UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
PROGRAMA DE PÓS-GRADUAÇÃO EM ESTATÍSTICA

JONATHA SOUSA PIMENTEL

Aceleração da estimação do modelo de máquinas de vetores de suporte para bases de dados massivos: Uma nova abordagem através da fusão de SVMs fracos e esféricos.

Recife

2023

JONATHA SOUSA PIMENTEL

Aceleração da estimação do modelo de máquinas de vetores de suporte para bases de dados massivos: Uma nova abordagem através da fusão de SVMs fracos e esféricos.

Dissertação apresentada ao Programa de Pós-Graduação em Estatística da Universidade Federal de Pernambuco, como requisito final para a obtenção do título de Mestre em 10 de fevereiro de 2023.

Área de Concentração: Estatística Aplicada

Orientador (a): Raydonal Ospina Martínez

Coorientador (a): Anderson Ara

Recife

2023

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

P644a Pimentel, Jonatha Sousa
Aceleração da estimação do modelo de máquinas de vetores de suporte para bases de dados massivos: uma nova abordagem através da fusão de SVMs fracos e esféricos / Jonatha Sousa Pimentel. – 2023.
96 f.: fig., tab.

Orientador: Raydonal Ospina Martínez.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CCEN, Estatística, Recife, 2023.
Inclui referências e apêndices.

1. Estatística aplicada. 2. Dados massivos. I. Martínez, Raydonal Ospina (orientador). II. Título.

310 CDD (23. ed.) UFPE - CCEN 2023-54

JONATHA SOUSA PIMENTEL

ACELERAÇÃO DA ESTIMAÇÃO DO MODELO DE MÁQUINAS DE VETORES DE SUPORTE PARA BASES DE DADOS MASSIVOS: UMA NOVA ABORDAGEM ATRAVÉS DA FUSÃO DE SVMS FRACOS E ESFÉRICOS

Dissertação apresentada ao Programa de Pós-Graduação em Estatística da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Estatística.

Aprovada em: 10 de fevereiro de 2023.

BANCA EXAMINADORA

_____

Prof. Dr. Anderson Ara - (coorientador)
Presidente - UFPR

_____

Prof. Dr. Leandro Chaves Rêgo
(Titular Interno) UFPE

_____

Prof. Telmo de Menezes e Silva Filho
(Titular Externo)
Department of Engineering Mathematics/Bristol University

Dedico esta dissertação a minha mãe que lutou desde sua infância para obter uma educação de qualidade, e me permitiu assim a chance de hoje me tornar mestre, e, aos meus tios José Wilson e Idália que me deram moradia e apoio nos momentos de dúvida e desistência durante o meu primeiro passo no mundo acadêmico, o que permitiu chegar até onde cheguei.

AGRADECIMENTOS

Gostaria de agradecer aos meus familiares por aguentar os momentos de surto durante os 2 anos de desenvolvimento deste trabalho. Aos amigos que ajudaram a aliviar durante alguns poucos períodos de descanso do trabalho. Aos meus colegas que me ajudaram durante o primeiro ano dedicado principalmente as disciplinas, e a minha namorada Ludmila por ter sido a principal a aguentar todas as minhas reclamações diárias. Não posso deixar de citar meus orientadores Professor Anderson Ara e Professor Raydonal Ospina, que me trouxeram para a UFPE e permitiram iniciar e finalizar este projeto. Devo agradecimentos também aos professores Paulo Canas e Rodrigo Bulhões (ex-professores da UFBA e hoje colegas de trabalho) que durante as várias reuniões também me auxiliariam, cada um do seu jeito, no desenvolvimento deste trabalho. E, não posso deixar de agradecer também aos meus colegas 'SVMers', que debateram por diversas vezes sobre temas próximos ao por mim pesquisado durante estes dois anos, e principalmente na pessoa de Ivalbert, por compartilhar alguns dos bancos de dados utilizados durante as simulações.

RESUMO

A capacidade de geração de dados nos últimos anos, atingiu níveis nunca antes vistos. Mais dados em apenas dois anos do que em 5 mil anos de história, aumento em 50 vezes do volume gerado em um período de 10 anos, termos como *Big Data*, *Machine Learning* e Inteligência Artificial cada vez mais comumente vistos e falados. Provocado por essa mudança, é necessária a utilização de técnicas capazes de analisar tal quantidade de dados, sejam elas estatísticas ou de aprendizado de máquina, onde se destaca a interseção entre as duas áreas. Dentre as diversas técnicas existentes e de destaque, a técnica a ser aqui trabalhada são as máquinas de vetores de suporte (SVM - do inglês *Support Vector Machine*), introduzida por Vapnik em 1995, que busca a separação e classificação de observações através da construção de hiperplano ótimo. Apesar do grande uso, os modelos SVM sofrem para serem estimados em bases de dados massivas, sendo necessária a utilização de técnicas para enfrentar tal problema. Frente a este problema, neste trabalho serão implementadas as técnicas *Sphere SVM*, *Speed Up SVM* para a classificação, além do modelo proposto *Fusion SVM*, para a utilização em dados massivos. Os resultados obtidos ao comparar os modelos para diferentes bases de dados simuladas e reais, apresentam modelos que permitem a manutenção da capacidade preditiva do modelo SVM, bem como a sua estimação em um décimo do tempo do modelo completo. Além disso, para bases de dados cada vez maiores, é possível observar o melhor desempenho do modelo Fusion SVM sobre os demais.

Palavras-chaves: máquinas de vetores de suporte; dados massivos; *coresets*; SVMs fracos; ganho de tempo computacional.

ABSTRACT

The data generation capacity in recent years has reached levels never seen before. More data in just two years than in 5,000 years of history, a 50-fold increase in the volume generated over a 10-year period, terms like Big Data, Machine Learning and Artificial Intelligence increasingly commonly seen and spoken. Caused by this change, it is necessary to use techniques capable of analyzing such an amount of data, whether statistics or machine learning, where the intersection between the two areas stands out. Among the various existing and prominent techniques, the technique to be worked on here is the support vector machines (SVM), introduced by Vapnik in 1995, which seeks to separate and classify observations through the construction of an optimal hyperplane. Despite the wide use, the SVM models suffer to be estimated in massive databases, requiring the use of techniques to face this problem. Faced with this problem, this work will implement the techniques Sphere SVM, Speed Up SVM for classification, in addition to the proposed model Fusion SVM, for use in massive data. The results obtained when comparing the models for different simulated and real databases, present models that allow the maintenance of the predictive capacity of the SVM model, as well as its estimation in one tenth of the time of the complete model. In addition, for increasingly larger databases, it is possible to observe the better performance of the Fusion SVM model over the others.

