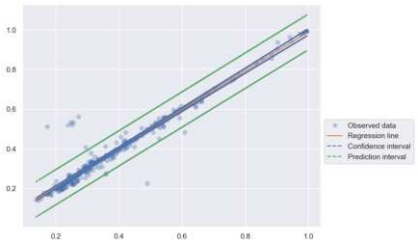


(xii) col_11

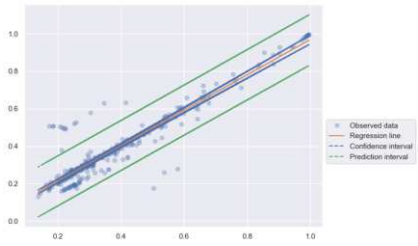




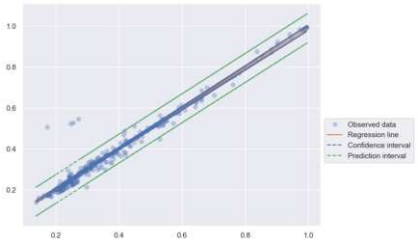
(xiii) col_12



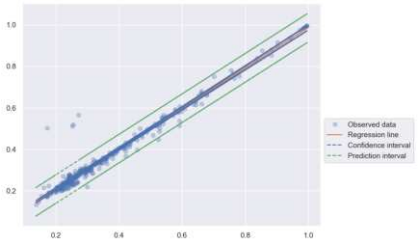
(xiv) col_13



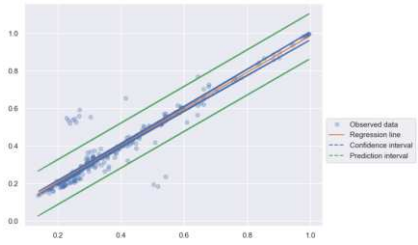
(xv) col_14



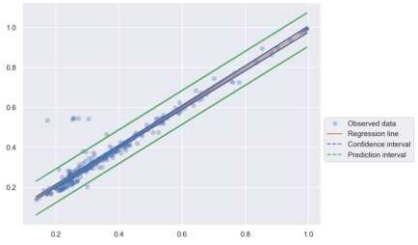
(xvi) col_15



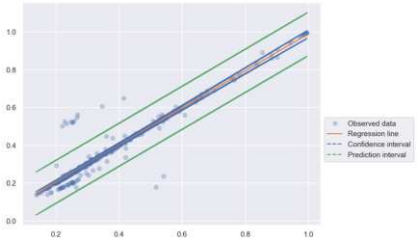
(xvii) col_16



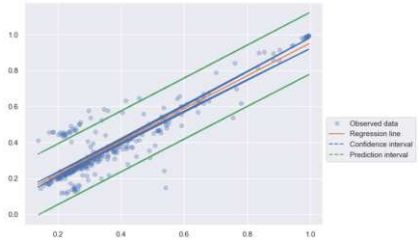
(xviii) col_17



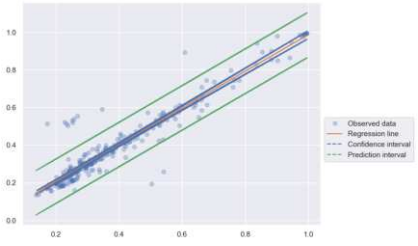
(xix) col_18



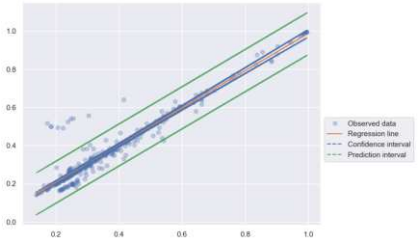
(xx) col_19



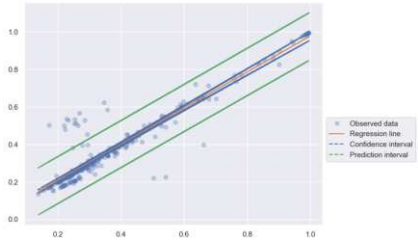
(xxi) col_20



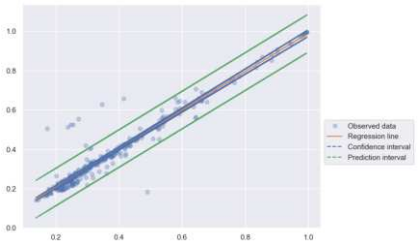
(xxii) col_21



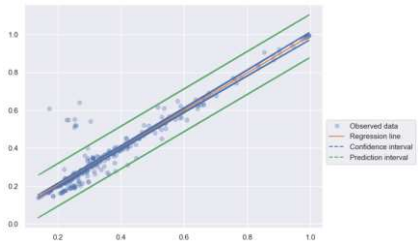
(xxiii) col_22



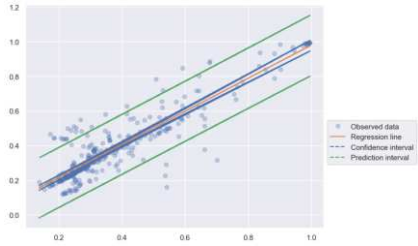
(xxiv) col_23



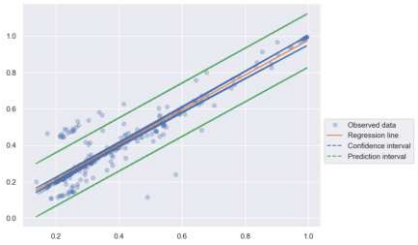
(xxv) col_24



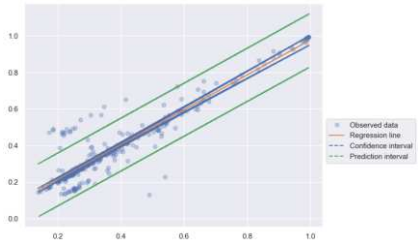
(xxvi) col_25



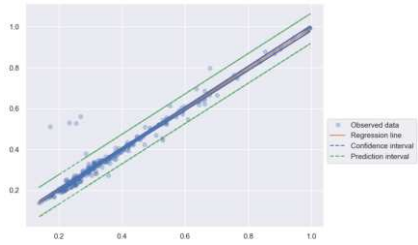
(xxvii) col_26



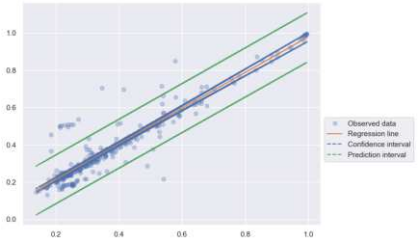
(xxviii) col_27



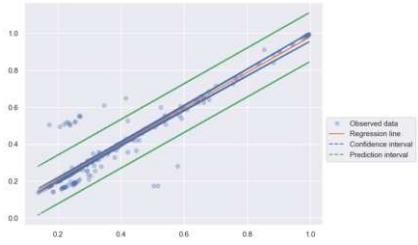
(xxix) col_28



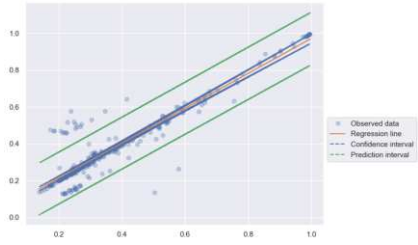
(xxx) col_29



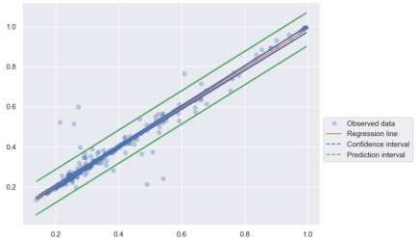
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

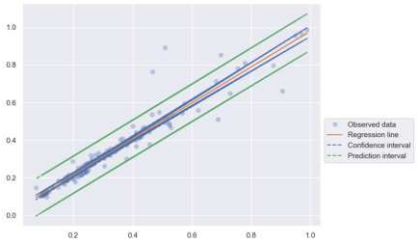


(xxxiv) col_33

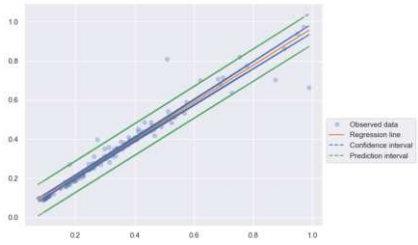
Source: Author's own elaboration

Figure 100 – OLS model fit for Wine dataset - Simple Imputation

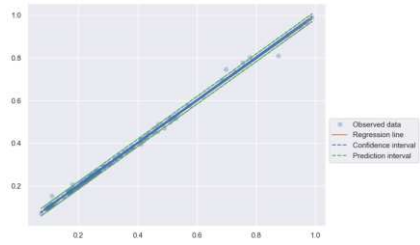
(i) alc



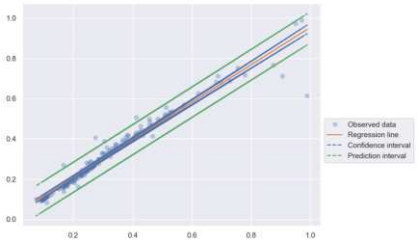
(ii) m_ac



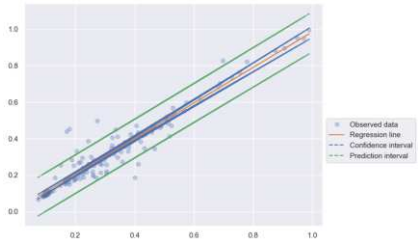
(iii) ash



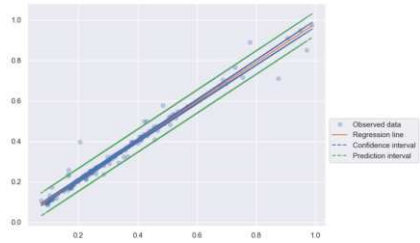
(iv) a_ash



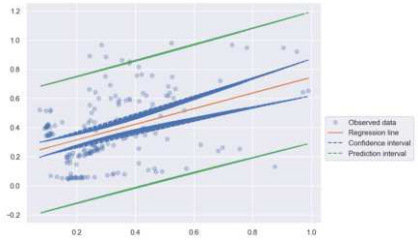
(v) mag



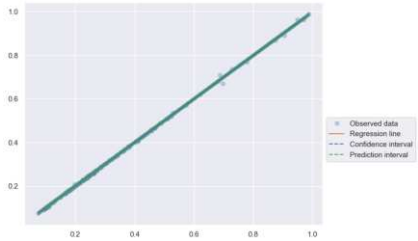
(vi) t_phe



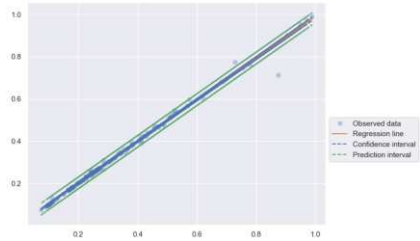
(vii) flav



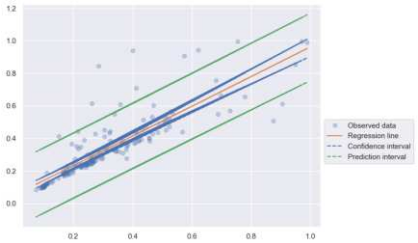
(viii) nf_ph



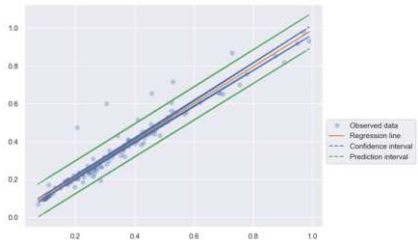
(ix) proa



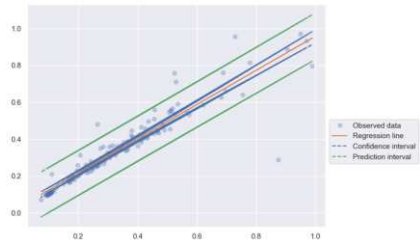
(x) col_i

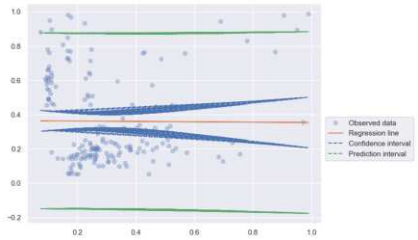


(xi) hue



(xii) od_dw

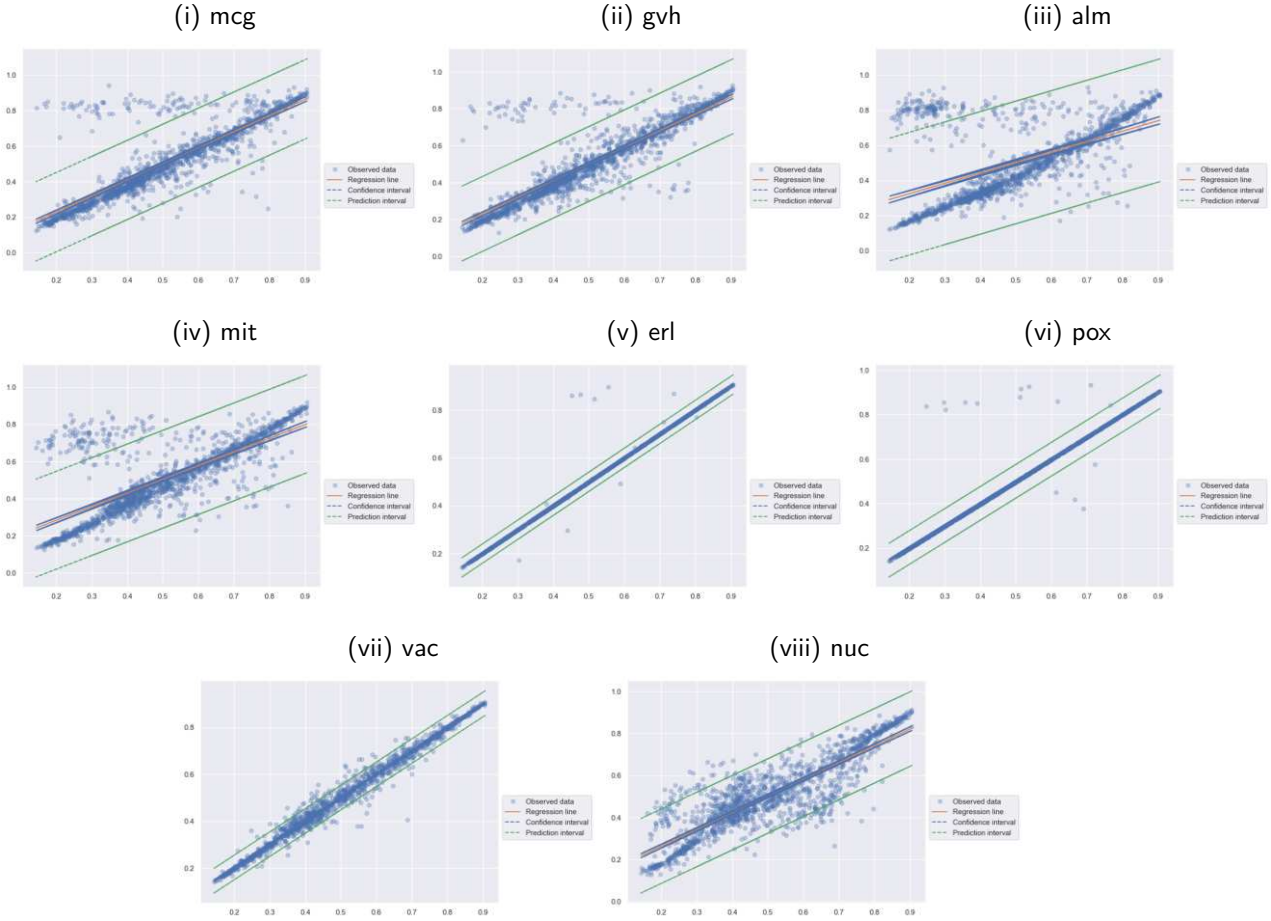




(xiii) prol

Source: Author's own elaboration

Figure 100 – OLS model fit for Yeast dataset - Simple Imputation



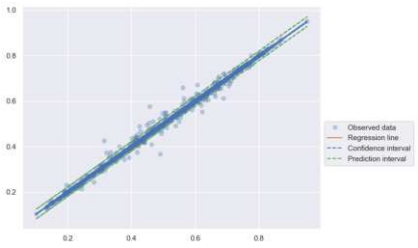
Source: Author's own elaboration

G.2 KNN IMPUTATION

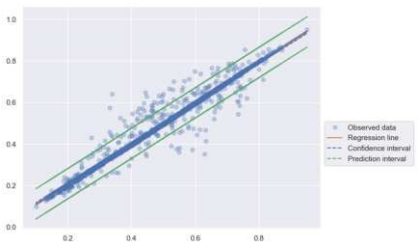
Figure 101 shows the results for the Contraceptive dataset, Figure 102 for the Pima dataset, Figure 103 for the Wheat dataset, Figure 104 for the Haberman dataset, Figure 105 for the Cleveland dataset, Figure 105 for the Ionosphere dataset, Figure 104 for the Wine dataset, and finally, Figure 104 shows the results for the Yeast dataset.