Keywords: support vector machine; massive data; coresets; weak SVMs; computational time gain.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\mathbb{R}$	Conjunto dos números reais
γ	Ângulo entre dois vetores
$P_{\bar{a}}$	Projeção de $\bar{a}$ sobre $\bar{b}$
$\bar{x}$	Vetor de observações
b	Viés (bias)
$\bar{w}$	Vetor de pesos
$\bar{\alpha}$	Multiplicadores de Lagrange
x_+	Observações acima do hiperplano
x_-	Observações abaixo do hiperplano
m	Distância entre as margens
ε	Variáveis de folga
C	Variável de custo
R	Espaço de entradas
F	Espaço de características
$\phi(.)$	Função de mapeamento
$k(\bar{x}, \bar{y})$	*Kernel* ou função *kernel*
δ_{ij}	Delta de Kronecker
S	Conjunto de observações ou pontos
r	Raio da esfera
c	Centro da esfera
Q	Conjunto *coreset*
ρ	Viés

$\tilde{k}(\bar{x}, \bar{y})$	Função *kernel* modificada
η	Variável de perturbação do modelo
X_r	Amostra dos dados
X	População
$\psi(.)$	Vetor violador
τ, β	Valores para atualização do centro do *Sphere SVM*
n	Tamanho da amostra
N	Tamanho da população
h	Tamanho do coreset

SUMÁRIO

1 INTRODUÇÃO

Em tempos de *Big Data*, termo recorrente nos últimos anos que se refere a grande quantidade de dados gerados atualmente e caracterizado pela velocidade de geração, volume e variedade de formatos, o avanço tecnológico visto nas últimas décadas provocaram uma geração de dados cada vez maior e mais rápida, nos levando então a chamada 'era dos dados', onde tratamos os dados como um novo petróleo, que precisa ser refinado e tratado para um melhor uso. Exemplo deste crescimento é o aumento em pelo menos 50 vezes do volume de dados gerados entre os anos de 2005 e 2015 (LU et al., 2021).

Frente a isso, surge dia após dia uma necessidade cada vez maior de se encontrar formas de coletar e armazenar essa elevada quantidade de dados. Como consequência, é preciso encontrar também formas de analisar estes dados, para que depois de refinados, possamos extrair valorosos *insights* que possam auxiliar na tomada de decisão.

Ao longo dos anos, as técnicas estatísticas vem cumprindo esta tarefa de transformar dados em informações, quantificando a incerteza e nos auxiliando na tomada de decisões. A vista disso, a ciência estatística desenvolveu e aprimorou técnicas e modelos durante anos, técnicas estas capazes de extrair com base nos poucos dados que se tinha disponível, as informações preciosas que se era necessário. Em determinadas áreas de estudo, conjunto de dados com algumas centenas ou com alguns milhares de observações era tudo o que se tinha disponível para a análise (BACK, 2001), sendo necessário assim técnicas especificas que possibilitassem resultados bons e confiáveis para tal quantidade de dados, que hoje podem ser considerados como escassos quando comparamos com as milhões de observações vistas atualmente.

Em consonância com a estatística, temos a classe de modelos de aprendizagem de máquina (ML - do inglês *Machine Learning*), que buscam produzir modelos capazes de aprender com os dados (IZBICKI; SANTOS, 2020), tendo seu principal foco na capacidade preditiva.

A partir do formato de aprendizado, as técnicas de aprendizado de máquina podem pertencer a um dos subgrupos, sendo os mais utilizados a aprendizagem supervisionada (dados rotulados que permitem ao modelo aprender com a resposta esperada e avaliar o quão errado está o modelo produzido) (KHAN; SHAH; BHAT, 2020; IWENDI et al., 2020; MAIA et al., 2020), aprendizagem não supervisionada (dados não rotulados que exigem do modelo a identificação de padrões que possam agrupar os dados sem possuir uma referência para comparação) (HASAN; ABDULAZEEZ, 2021; GLIELMO et al., 2021) e aprendizagem por reforço (onde o mo-

delo é livre para aprender tendo como única métrica de avaliação o *feedback* por meio de recompensas que indicam o quão bem está o processo de aprendizagem) (CHEN et al., 2021).

Dentro destes diversos campos de aprendizagem totalmente baseada em dados, existe então o desafio em encontrar, desenvolver e utilizar técnicas que possam ser aplicadas a estas grandes bases de dados (AL-JARRAH et al., 2015), de forma a resultar em respostas precisas, e dentro de um tempo aceitável para o prosseguimento da análise.

Pensando nisso, e também com o advento de tecnologias que permitem a construção de computadores cada vez mais rápidos e poderosos, hoje a utilização de GPUs e computação em paralelo para acelerar o treinamento de modelos tem se tornado cada vez mais comuns (LI et al., 2015; SINGH et al., 2021), e mesmo assim, ainda existem modelos que a depender da quantidade de dados ainda podem levar dias para seu ajuste.

Visto que apenas a melhora do *hardware* em alguns casos não é suficiente para se atingir o objetivo, volta-se a atenção para o desenvolvimento e melhora dos diversos métodos de aprendizado de máquina.

Dentre tantos métodos que existem e são utilizados diariamente, o método base a ser aqui abordado e utilizado será a Máquina de Vetores de Suporte (SVM - do inglês *Support Vector Machine*), técnica considerada 'nova' e que desperta interesse desde o início do seu desenvolvimento. Tal modelo trata-se de um algoritmo de aprendizado de máquina supervisionado proposto por VAPNIK (1995), fundamentado na teoria do aprendizado estatístico (VAPNIK, 1998), que em suma, busca construir hiperplanos ótimos para classificar rótulos - originalmente binários - bem como utiliza a kernealização para aumentar a flexibilidade do modelo (WANG et al., 2003). Em outras palavras, o SVM busca separar grupos de observações traçando um hiperplano entre elas, de forma a maximizar a distância entre as margens superior e inferior ao hiperplano.

Classificação de imagens (SHEYKHMOUSA et al., 2020), simulação de produtos manufaturados antes da sua fabricação através de análises virtuais do seu desempenho (MUTHUKRISHNAN et al., 2020), detecção e classificação de tumores (GURBINĂ; LASCU; LASCU, 2019), reconhecimento de emoções a partir da fala de um orador (BHAVAN et al., 2019) e análise de sentimentos a partir de *tweets* (STYAWATI; HENDRASTUTY; ISNAIN, 2021) são alguns dos exemplos de aplicações das máquinas de vetores de suporte, demonstrando sua ampla utilização nas mais diversas áreas de estudo.

Assim como muitos modelos de aprendizagem, o SVM possui um problema de ajuste quando tratamos de grandes bases de dados. Problema este, que está diretamente ligado a

forma de implementação do modelo, que ao realizar o uso dos multiplicadores de Lagrange em seu processo de busca pelos chamados vetores de suporte, torna o modelo com um custo computacional diretamente proporcional ao tamanho da base de dados utilizada para estimá-lo.

Por se tratar de um modelo com certas vantagens frente a outras técnicas de aprendizado de máquina, como por exemplo seu alto poder de generalização e forte fundamentação teórica, este problema tem atraído significativa atenção de diversos pesquisadores, que buscam resolvê-lo através da melhora da fase de treinamento ou com a extração de conjuntos de treinamentos reduzidos para o modelo SVM (NALEPA; KAWULOK, 2019).

A partir disso, diversos trabalhos foram desenvolvidos com temáticas diferentes, buscando a superação de tal dificuldade para o modelo, dentre eles, a técnica que será aqui apresentada chamada de *Sphere Support Vector Machine* proposta por (STRACK et al., 2013). O *Sphere SVM* é basicamente um novo algoritmo que propõe uma nova reformulação a técnica *Ball Vector Machine* (TSANG; KOCSOR; KWOK, 2007), que por sua vez é a versão mais simples do *Core Vector Machine* (TSANG et al., 2005), que ao formular o modelo SVM como um *Minimum Enclosing Ball* (MEB) e com a utilização de um subconjunto dos dados que recebe o nome de *coresets*, construiu uma abordagem diferente para o treinamento do modelo SVM para grandes bases de dados.