Figure 101 – OLS model fit for Contraceptive dataset - KNN Imputation

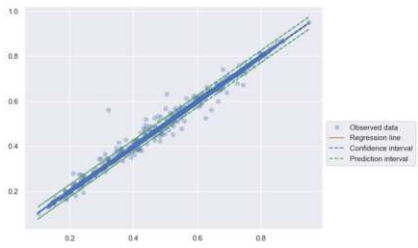
(i) w_age



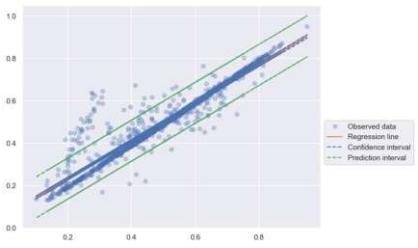
(ii) w_edu



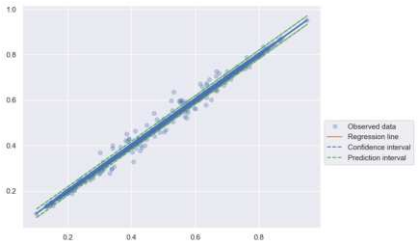
(iii) h_edu



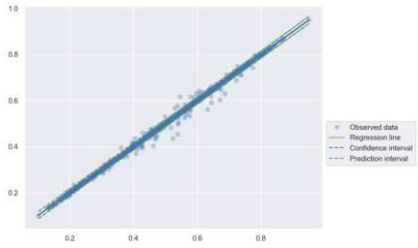
(iv) n_chd



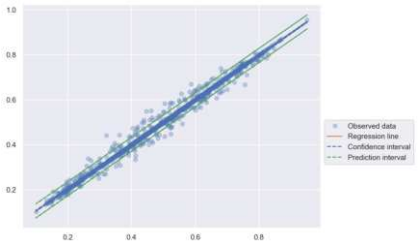
(v) w_rel



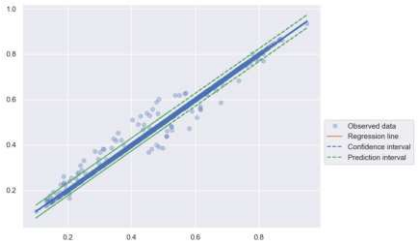
(vi) w_wrk



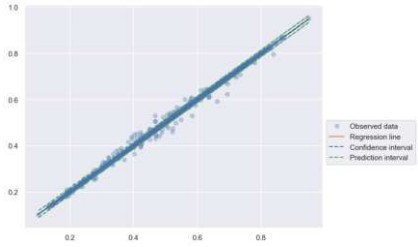
(vii) stl_i



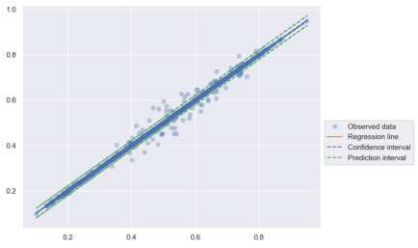
(viii) m_exp



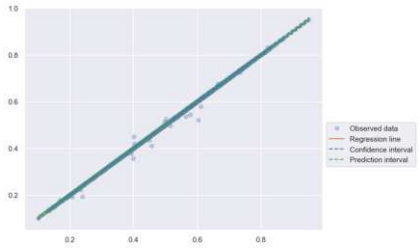
(ix) h_occ_2



(x) h_occ_3



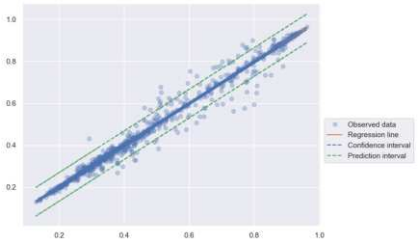
(xi) h_occ_4



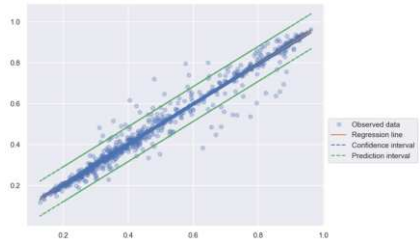
Source: Author's own elaboration

Figure 102 – OLS model fit for Pima dataset - KNN Imputation

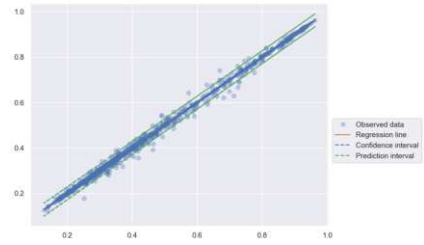
(i) n_preg



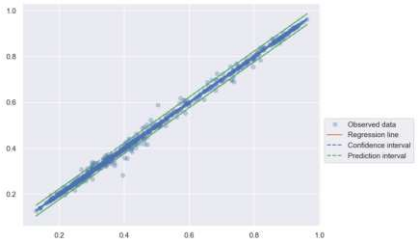
(ii) pl_glu



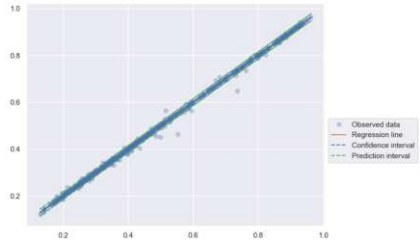
(iii) di_BP



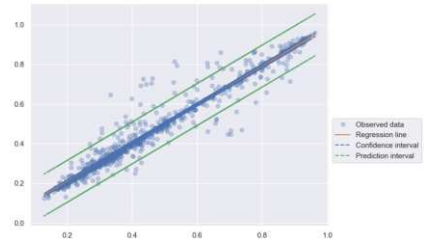
(iv) tsft



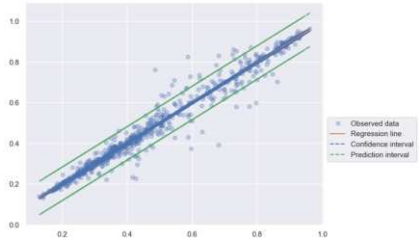
(v) 2h_si



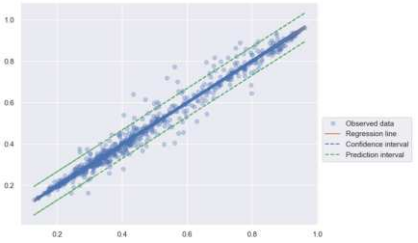
(vi) b_mass



(vii) di_pf



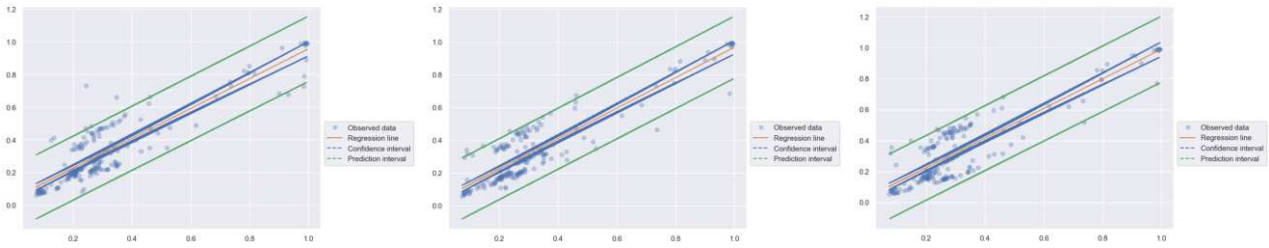
(viii) age



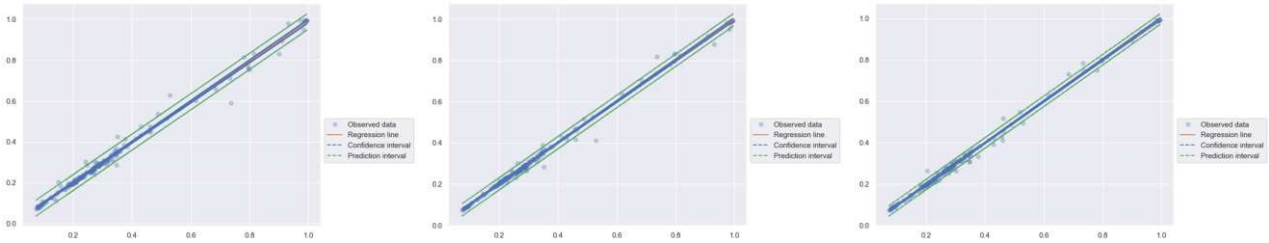
Source: Author's own elaboration

Figure 103 – OLS model fit for Wheat dataset - KNN Imputation

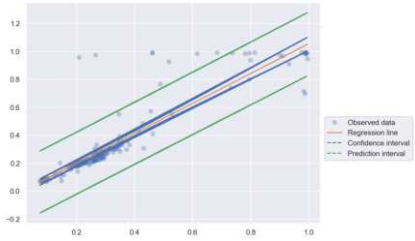
(i) area (ii) per (iii) comp



(iv) len_k (v) wid_k (vi) asy_c



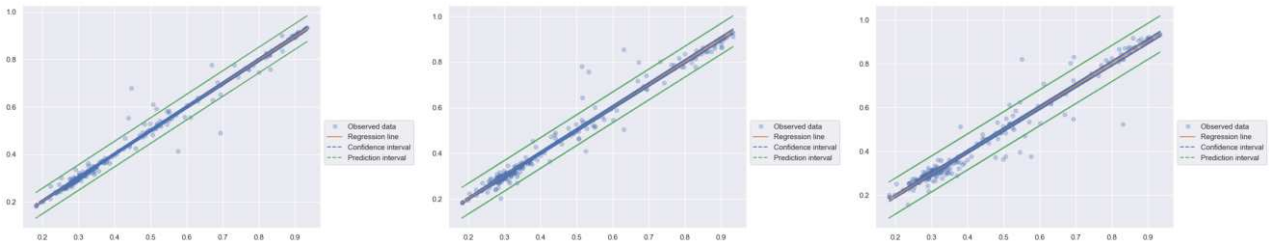
(vii) len_kg



Source: Author's own elaboration

Figure 104 – OLS model fit for Haberman dataset - KNN Imputation

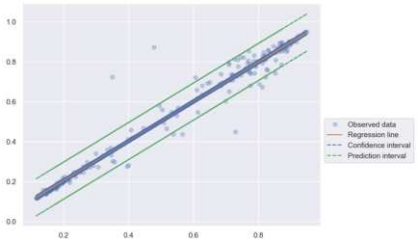
(i) age (ii) y_op (iii) pand



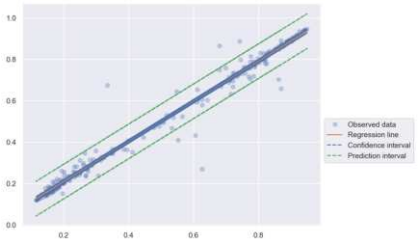
Source: Author's own elaboration

Figure 105 – OLS model fit for Cleveland dataset - KNN Imputation

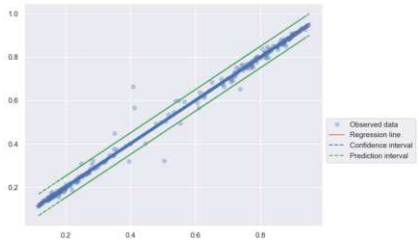
(i) age



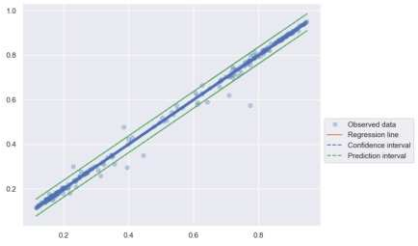
(ii) sex



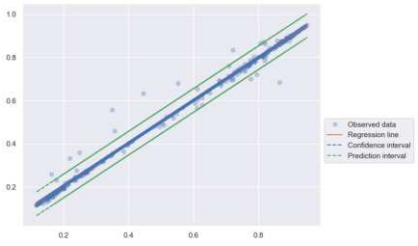
(iii) trestbps



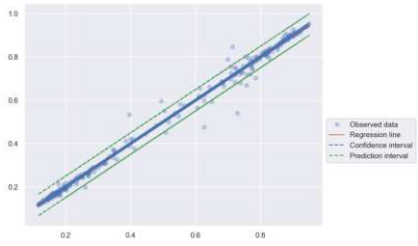
(iv) chol



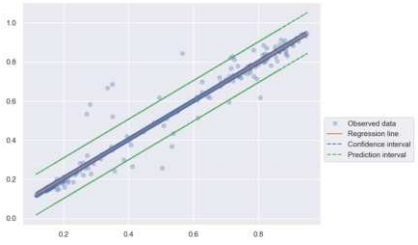
(v) fbs



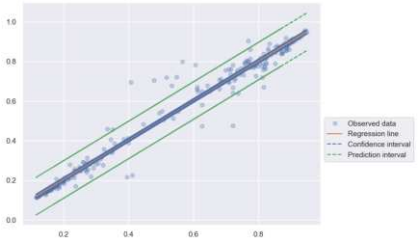
(vi) thalach



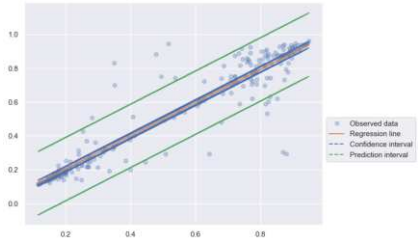
(vii) exang



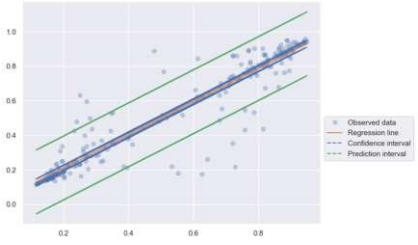
(viii) o_peak



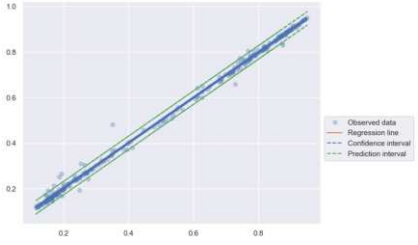
(ix) ca



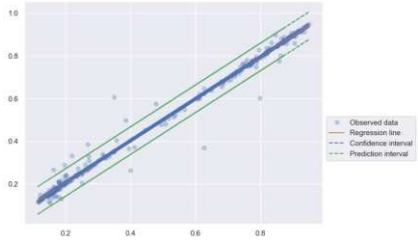
(x) thal