1.1 OBJETIVOS

Com base nisso, o objetivo principal deste trabalho é a implementação de técnicas capazes de estimar as máquinas de vetores de suporte para bases de dados massivos. Além disso, os objetivos específicos deste trabalho são: (i) Adaptação do modelo *Speed Up SVM* para classificação de observações. (ii) Implementação da técnica *Sphere Support Vector Machine* (STRACK et al., 2013), técnica existente e utilizada para dados massivos. (iii) Construção de um novo modelo que também acelere a estimação do SVM através da junção dos *Weak SVMs* provenientes do *Speed Up SVM* e a construção de *coresets* provenientes do *Sphere SVM*. (iv) Por fim, realizar a comparação de todos os modelos, propostos, implementados e existentes com o modelo SVM completo.

O restante deste trabalho é descrito da seguinte forma. Inicialmente são descritos alguns conceitos mais básicos, como também é apresentada a teoria por trás do modelo de Máquinas de Vetores de Suporte e a ideia do problema de geometria computacional *Minimum Enclosing Ball* que será fundamental para as técnicas apresentadas a seguir. Na sequência, é construída

uma revisão de literatura de técnicas utilizadas para enfrentar o problema a ser abordado, sendo também apresentadas as técnicas a serem utilizadas neste trabalho. Então, serão apresentados os resultados obtidos para as técnicas utilizadas através de simulações e conjuntos de dados reais, finalizando o trabalho com um discussão dos resultados e indicações de possíveis trabalhos futuros.

2 CONCEITOS BÁSICOS

Neste capítulo serão apresentados alguns conceitos iniciais que serão de extrema importância para o entendimento dos métodos a serem discutidos e implementados no próximo capitulo.

2.1 PRELIMINARES

Como pontapé inicial, com o objetivo de facilitar a compreensão dos conceitos que serão apresentados a partir da seção 2.2, será realizada uma breve revisão de alguns conceitos mais básicos de matemática e geometria.

2.1.1 Vetores

Um segmento de reta com comprimento e direção é chamado de vetor, sendo um vetor de comprimento 1 chamado de vetor unitário. Dado os componentes de mesma posição de um vetor em $\mathbb{R}^n$, seja $\bar{p} = (p_1, \cdots, p_n)$ um vetor, podemos calcular seu tamanho ou norma, que pode ser escrito como $|\bar{p}|$ e definido por:

$$|\bar{p}| = \sqrt{p_1^2 + \cdots + p_n^2}.$$

Sejam $\bar{a} = (a_1, \cdots, a_n)$, $\bar{b} = (b_1, \cdots, b_n)$ vetores em $\mathbb{R}^n$, então $\bar{a}$ e $\bar{b}$ são iguais se, e somente se os componentes correspondentes de cada vetor são iguais. Formalmente, temos:

$$\bar{a} = \bar{b}, \text{ se } a_1 = b_1, \cdots, a_n = b_n.$$

Sejam $\bar{a} = (a_1, \cdots, a_n)$, $\bar{b} = (b_1, \cdots, b_n)$ e $\bar{c} = (c_1, \cdots, c_n)$ vetores em $\mathbb{R}^n$, então definimos a adição de vetores como:

$$\bar{c} = \bar{a} + \bar{b}$$

tal que, $c_1 = a_1 + b_1$, $\cdots$, $c_n = a_n + b_n$.

O vetor $\bar{0} = (0, \cdots, 0)$ é chamada de vetor nulo, possuindo n elementos, tem tamanho 0 e pode ser considerado o ponto de origem de um sistema de coordenadas n-dimensional. Por fim, temos o vetor oposto $(-\bar{a})$, que é o vetor $\bar{a}$ com todos os componentes negativos:

$$-\bar{a} = -(a_1, \cdots, a_n) = (-a_1, \cdots, -a_n).$$

Nas tabelas 1 e 2 são apresentados alguns axiomas de soma de vetores e de multiplicação de vetores por um escalar. Um conjunto V de vetores em $\mathbb{R}^n$ é chamado de espaço vetorial real se os axiomas de adição de vetores e multiplicação de vetores por um escalar apresentados nas Tabelas 1 e 2 são satisfeitos.

Tabela 1 – Axiomas de adição de vetores

Comutatividade	$\bar{a} + \bar{b} = \bar{b} + \bar{a}$
Associatividade	$(\bar{a} + \bar{b}) + \bar{c} = \bar{a} + (\bar{b} + \bar{c})$
Identidade	$\bar{a} + \bar{0} = \bar{a}$
Reciprocidade	$\bar{a} + (-\bar{a}) = \bar{0}$

Fonte: Autor.

Tabela 2 – Axiomas de multiplicação de um vetor por escalar

Distributiva I	$q(\bar{a} + \bar{b}) = q\bar{a} + q\bar{b}$
Distributiva II	$(p + q)\bar{a} = p\bar{a} + q\bar{a}$
Associativa	$(pq)\bar{a} = p(q\bar{a})$
Identidade	$1\bar{a} = \bar{a}$

Fonte: Autor.

Dado dois vetores $\bar{a} = (a_1, \cdots, a_n)$ e $\bar{b} = (b_1, \cdots, b_n)$ em um espaço vetorial real n-dimensional $\mathbb{R}^n$, defini-se o produto escalar $\bar{a} \cdot \bar{b}$ como:

$$\bar{a} \cdot \bar{b} = a_1 b_1 + \cdots + a_n b_n = |\bar{a}||\bar{b}| cos(\gamma)$$

onde $|\bar{a}|$ e $|\bar{b}|$ são as normas dos vetores $\bar{a}$ e $\bar{b}$, respectivamente, e γ é o ângulo entre os dois vetores. Dois vetores $\bar{a}$ e $\bar{b}$ com $|\bar{a}| > 0$ e $|\bar{b}| > 0$ são ditos ortogonais se formam um ângulo de $90°$ entre si. Pela definição acima, temos que se $\bar{a}$ e $\bar{b}$ são ortogonais $\bar{a} \cdot \bar{b} = 0$.

Na Tabela 3, são vistas algumas identidades algébricas para o produto escalar.

Tabela 3 – Identidades algébricas do produto escalar

Linearidade	$(p\bar{a} + q\bar{b}) \cdot \bar{c} = p\bar{a} \cdot \bar{c} + q\bar{b} \cdot \bar{c}$
Simetria	$\bar{a} \cdot \bar{b} = \bar{b} \cdot \bar{a}$
Não negatividade	$\bar{a} \cdot \bar{a} \geq 0$
Não degeneração	$\bar{a} \cdot \bar{a} = 0, \text{ se } \bar{a} = 0$

Fonte: Autor.

Além das identidades apresentadas na Tabela 3, temos as seguintes identidades que se mostram bastante úteis. Começando pela relação entre a norma de um vetor e o produto

escalar.

$$|\bar{a}| = \sqrt{\bar{a} \cdot \bar{a}}.$$

Na sequência temos uma derivação para o $cos(\gamma)$ onde:

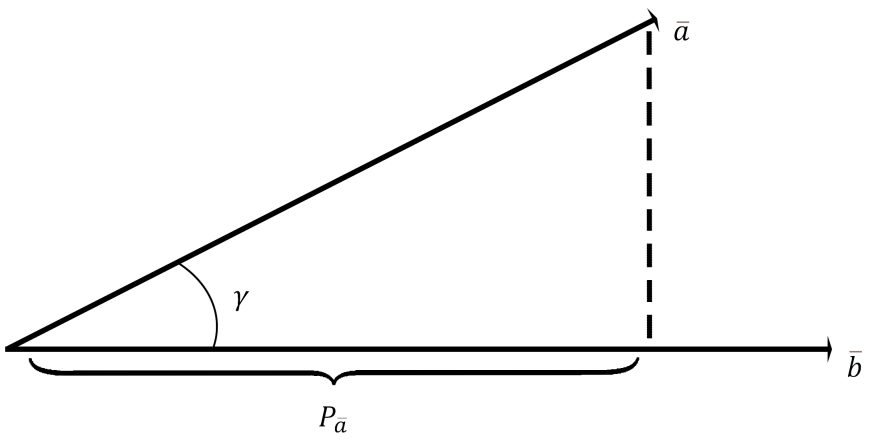
$$cos(\gamma) = \frac{\bar{a} \cdot \bar{b}}{|\bar{a}||\bar{b}|}.$$

Seja $\bar{a}$ e $\bar{b}$ vetores em $\mathbb{R}^n$ que formam um ângulo γ entre eles, dizemos que $P_{\bar{a}}$ é a projeção de $\bar{a}$ na direção de $\bar{b}$, de forma que:

$$p_{\bar{a}} = |\bar{a}|cos(\gamma) = \frac{|\bar{a}|(\bar{a} \cdot \bar{b})}{|\bar{a}||\bar{b}|} = \frac{\bar{a} \cdot \bar{b}}{|\bar{b}|}.$$

Na Figura 1, temos a representação geométrica da projeção do vetor $\bar{a}$ na direção de $\bar{b}$.

Figura 1 – Representação geométrica da projeção do vetor $\bar{a}$ na direção de $\bar{b}$.



Fonte: Adaptado de HAMEL, 2011.

2.1.2 Retas, Planos e Hiperplanos

Com o objetivo de separar dois grupos de observações, a forma mais simples e direta de se fazer isso é traçar uma reta, um plano ou hiperplano (a depender da dimensão dos dados) para separar estes grupos. Com isso em mente, seja uma função linear dada por:

$$f(x) = y = -mx$$

com $x, y, m \in \mathbb{R}$, esta função é interpretada como uma reta com um conjunto de pontos $(x, y) \in \mathbb{R}^2$, onde:

$$f(x) = mx + y = 0.$$

Estando em um espaço bidimensional, definindo $w_1 = m$ e $w_2 = 1$, podemos escrever:

$$f(x) = w_1 x + w_2 y = 0$$

que pode ser visto como o produto escalar entre os vetores $\bar{w} = (w_1, w_2)$ e $\bar{x} = (x, y)$:

$$f(x) = w_1 x + w_2 y = \bar{w} \cdot \bar{x} = 0.$$

Como o produto escalar entre os dois vetores é zero, implica que $\bar{w}$ e $\bar{x}$ são ortogonais, sendo $\bar{x}$ interpretado como o vetor de posição para qualquer ponto (x, y) pertencente a reta $mx + y = 0$, e o $\bar{w}$ chamado de vetor normal. Como esta reta passa pela origem, o vetor de posição $\bar{x}$ é paralelo a reta para qualquer ponto pertencente a ela. Como $\bar{w}$ é ortogonal a $\bar{x}$, que é paralelo a reta, então $\bar{w}$ é ortogonal a reta.

Observando um caso mais geral, ao inserir uma variável de viés (*bias*) $b \in \mathbb{R}$, a reta será escrita como:

$$f(x) = y = -mx + b.$$

De modo análogo para o caso em que a reta passa pela origem, pode ser mostrado que:

$$\bar{w} \cdot \bar{x} = b$$

com $\bar{w} = (m, 1)$ e $\bar{x} = (x, y)$. Observe que o vetor normal $\bar{w}$ e o vetor de posição $\bar{x}$ não são mais ortogonais, dado que a reta não passa mais pela origem, porém, $\bar{w}$ segue ortogonal a reta. Através do produto escalar entre vetores $\bar{w} \cdot \bar{x} = b$, que permite especificar a equação de uma reta qualquer, podemos levar os conceitos vistos anteriormente para um espaço tridimensional. Sejam $\bar{w} = (w_1, w_2, w_3)$ e $\bar{x} = (x, y, z)$, então:

$$\bar{w} \cdot \bar{x} = b$$

descreve um plano com $\bar{w}$ vetor normal ao plano e $\bar{x}$ o vetor de posição dos pontos no plano. Da mesma forma, se considerarmos $\bar{w} = (w_1, \cdots, w_n)$ e $\bar{x} = (x_1, \cdots, x_n)$, a equação $\bar{w} \cdot \bar{x} = b$ define um hiperplano no espaço n-dimensional.

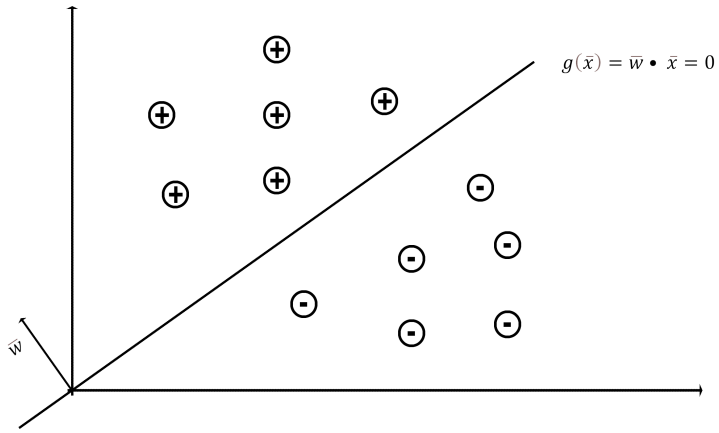
2.1.3 Superfícies de Decisão Linear

Considerando um espaço bidimensional, para um conjunto de dados linearmente separável, ou seja, conseguimos separar as observações perfeitamente traçando uma reta entre elas, imaginemos que esta reta passa pela origem, sendo definida por:

$$g(\bar{x}) = \bar{w} \cdot \bar{x} = 0$$

com $\bar{x}, \bar{w} \in \mathbb{R}^2$, sendo então chamada de superfície de decisão (Vide Figura 2). Uma vez que tenhamos uma superfície de decisão, ela pode ser utilizada para classificar qualquer ponto no espaço, informando se este ponto esta abaixo ou acima da superfície.

Figura 2 – Superfície de decisão simples separando dois grupos de observações.



Fonte: Adaptado de HAMEL, 2011.

Considerando um ponto $\bar{a} \in \mathbb{R}^2$, podemos classifica-lo como uma observação positiva ($+$, acima da superfície) ou negativa ($-$, abaixo da superfície) de forma direta, utilizando o vetor normal $\bar{w}$. Para isso, basta calcular:

$$g(\bar{a}) = \bar{w} \cdot \bar{a} = |\bar{w}||\bar{a}|cos(\gamma) = k$$

que terá um valor positivo para k se $\bar{a}$ estiver acima da superfície de decisão, o que implica em $\gamma \leq 90°$ e um valor negativo para k se $\bar{a}$ estiver abaixo da superfície de decisão, o que implica em $\gamma > 90°$, onde γ representa o ângulo entre os vetores $\bar{w}$ e $\bar{a}$, como é apresentado na Figura 3. Com isso, definimos nossa função de decisão como:

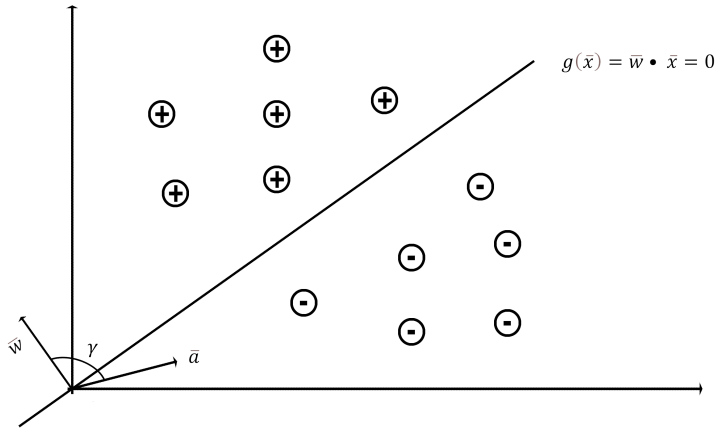
$$f(\bar{x}) = \begin{cases} +1, & \text{se } g(\bar{x}) \geq 0 \\ -1, & \text{se } g(\bar{x}) < 0. \end{cases}$$

para todo $\bar{x} \in \mathbb{R}^2$.