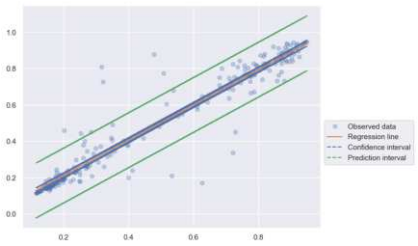


(xi) cp_2.0

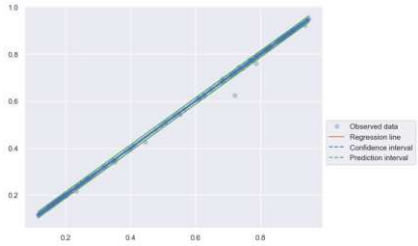


(xii) cp_3.0

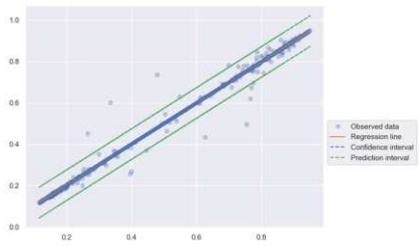




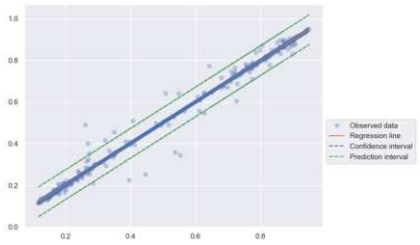
(xiii) cp_4.0



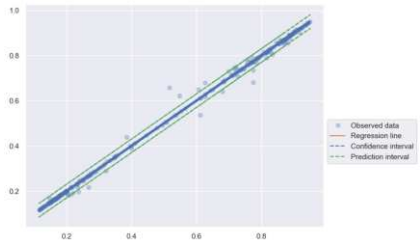
(xiv) restecg_1.0



(xv) restecg_2.0



(xvi) slope_2.0

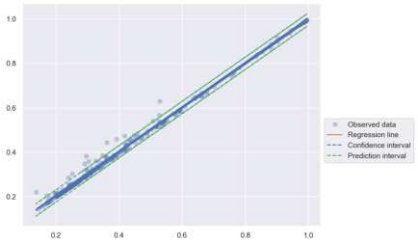


(xvii) slope_3.0

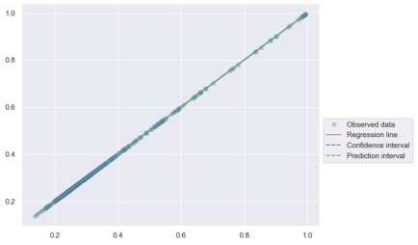
Source: Author's own elaboration

Figure 105 – OLS model fit for Ionosphere dataset - KNN Imputation

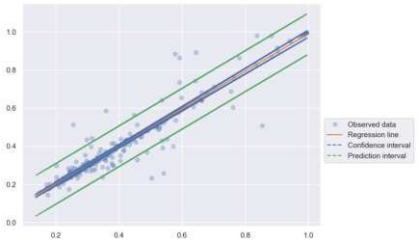
(i) col_0



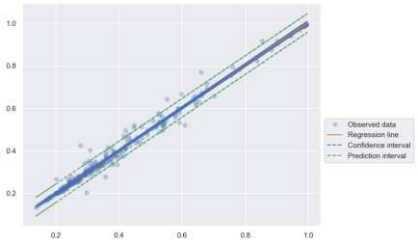
(ii) col_1



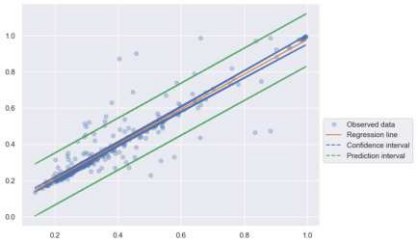
(iii) col_2



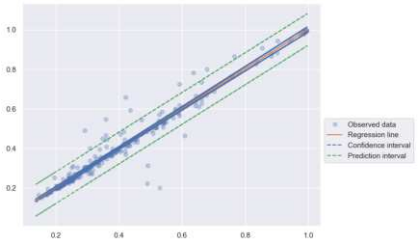
(iv) col_3



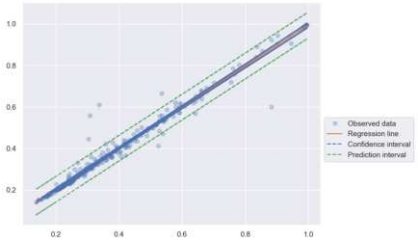
(v) col_4



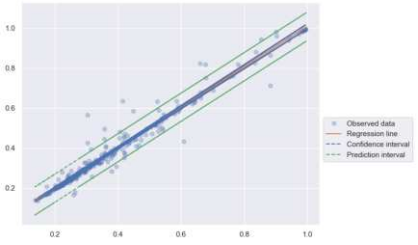
(vi) col_5



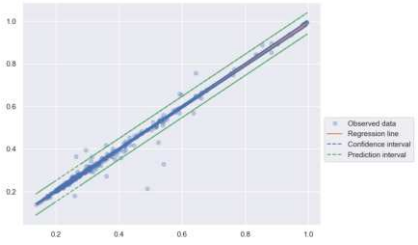
(vii) col_6



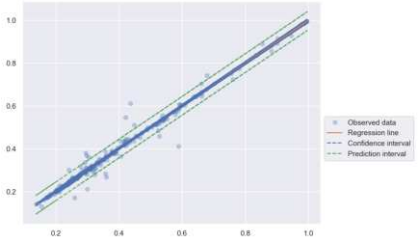
(viii) col_7



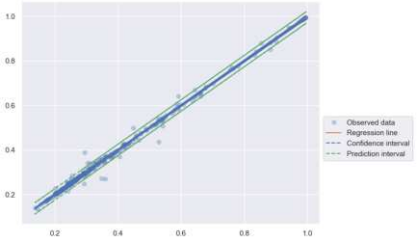
(ix) col_8



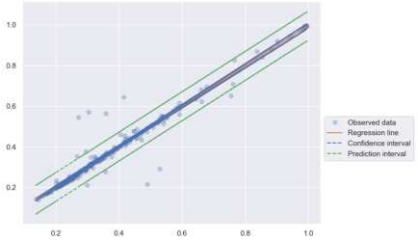
(x) col_9

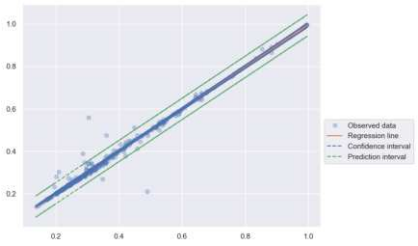


(xi) col_10

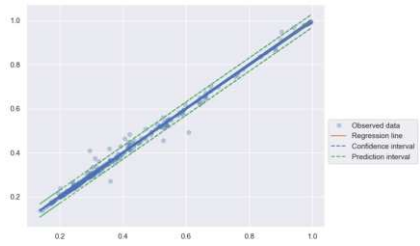


(xii) col_11

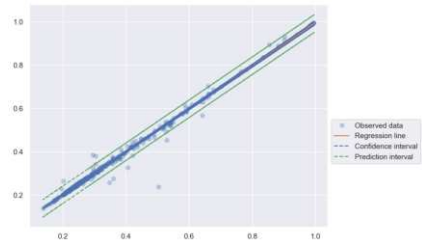




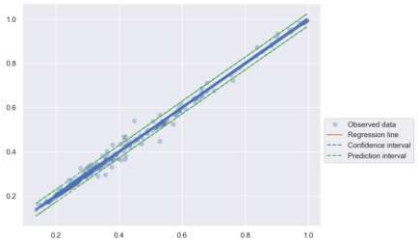
(xiii) col_12



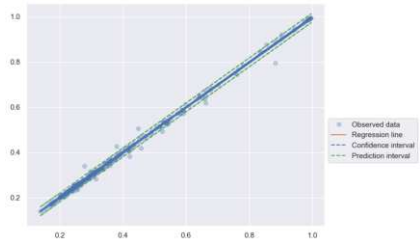
(xiv) col_13



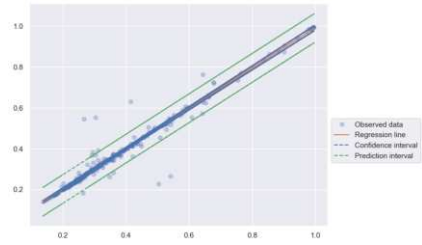
(xv) col_14



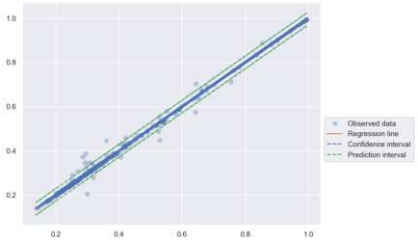
(xvi) col_15



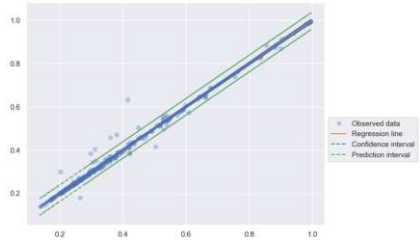
(xvii) col_16



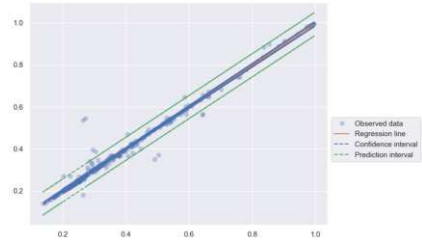
(xviii) col_17



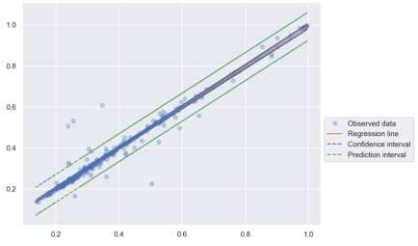
(xix) col_18



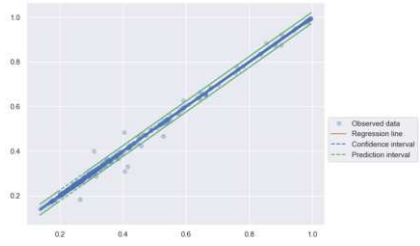
(xx) col_19



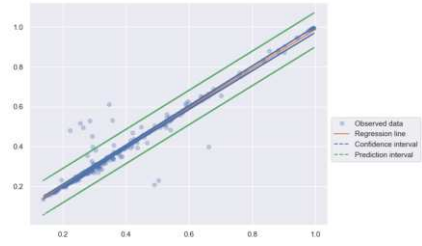
(xxi) col_20



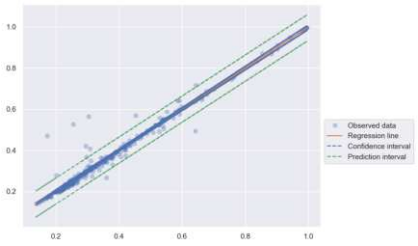
(xxii) col_21



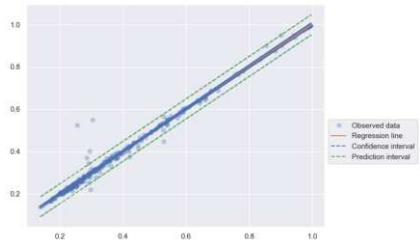
(xxiii) col_22



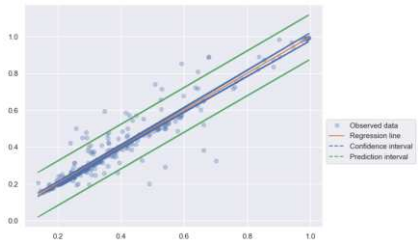
(xxiv) col_23



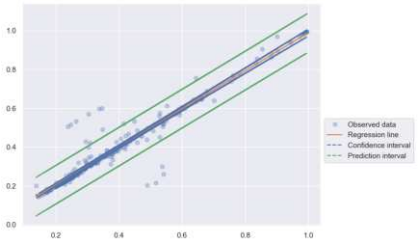
(xxv) col_24



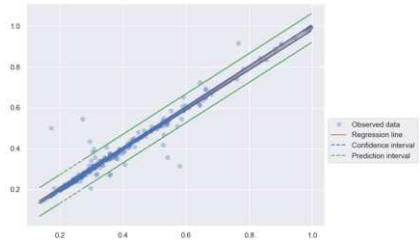
(xxvi) col_25



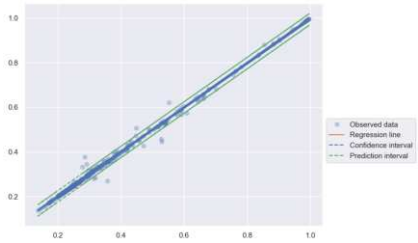
(xxvii) col_26



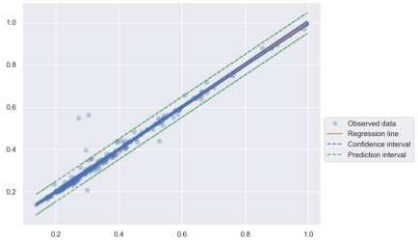
(xxviii) col_27



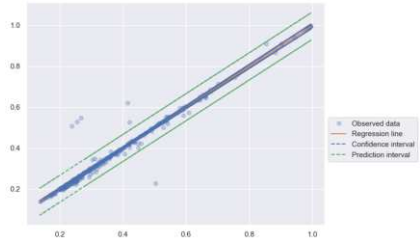
(xxix) col_28



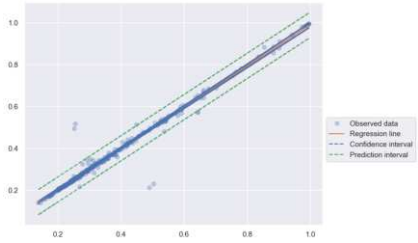
(xxx) col_29



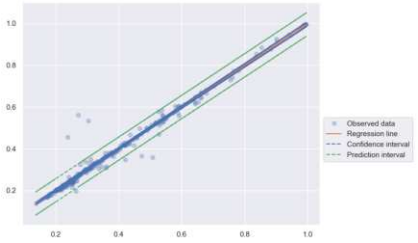
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

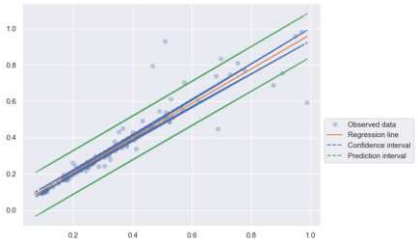


(xxxiv) col_33

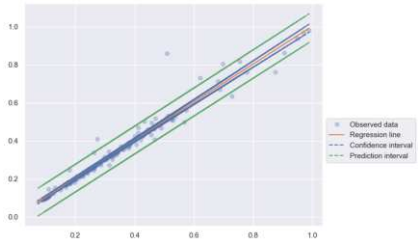
Source: Author's own elaboration

Figure 104 – OLS model fit for Wine dataset - KNN Imputation

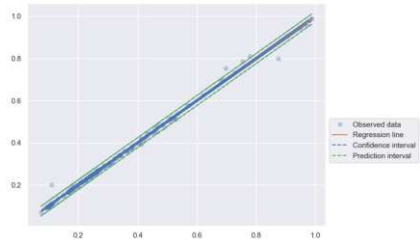
(i) alc



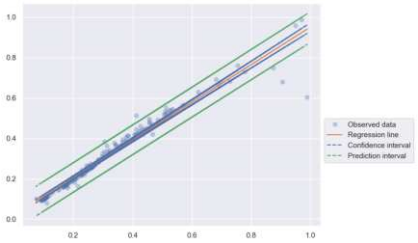
(ii) m_ac



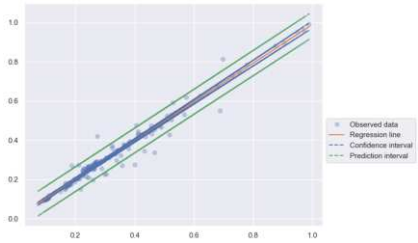
(iii) ash



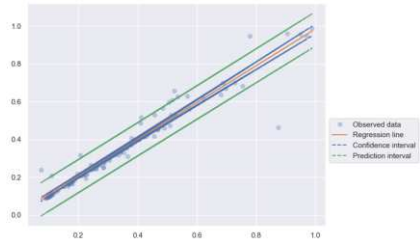
(iv) a_ash



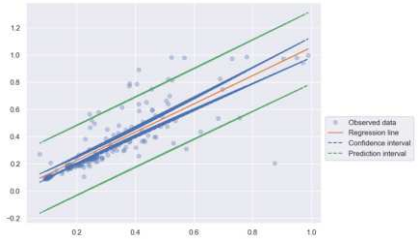
(v) mag



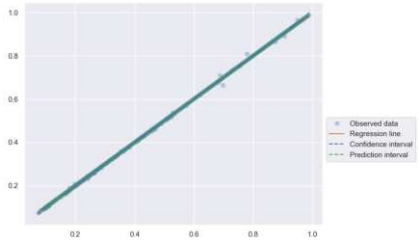
(vi) t_phe



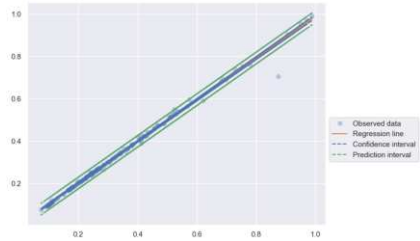
(vii) flav



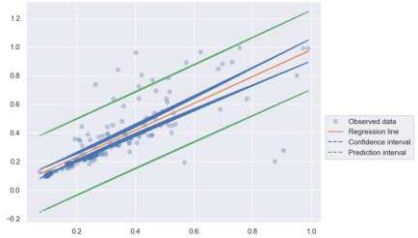
(viii) nf_ph



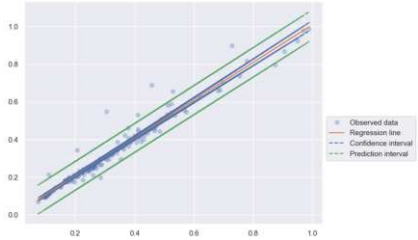
(ix) proa



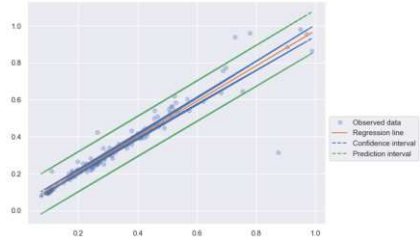
(x) col_i

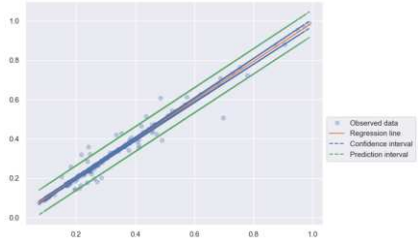


(xi) hue



(xii) od_dw

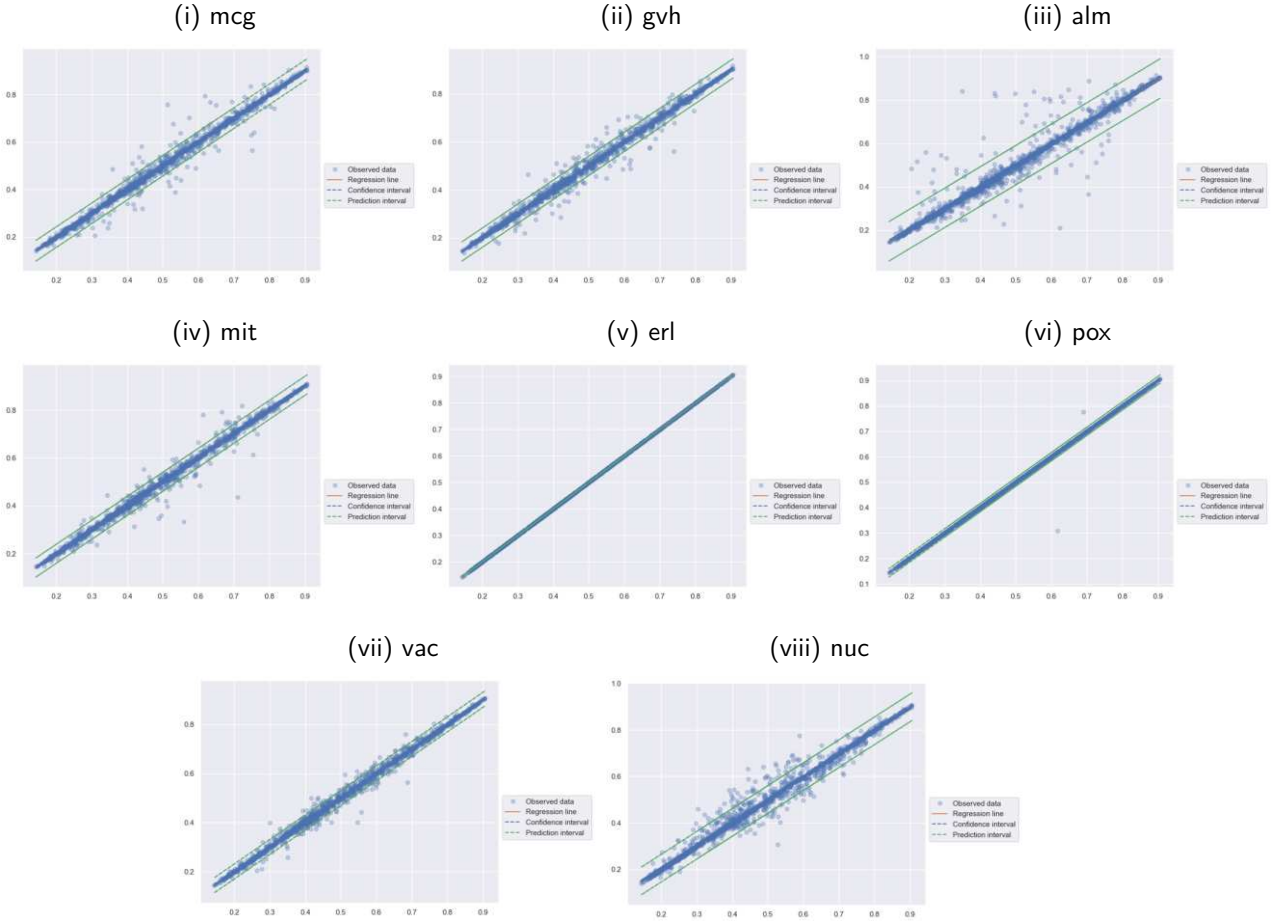




(xiii) prol

Source: Author's own elaboration

Figure 104 – OLS model fit for Yeast dataset - KNN Imputation



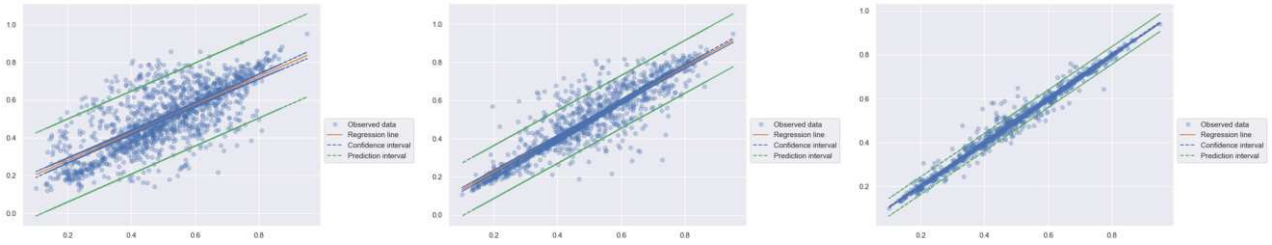
Source: Author's own elaboration

G.3 ITERATIVE IMPUTATION

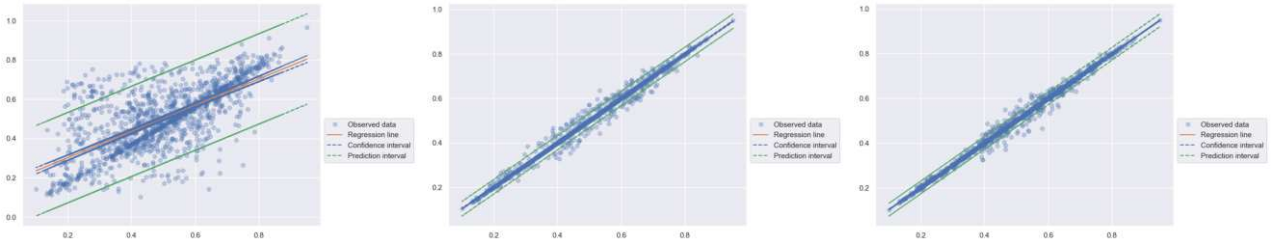
Figure 105 shows the results for the Contraceptive dataset, Figure 106 for the Pima dataset, Figure 107 for the Wheat dataset, Figure 108 for the Haberman dataset, Figure 109 for the Cleveland dataset, Figure 109 for the Ionosphere dataset, Figure 108 for the Wine dataset, and finally, Figure 108 shows the results for the Yeast dataset.