Ainda em $\mathbb{R}^2$, ao relaxarmos um pouco nossa superfície de decisão com a adição de um viés, nossa superfície não irá mais passar pela origem e será representada como:

$$g(\bar{x}) = \bar{w} \cdot \bar{x} = b$$

Figura 3 – Classificação de um novo ponto $\bar{a}$ utilizando uma superfície de decisão.



Fonte: Adaptado de HAMEL, 2011.

sendo b o viés. No caso em que a superfície de decisão passa pela origem, podíamos classificar uma nova observação $\bar{a}$ calculando o produto escalar entre o vetor normal $\bar{w}$ e a nova observação, pois ambos partiam da origem, o que não mais ocorre com a adição do viés b. Porém, este fato pode ser recriado através da subtração de vetores.

Seja um ponto $\bar{c}$ pertencente a superfície de decisão, tal que

$$g(\bar{c}) = \bar{w} \cdot \bar{c} = b$$

nosso vetor $\bar{a}$ pode ser escrito como

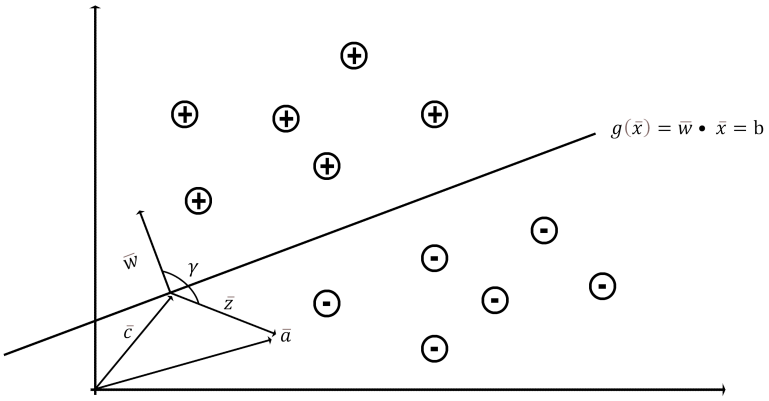
$$\bar{a} = \bar{c} + \bar{z} \rightarrow \bar{z} = \bar{a} - \bar{c}.$$

Com esta modificação teremos o vetor $\bar{z}$ como nosso vetor de posição que partirá até $\bar{a}$ do mesmo ponto que $\bar{w}$ (Vide Figura 4), podendo assim classificar a nova observação novamente através do simples cálculo de um produto escalar.

$$\bar{w} \cdot \bar{z} = |\bar{w}||\bar{z}|cos(\gamma) = k \tag{2.1}$$

Como anteriormente visto, k será um valor positivo caso $\bar{a}$ esteja acima da superfície de decisão, o que implica em $\gamma \leq 90°$, e k será um valor negativo se $\bar{a}$ estiver abaixo da superfície de decisão, o que implica em $\gamma > 90°$, com γ sendo o ângulo entre $\bar{w}$ e $\bar{z}$.

Figura 4 – Classificação de um novo ponto $\bar{a}$ utilizando uma superfície de decisão para o caso em que a superfície de decisão não passa pela origem.



Fonte: Adaptado de HAMEL, 2011.

A equação 2.1 pode ser melhor escrita ao substituirmos $\bar{z} = \bar{a} - \bar{c}$, de forma que:

$$g(\bar{z}) = \bar{w} \cdot \bar{z}$$
$$g(\bar{z}) = \bar{w} \cdot (\bar{a} - \bar{c})$$
$$g(\bar{z}) = \bar{w} \cdot \bar{a} - \bar{w} \cdot \bar{c}$$
$$g(\bar{z}) = \bar{w} \cdot \bar{a} - b$$
$$g(\bar{z}) = g(\bar{a}) - b.$$

Assim, nossa função de decisão é escrita como:

$$f(\bar{x}) = \begin{cases} +1, & \text{se } g(\bar{x}) - b \geq 0 \\ -1, & \text{se } g(\bar{x}) - b < 0. \end{cases}$$

As funções de decisão foram desenvolvidas para o caso bidimensional, entretanto, nada impede que a mesma estrutura seja desenvolvida para superfícies de decisão em espaços com dimensões superiores. Desse modo, para uma superfície de decisão $g(\bar{x}) = \bar{w} \cdot \bar{x} = b$, em um espaço n-dimensional, a função de decisão é dada por:

$$f(\bar{x}) = \mathrm{sinal}(\bar{w} \cdot \bar{x} - b)$$

com $\bar{w}, \bar{x} \in \mathbb{R}^n$ e $b \in \mathbb{R}$, de forma que a função sinal é dada por:

$$sinal(k) = \begin{cases} +1, & \text{se } k \geq 0 \\ -1, & \text{se } k < 0. \end{cases}$$

para todo $k \in \mathbb{R}$.

2.1.4 Multiplicadores de Lagrange

O método dos Multiplicadores de Lagrange consiste numa técnica bastante utilizada para encontrar máximos e mínimos de funções, sujeito a um conjunto de restrições.

Considere uma situação em que queremos minimizar uma função $f(\bar{x})$ sujeita a um conjunto de restrições $g_i(\bar{x}) \geq 0,\ \forall \bar{x} \in \mathbb{R}^n$ com $i = 1, \cdots, L$. Este problema de otimização pode ser escrito como:

$$\min_{\bar{x}} f(\bar{x})$$

e é conhecido como problema de otimização *primal*. Com isso, podemos definir uma função chamada Lagrangiano que é denotada por $L(\bar{\alpha}, \bar{x})$. Essa função combina a função $f(\bar{x})$ com um conjunto de restrições $g_i(\bar{x})$, sendo formalmente escrita como:

$$L(\bar{\alpha}, \bar{x}) = f(\bar{x}) - \sum_{i=1}^{L} \alpha_i g_i(\bar{x})$$

onde $\alpha_1, \cdots, \alpha_L$ são chamados multiplicadores de Lagrange, existindo um multiplicador para cada restrição $g_i(\bar{x})$.

Com base na função Lagrangiana $L(\bar{\alpha}, \bar{x})$ é definido então o problema de otimização Lagrangiano duplo, que apresenta os dois operadores opostos, mínimo e máximo, e tem como objetivo minimizar por uma escolha de $\bar{x}$, chamada variável *primal*, e maximizar por uma escolha de $\bar{\alpha}$, chamada de variável *dual*. O problema é dado por:

$$\max_{\bar{\alpha}} \min_{\bar{x}} L(\bar{\alpha}, \bar{x}) = \max_{\bar{\alpha}} \min_{\bar{x}} \left(f(\bar{x}) - \sum_{i=1}^{L} \alpha_i g_i(\bar{x}) \right)$$

com $\alpha_i \geq 0$ para $i = 1, \cdots, L$ e $\bar{x} \in \mathbb{R}^n$.