Figure 105 – OLS model fit for Contraceptive dataset - Iterative Imputation

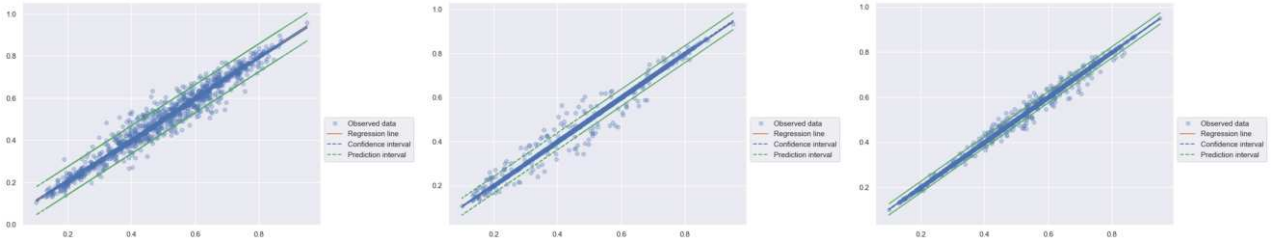
(i) w_age (ii) w_edu (iii) h_edu



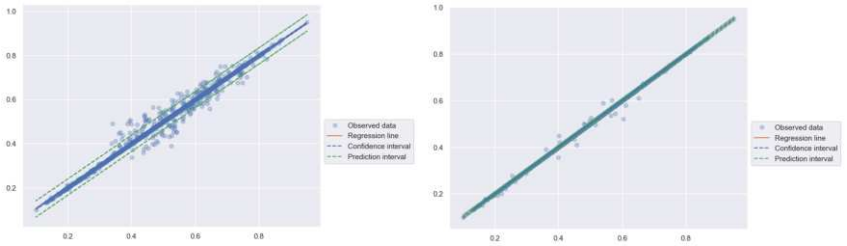
(iv) n_chd (v) w_rel (vi) w_wrk



(vii) stl_i (viii) m_exp (ix) h_occ_2



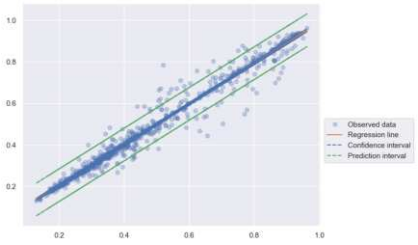
(x) h_occ_3 (xi) h_occ_4



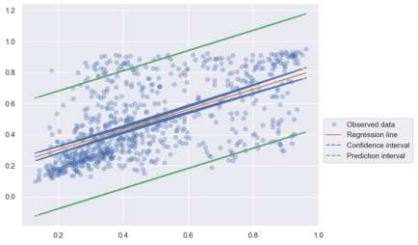
Source: Author's own elaboration

Figure 106 – OLS model fit for Pima dataset - Iterative Imputation

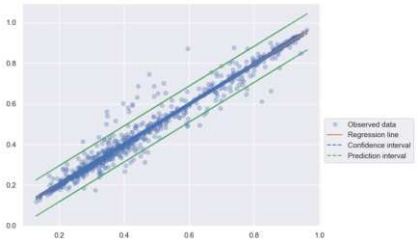
(i) n_preg



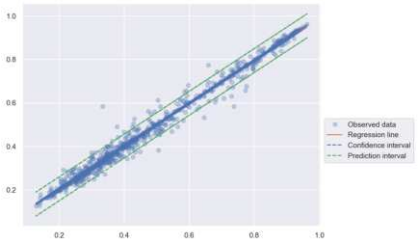
(ii) pl_glu



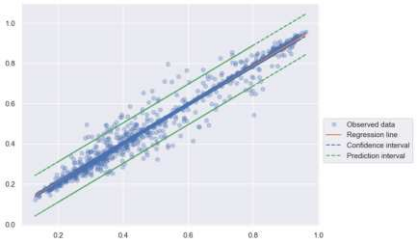
(iii) di_BP



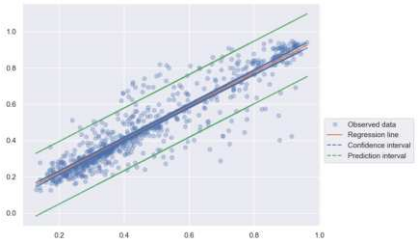
(iv) tsft



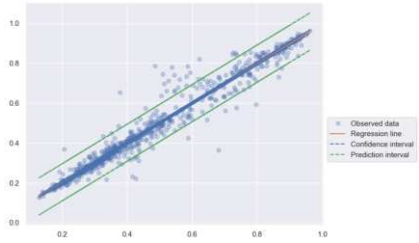
(v) 2h_si



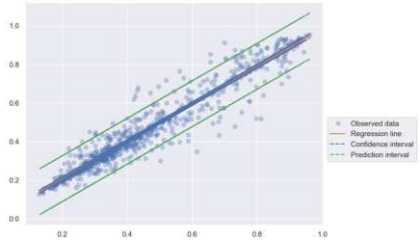
(vi) b_mass



(vii) di_pf



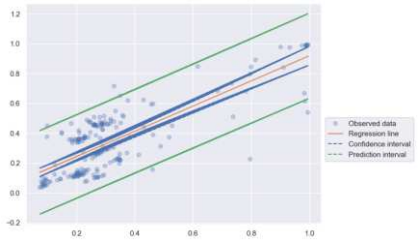
(viii) age



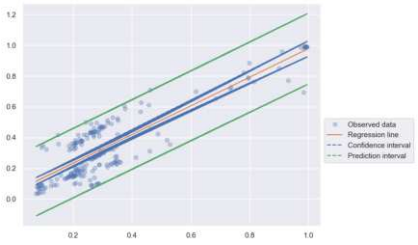
Source: Author's own elaboration

Figure 107 – OLS model fit for Wheat dataset - Iterative Imputation

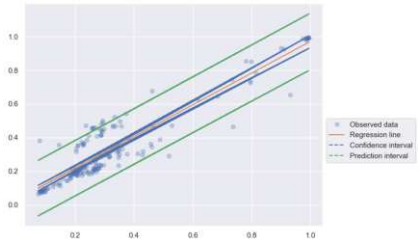
(i) area



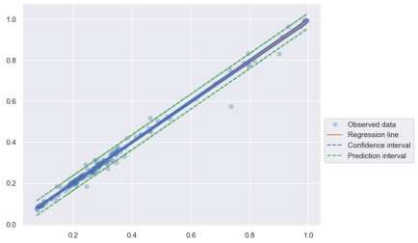
(ii) per



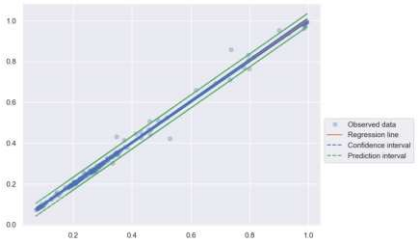
(iii) comp



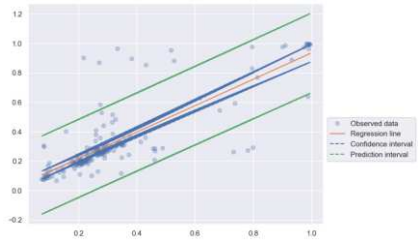
(iv) len_k



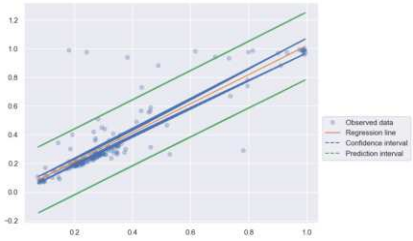
(v) wid_k



(vi) asy_c



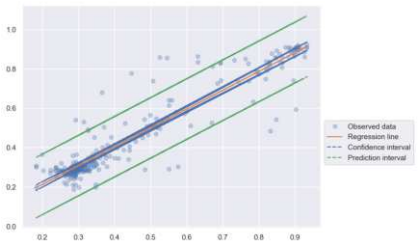
(vii) len_kg



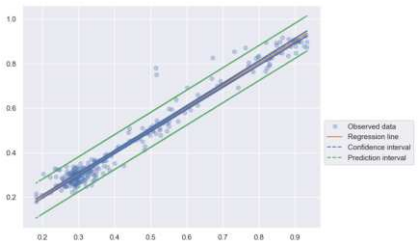
Source: Author's own elaboration

Figure 108 – OLS model fit for Haberman dataset - Iterative Imputation

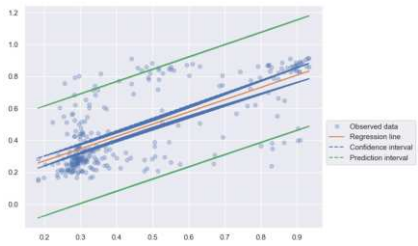
(i) age



(ii) y_op



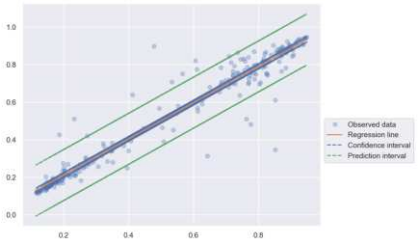
(iii) pand



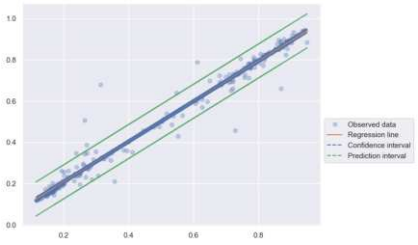
Source: Author's own elaboration

Figure 109 – OLS model fit for Cleveland dataset - Iterative Imputation

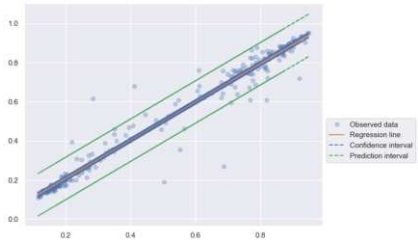
(i) age



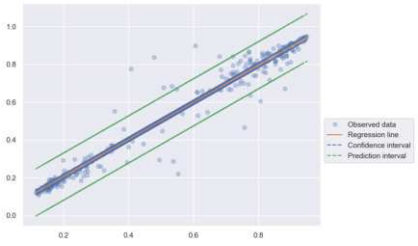
(ii) sex



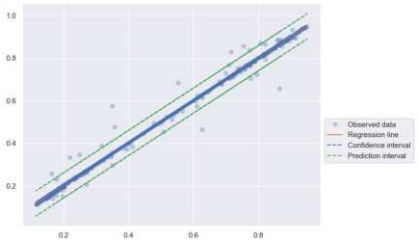
(iii) trestbps



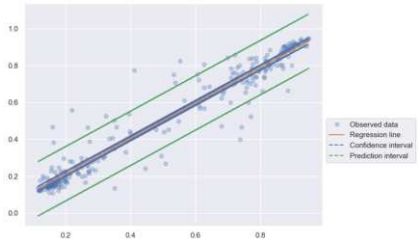
(iv) chol



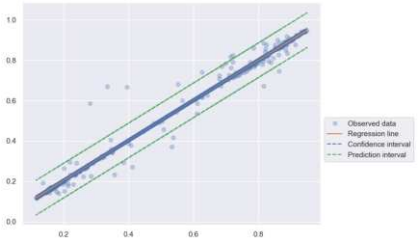
(v) fbs



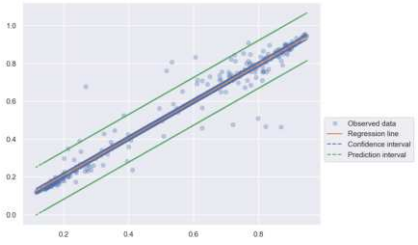
(vi) thalach



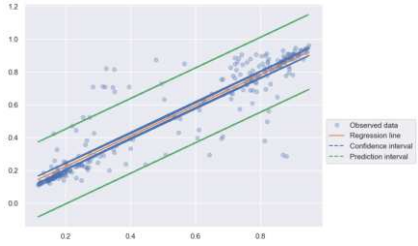
(vii) exang



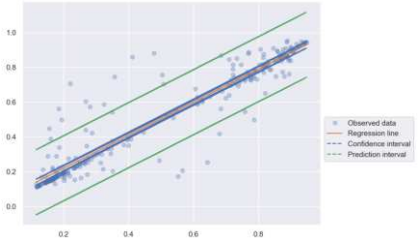
(viii) o_peak



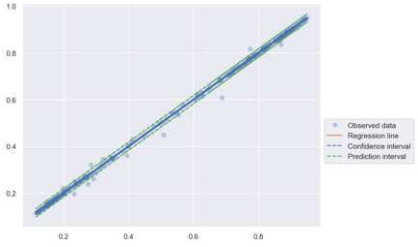
(ix) ca



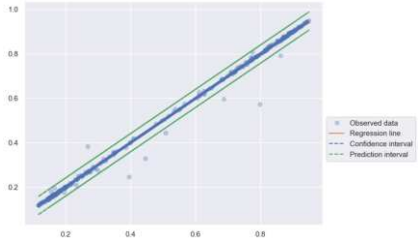
(x) thal



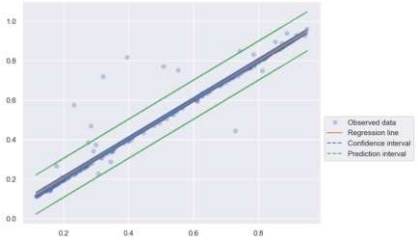
(xi) cp_2.0



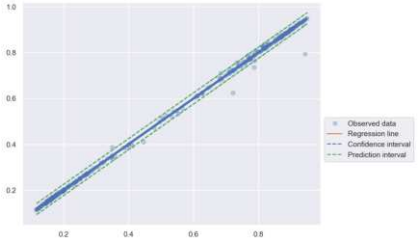
(xii) cp_3.0



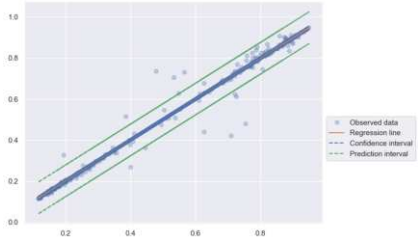
(xiii) cp_4.0

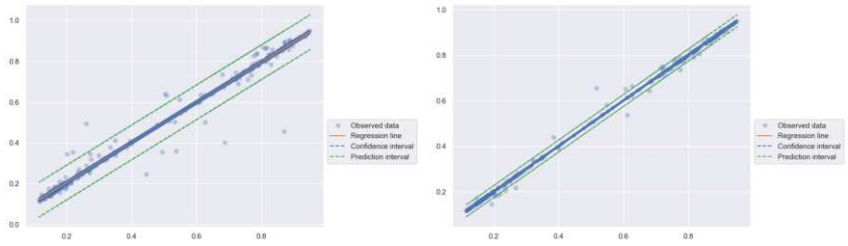


(xiv) restecg_1.0



(xv) restecg_2.0



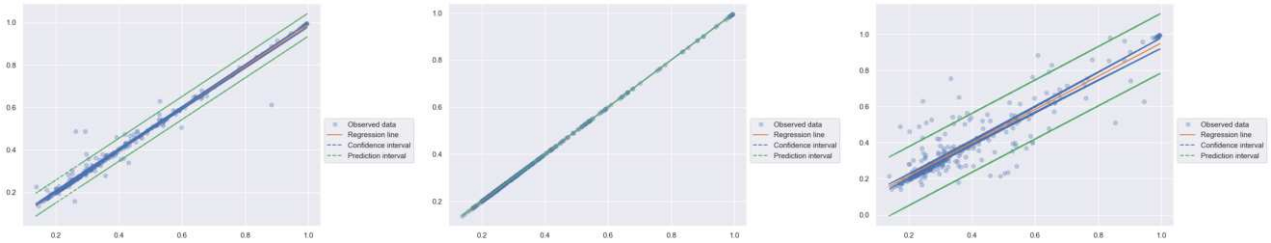


(xvi) slope_2.0 (xvii) slope_3.0

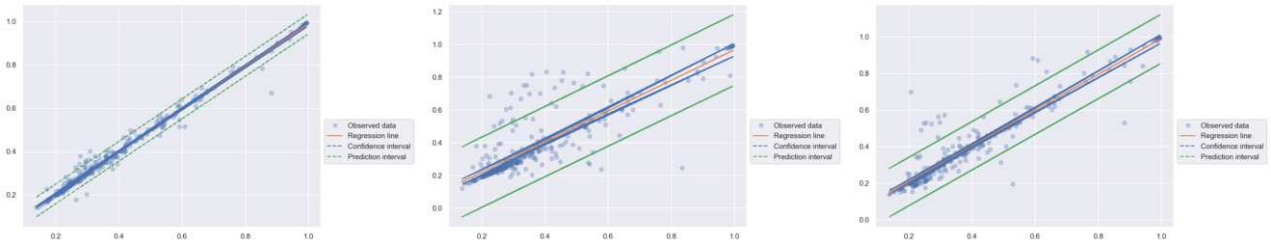
Source: Author's own elaboration

Figure 109 – OLS model fit for Ionosphere dataset - Iterative Imputation

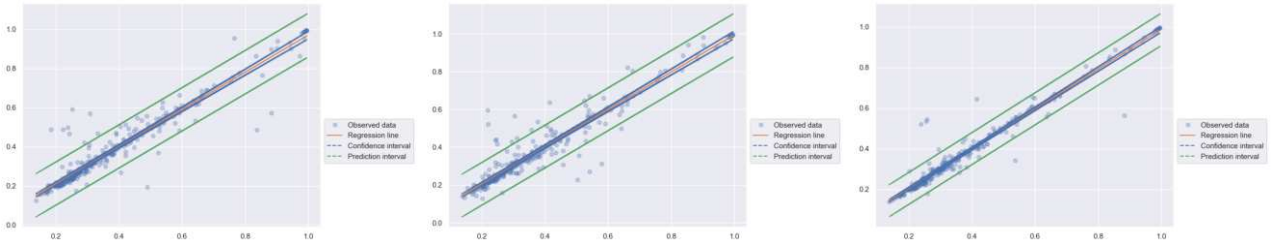
(i) col_0 (ii) col_1 (iii) col_2



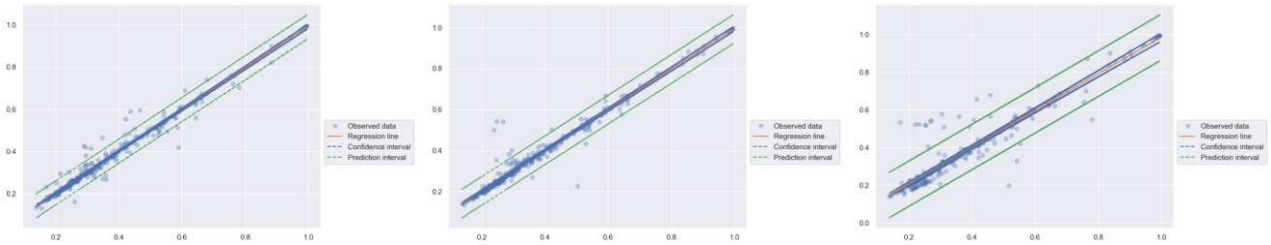
(iv) col_3 (v) col_4 (vi) col_5

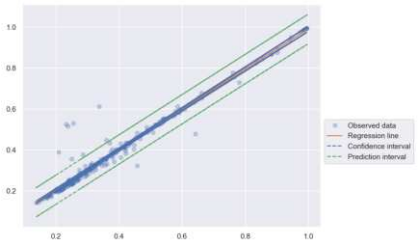


(vii) col_6 (viii) col_7 (ix) col_8

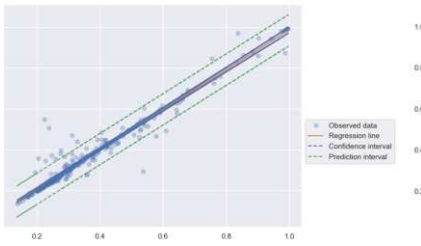


(x) col_9 (xi) col_10 (xii) col_11

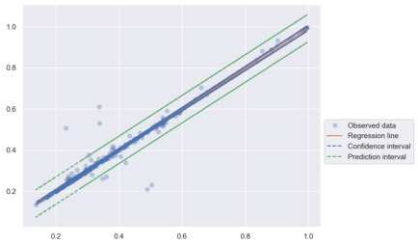




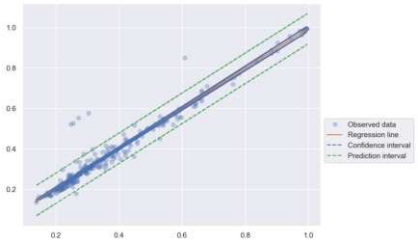
(xiii) col_12



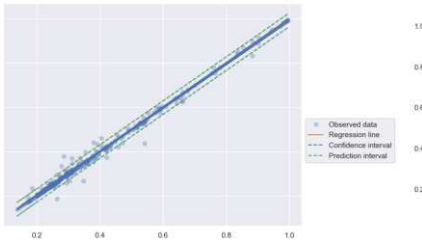
(xiv) col_13



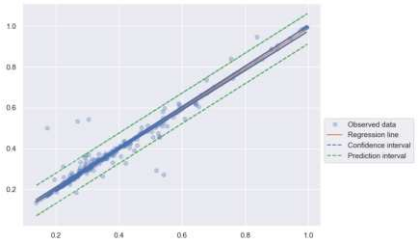
(xv) col_14



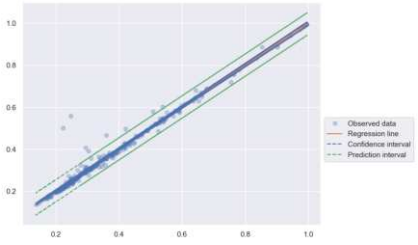
(xvi) col_15



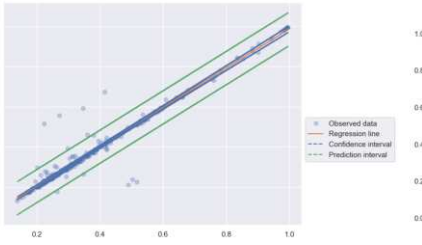
(xvii) col_16



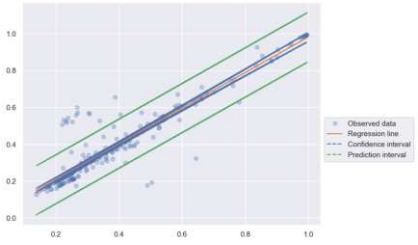
(xviii) col_17



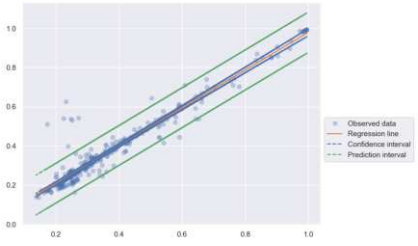
(xix) col_18



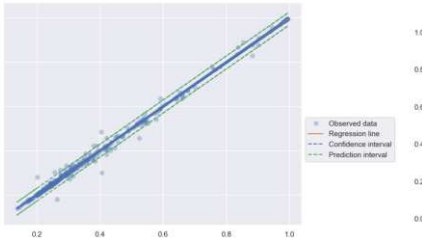
(xx) col_19



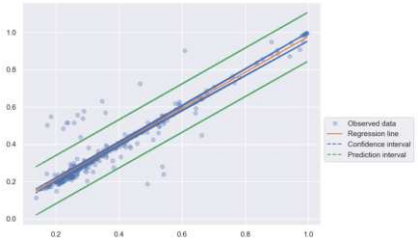
(xxi) col_20



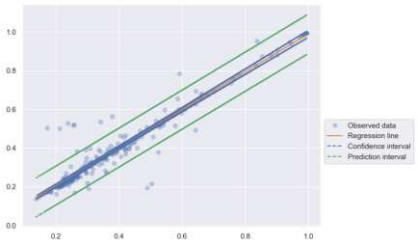
(xxii) col_21



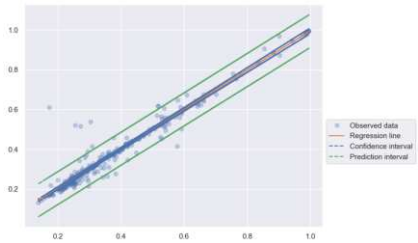
(xxiii) col_22