Por apresentar os dois operadores, o problema de otimização Lagrangiano duplo pode ser visto como um operador que retorna uma função parcialmente avaliada, devendo ser otimizada por outro operador. Dessa forma, ao fixarmos a variável *primal* $\bar{x}$ para o valor $\bar{x}^*$, temos:

$$\max_{\bar{\alpha}} L(\bar{\alpha}, \bar{x}^*) = \max_{\bar{\alpha}} \left(f(\bar{x}^*) - \sum_{i=1}^{L} \alpha_i g_i(\bar{x}^*) \right)$$

e ao fixarmos a variável *dual* $\bar{\alpha}$ para o valor $\bar{\alpha}^*$, temos:

$$\min_{\bar{x}} L(\bar{\alpha}^*, \bar{x}) = \min_{\bar{x}} \left(f(\bar{x}) - \sum_{i=1}^{L} \alpha_i^* g_i(\bar{x}) \right).$$

As soluções para o problema de otimização Lagrangiano duplo, são pontos que maximizam a função $L(\bar{\alpha}, \bar{x})$ em relação a $\bar{\alpha}$ e minimizam em relação a $\bar{x}$, o que implica que as soluções

são pontos de sela no gráfico de $L(\bar{\alpha}, \bar{x})$. Para o caso em que a função $f(\bar{x})$ é convexa e as restrições $g_i(\bar{x})$ são lineares, o ponto de sela é único.

Como o ponto de sela representa uma solução, $L(\bar{\alpha}, \bar{x})$ será mínimo com relação a $\bar{x}$ e a derivada parcial de $L(\bar{\alpha}, \bar{x})$ em relação a $\bar{x}$ deve ser 0, isto é:

$$\frac{\delta(L(\bar{\alpha}, \bar{x}))}{\delta \bar{x}} = \bar{0}.$$

Uma das propriedades mais importantes do problema de otimização Lagrangiano duplo é que sob certas condições, a sua solução é também solução para o problema de otimização *primal*. Com isso, sejam $\bar{\alpha}^*$ e $\bar{x}^*$ soluções para o Lagrangiano duplo, tal que:

$$\max_{\bar{\alpha}} \min_{\bar{x}} L(\bar{\alpha}, \bar{x}) = L(\bar{\alpha}^*, \bar{x}^*) = \left(f(\bar{x}^*) - \sum_{i=1}^{L} \alpha_i^* g_i(\bar{x}^*) \right),$$

então, $\bar{x}$ será solução para o problema de otimização *primal* se, e somente se as seguintes condições forem respeitadas:

$$\frac{\delta L(\bar{\alpha}^*, \bar{x}^*)}{\delta \bar{x}^*} = \bar{0}$$

$$\alpha_i^* g_i(\bar{x}^*) = 0$$

$$g_i(\bar{x}^*) \geq 0$$

$$\alpha_i^* \geq 0.$$

Essas condições são conjuntamente conhecidas como condições de *Karush-Kuhn-Tucker* ou simplesmente condições KKT.

Com todas estas definições e conceitos apresentados, podemos construir o modelo de máquina de vetores de suporte, e todas as técnicas derivadas dele de forma que sua compreensão seja garantida.

2.2 SUPPORT VECTOR MACHINE

A forma mais simples do SVM e também mais básica é o chamado SVM Linear, que busca classificar os dados linearmente separáveis traçando uma reta entre os grupos, possuindo margens rígidas (classifica o modelo evitando classificações erradas, porém, dificultando sua

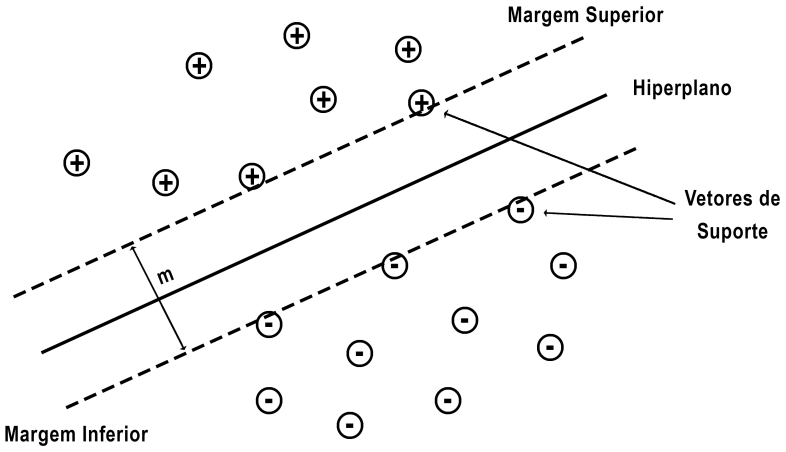
generalização) ou margens suaves (insere variáveis de folga que permitem algumas classificações erradas, porém, favorecem a generalização do modelo). Para casos mais complexos, junto da utilização do chamado truque do *Kernel*, é possível utilizar o modelo SVM para a classificação de problemas que não sejam linearmente separáveis.

Além de classificação, o SVM pode ser também utilizado para a regressão com o *Support Vector Regression* (PANAHI et al., 2020), detecção de *outliers* através do *One Class* SVM (MANEVITZ; YOUSEF, 2001), possuindo ainda sua forma regularizada, sendo chamado de L2-SVM (DOKTORSKI, 2011).

2.2.1 Classificador Linear

Para entender como funciona o modelo de máquina de vetores de suporte, comecemos pelo chamado SVM de margens rígidas. A técnica SVM busca separar observações utilizando um hiperplano ótimo, de forma que as margens formadas pelos vetores de suporte possuam a maior distância possível entre elas, sendo o hiperplano ótimo equidistante das duas margens. Para o modelo com margens rígidas, não é permitido que observações atravessem as margens (Vide Figura 5).

Figura 5 – Representação do modelo SVM com margens rígidas.



Fonte: Adaptado de HAMEL, 2011.

Seja o hiperplano ótimo uma superfície de decisão, ela é definida formalmente como:

$$\bar{w} \cdot \bar{x} = b$$

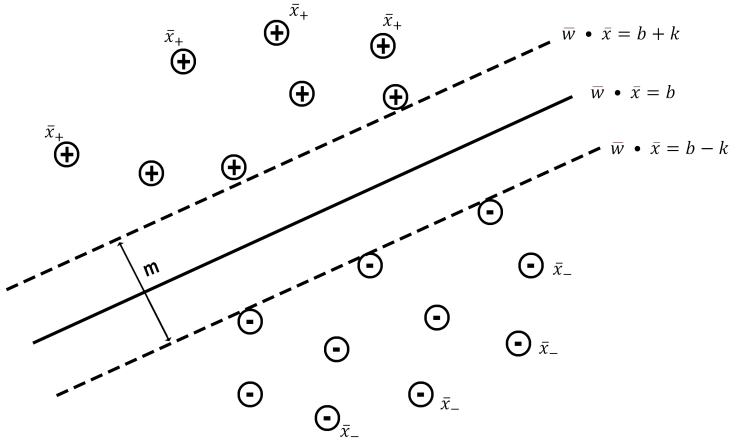
com $\bar{w}$ o vetor de coeficientes, $\bar{x}$ o vetor de observações e b o viés. Com base em nosso hiperplano ótimo, podemos definir também as equações das margens equidistantes ao hiperplano:

$$\bar{w} \cdot \bar{x}_+ = b + k$$

$$\bar{w} \cdot \bar{x}_- = b - k$$

sendo k uma constante, $\bar{x}_-$ representando os vetores abaixo da margem inferior, e $\bar{x}_+$ representando os vetores acima da margem superior (Vide Figura 6).