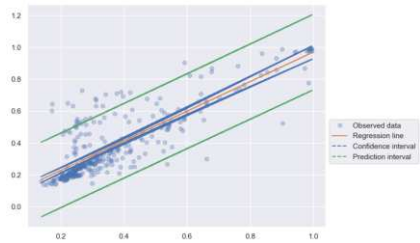
(xxiv) col_23



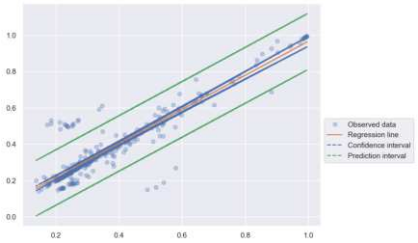
(xxv) col_24



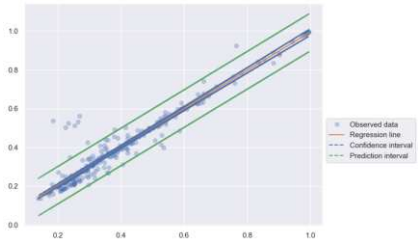
(xxvi) col_25



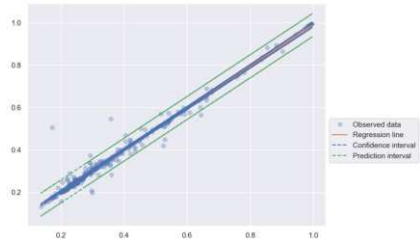
(xxvii) col_26



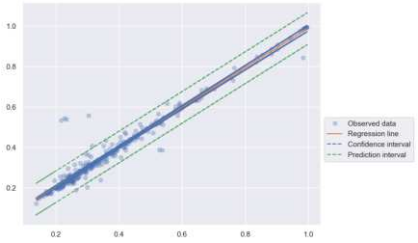
(xxviii) col_27



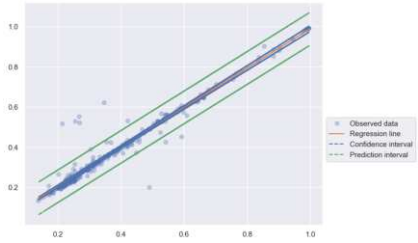
(xxix) col_28



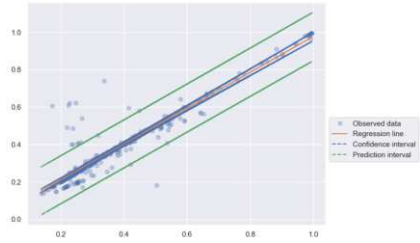
(xxx) col_29



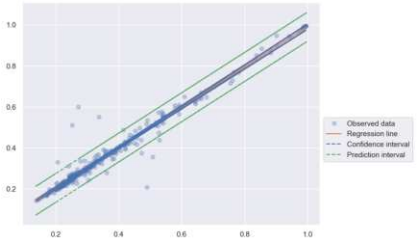
(xxxi) col_30



(xxxii) col_31



(xxxiii) col_32

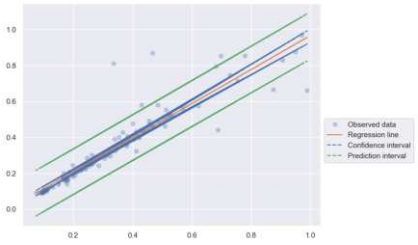


(xxxiv) col_33

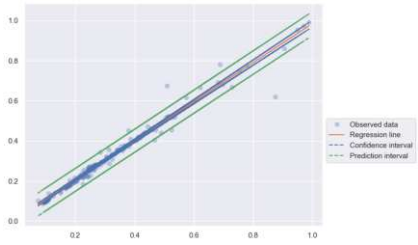
Source: Author's own elaboration

Figure 108 – OLS model fit for Wine dataset - Iterative Imputation

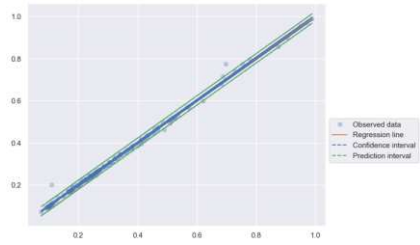
(i) alc



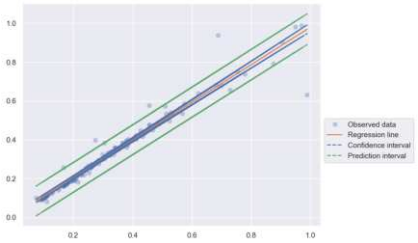
(ii) m_ac



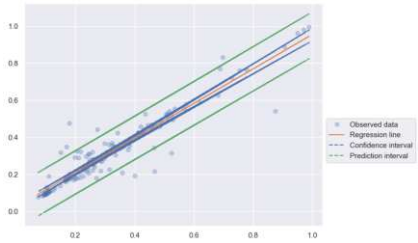
(iii) ash



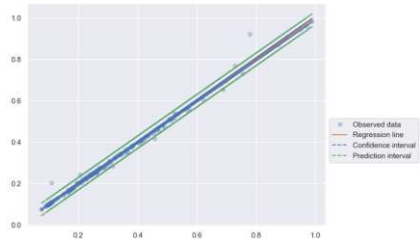
(iv) a_ash



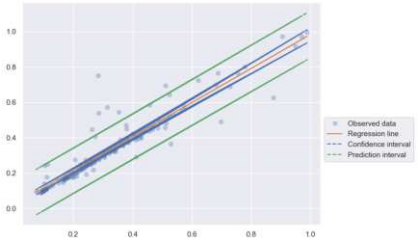
(v) mag



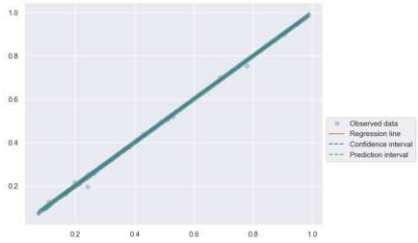
(vi) t_phe



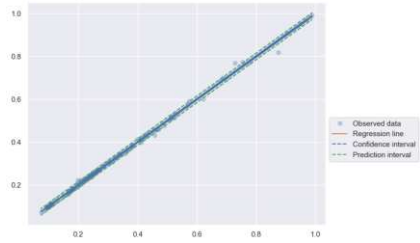
(vii) flav



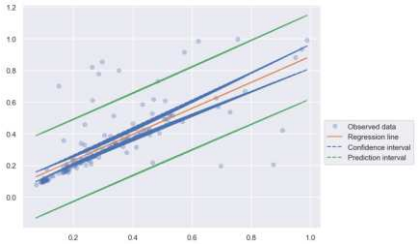
(viii) nf_ph



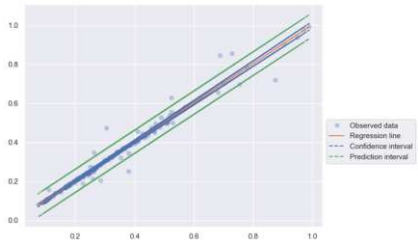
(ix) proa



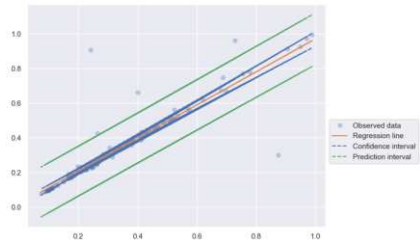
(x) col_i

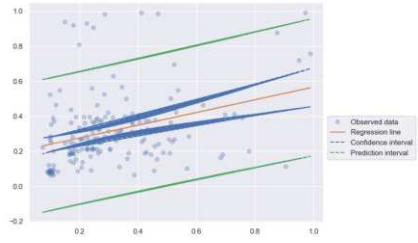


(xi) hue



(xii) od_dw



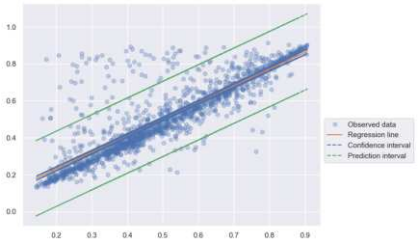


(xiii) prol

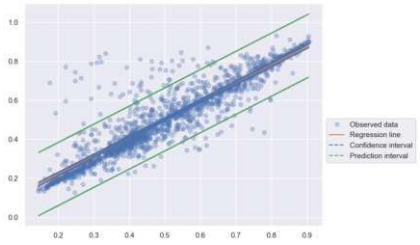
Source: Author's own elaboration

Figure 108 – OLS model fit for Yeast dataset - Iterative Imputation

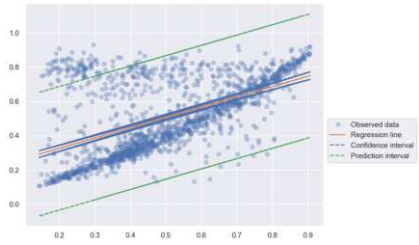
(i) mcg



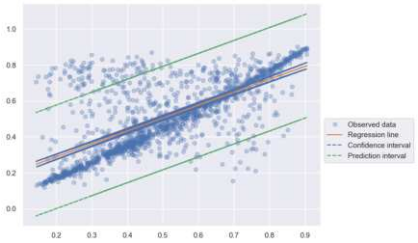
(ii) gvh



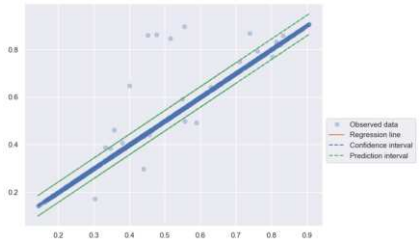
(iii) alm



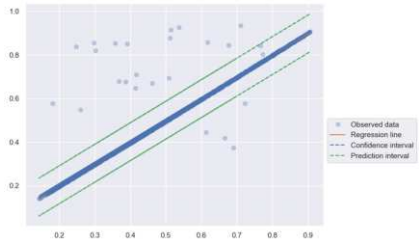
(iv) mit



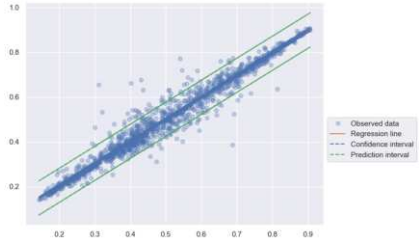
(v) erl



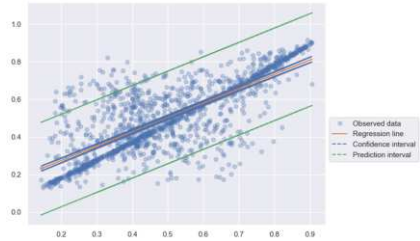
(vi) pox



(vii) vac



(viii) nuc



Source: Author's own elaboration