Figura 6 – Representação do modelo SVM com margens rígidas com suas equações.



Fonte: Adaptado de HAMEL, 2011.

Por definição, a distância entre as duas margens é dada por m, sendo esta distância calculada como a projeção do vetor $\bar{x}_+ - \bar{x}_-$ na direção do vetor normal ao hiperplano $\bar{w}$.

$$m = \frac{2k}{|\bar{w}|}.$$

Assim, precisamos agora maximizar m, e para isso, serão aplicadas algumas modificações. Primeiro, ao invés de maximizar m, iremos minimizar seu inverso, de forma que teremos o mesmo resultado, ou seja:

$$\max_{\bar{w}} \frac{2k}{|\bar{w}|} = \min_{\bar{w}} \frac{|\bar{w}|}{2k}.$$

Na sequência, a norma de $\bar{w}$ dada por $|\bar{w}|$ será elevada ao quadrado para facilitar algumas contas, sem alterar nosso problema de otimização, dado que ele é invariante a transformação com a função quadrática, sendo ela monótona para $x \geq 0$, e assim, otimizar sobre x^2 é o

mesmo que otimizar sobre x.

$$\min_{\bar{w}} \frac{|\bar{w}|}{2k} = \min_{\bar{w}} \frac{|\bar{w}|^2}{2k} = \min_{\bar{w}} \frac{1}{2k} \bar{w} \cdot \bar{w}.$$

Por fim, temos que a otimização é invariante sob escala com uma constante, e com isso, o valor da constante k pode ser definido arbitrariamente, e por conveniência será adotado $k = 1$.

$$\min_{\bar{w}} \frac{1}{2k} \bar{w} \cdot \bar{w} = \min_{\bar{w}} \frac{1}{2} \bar{w} \cdot \bar{w}. \tag{2.2}$$

Nosso problema de otimização apresentado na equação 2.2 estará sujeito as seguintes restrições:

$$\bar{w} \cdot (y_i \bar{x}_i) \geq 1 + y_i b, \quad \forall (\bar{x}_i, y_i). \tag{2.3}$$

Em outras palavras, as restrições definidas na equação 2.3, significa dizer que os pontos rotulados como $+1$ estarão sobre ou acima da margem superior, e os pontos rotulados como -1 estarão sobre ou abaixo da margem inferior, sendo assim, cada ponto no conjunto de dados será uma restrição para as margens. Portanto, para maximizar a distância entre as margens, basta:

$$\min_{\bar{w},b} \frac{1}{2} \bar{w} \cdot \bar{w} \tag{2.4}$$

sujeito as restrições apresentadas em 2.3 com $b \in \mathbb{R}$ e $\bar{w} \in \mathbb{R}^n$. O problema de otimização definido na equação 2.4 pode ser chamado também de problema de otimização *primal*.

Dando continuidade, podemos utilizar os multiplicadores de Lagrange para tornar nosso problema mais tratável. Dessa forma, construindo o Lagrangiano, função que une o problema de otimização *primal* (2.4) e suas restrições (2.3), temos:

$$L(\bar{\alpha}, \bar{w}, b) = \frac{1}{2} \bar{w} \cdot \bar{w} - \sum_{i=1}^{L} \alpha_i (y_i (\bar{w} \cdot \bar{x}_i - b) - 1) \tag{2.5}$$

onde $\alpha_i \geq 0$ com $i = 1, \cdots, L$ representa os multiplicadores de Lagrange.

Tendo como base o Lagrangiano (2.5), é preciso tomar a derivada parcial em relação a cada uma das variáveis *primais* $\bar{w}$ e $\bar{b}$, e então avaliar o ponto de sela de cada uma delas, igualando-as a 0. Com isso, temos:

$$\bar{w}^* = \sum_{i=1}^{L} \alpha_i y_i \bar{x}_i \quad \text{e} \quad \sum_{i=1}^{L} \alpha_i y_i = 0. \tag{2.6}$$

Substituindo 2.6 na equação 2.5, temos:

$$L(\bar{\alpha}, \bar{w}^*, b^*) = \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j.$$

Dessa forma, nosso problema de otimização é definido como:

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j \right) \tag{2.7}$$

sujeito as restrições

$$\sum_{i=1}^{L} \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

com $i = 1, \cdots, L$. O problema de otimização definido na equação 2.7 pode ser chamado também de problema de otimização *dual*. Resolvendo o problema de otimização *primal* em 2.4 e o problema de otimização *dual* em 2.7, sendo $\bar{w}^*$, b^* e $\bar{\alpha}$ as soluções, as respectivas funções de decisão são dadas por:

$$f(\bar{x}) = sinal(\bar{w}^* \cdot \bar{x} - b^*),$$

e

$$f(\bar{x}) = sinal \left(\sum_{i=1}^{L} \alpha_i^* y_i \bar{x}_i \cdot \bar{x} - b^* \right),$$

de forma que a função sinal:

$$sinal(k) = \begin{cases} +1, & \text{se } k \geq 0 \\ -1, & \text{se } k < 0. \end{cases}$$

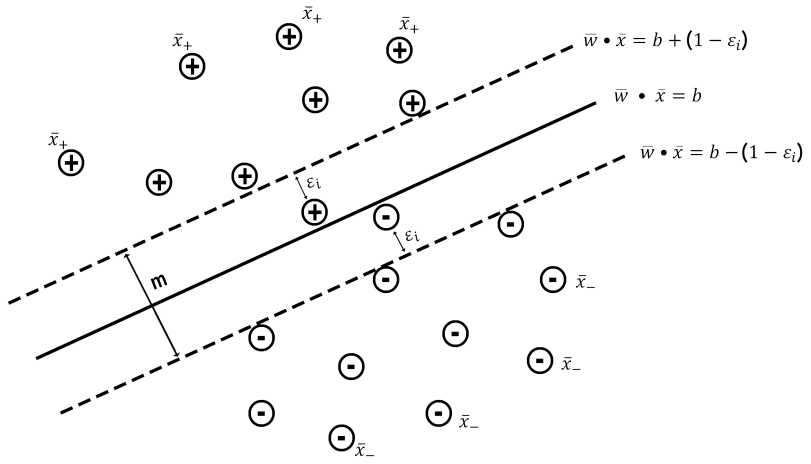
para $k \in \mathbb{R}$. Apesar de possuir duas formas de solução, classificador *primal* e classificador *dual*, na literatura é comum a utilização apenas do classificador *dual*, dada a possibilidade de generalização do modelo.

Diferentemente do SVM de margens rígidas, caracterizado por não permitir que as observações atravessem as margens construídas pelos vetores de suporte, de forma a evitar erros de classificação, para o SVM de margens suaves é permitido que algumas observações atravessem as margens, e assim, garanta um poder maior de generalização (Vide Figura 7).

Para transformar o modelo de margens rígidas para margens suaves, são incorporadas as margens as chamadas variáveis de folga denotadas por $\varepsilon_i > 0$, $\forall i = 1, \cdots, L$. Se repetimos o mesmo processo realizado para encontrar a distância entre as margens no caso do SVM com margens rígidas, iremos chegar a um problema de otimização *primal* semelhante ao visto anteriormente (equação 2.4), definido por:

$$\min_{\bar{w}, b, \bar{\varepsilon}} \frac{1}{2} \bar{w} \cdot \bar{w} + C \sum_{i=1}^{L} \varepsilon_i \tag{2.8}$$

Figura 7 – Representação do modelo SVM com margens suaves.



Fonte: Adaptado de HAMEL, 2011.

sujeito as restrições

$$y_i(\bar{w} \cdot \bar{x}_i - b) + \varepsilon_i - 1 \geq 0 \quad \text{e} \quad \varepsilon_i \geq 0$$

com $i = 1, \cdots, L$, $\bar{\varepsilon} = (\varepsilon_1, \cdots, \varepsilon_L)$ e $C > 0$.

A constante C funciona como um termo de regularização, também chamada de custo, que irá impor um peso a minimização dos erros, sendo ela inclusa pois não existe um limite para o número de classificações erradas. Dessa forma, o problema de otimização *primal* definido na equação 2.8 torna-se um *trade-off* entre o tamanho da distância entre as margens e o tamanho do erro, onde o erro é a soma dos valores das variáveis de folga.

Transformando a equação 2.8 num problema de otimização *dual* com a utilização dos multiplicadores de Lagrange, ele será escrito como:

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j \bar{x}_i \cdot \bar{x}_j \right), \tag{2.9}$$

sujeito as restrições

$$\sum_{i=1}^{L} \alpha_i y_i = 0, \tag{2.10}$$

$$0 \leq \alpha_i \leq C \tag{2.11}$$

com $i = 1, \cdots, L$ e $C > 0$ a constante de custo. Como é possível observar, nossa função 2.9 apresenta a mesma estrutura vista para o SVM de margens rígidas (2.7), e de fato não existem diferenças entre os problemas de otimização, sendo esta diferença vista apenas nas restrições (2.10 e 2.11).

Por fim, sejam $\bar{w}^*$, b^* e $\bar{\alpha}^*$ soluções para os problemas de otimização descritos, a função de decisão para os classificadores *primal* e *dual* do SVM de margens suaves são dadas respectivamente por:

$$f(\bar{x}) = sinal(\bar{w}^* \cdot \bar{x} - b^*)$$

e

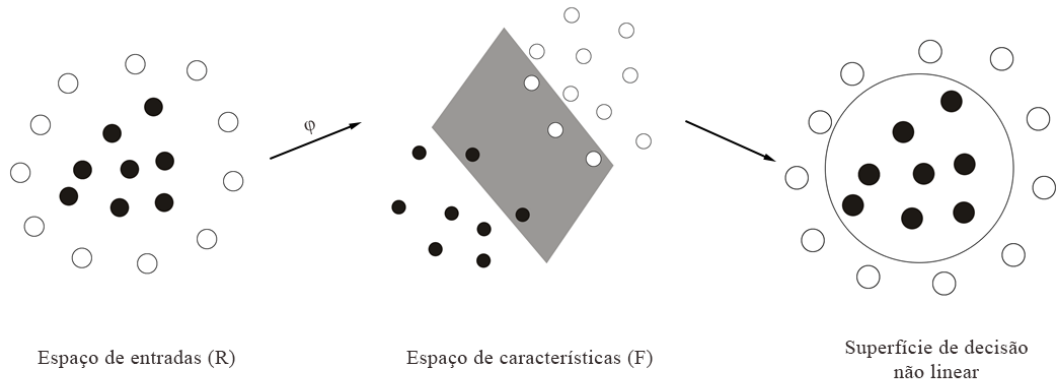
$$f(\bar{x}) = sinal\left(\sum_{i=1}^{L} \alpha_i^* y_i \bar{x}_i \cdot \bar{x} - b^*\right).$$

2.2.2 Classificador não linear: O truque do *Kernel*

Até agora, o modelo SVM apresentado pode ser utilizado apenas para classificação de problemas linearmente separáveis. Entretanto, no mundo real são poucos os conjuntos de dados que apresentam tal comportamento. Frente a isso, temos a característica que torna as Máquinas de Vetores de Suporte tão notáveis, sua estrutura linear básica é facilmente estendida para o caso em que o conjunto de dados não é linearmente separável (HAMEL, 2011).

A ideia por trás dessa extensão é a de mapear os dados de treinamento, chamado de espaço de entradas (*Input Space*) (R), de forma que estes dados sejam levados para um espaço de maior dimensão que será chamado de espaço de características (*Feature Space*) (F), onde os dados são linearmente separáveis. O processo de mapeamento é realizado através de uma função de mapeamento $\phi : \mathbb{R}^n \to \mathbb{R}^m$ com $m \geq n$, e após a classificação, os dados são devolvidos a sua dimensão original.

Figura 8 – O Truque do *Kernel*.



Fonte: Adaptado de PIMENTEL, 2021.

Dada uma função de mapeamento $\phi : \mathbb{R}^n \to \mathbb{R}^m$ com $m \geq n$, uma função na forma:

$$k(\bar{x}, \bar{y}) = \phi(\bar{x}) \cdot \phi(\bar{y})$$

onde $\bar{x}$, $\bar{y} \in \mathbb{R}^n$, é chamada de *kernel* ou função *kernel*.

As funções *kernel* avaliam um produto escalar no espaço de características (F), porém, a característica que define o *kernel* é um valor calculado no espaço de entradas (R). Com isso, podemos reescrever nosso problema de otimização *dual* em termos da função *kernel*:

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j \phi(\bar{x}_i) \cdot \phi(\bar{x}_j) \right)$$

assim como nossa função de decisão:

$$f(\bar{x}) = sinal \left(\sum_{i=1}^{L} \alpha_i^* y_i \phi(\bar{x}_i) \cdot \phi(\bar{x}) - b^* \right).$$

Esse processo é conhecido como o truque do *kernel* (*kernel trick*), ou seja, usando qualquer função *kernel* apropriada, podemos tirar vantagem do mapeamento em espaços de características (F), sem pagar o preço de realmente calcular o mapeamento explícito, uma vez que os cálculos no espaço de características (F) sempre simplificam os cálculos no espaço de entradas (R).

Ao selecionar a função *kernel* criteriosamente podemos controlar a complexidade do modelo. O truque está em encontrar o *kernel* apropriado para construir o modelo para um determinado conjunto de dados. O processo de selecionar um *kernel* e os valores associados de seus parâmetros livres é chamado de *feature search*. A Tabela 4 lista alguns dos *kernels* comumente utilizados.

Tabela 4 – *Kernels* mais utilizados

Nome do *Kernel*	$k(\bar{x}, \bar{y})$	Parâmetros
Linear	$\bar{x} \cdot \bar{y}$	-
Polinomial Homogêneo	$(\bar{x} \cdot \bar{y})^d$	$d \geq 2$
Polinomial Não Homogêneo	$(\bar{x} \cdot \bar{y} + c)^d$	$d \geq 2, c > 0$
Gaussiano	$exp(\frac{-\lvert\bar{x}-\bar{y}\rvert^2}{2\sigma^2})$	$\sigma > 0$

Fonte: Autor.

2.2.3 SVM regularizado: L2 SVM

Seja o modelo SVM de margens suaves como o anteriormente definido em 2.8, observa-se que, a soma das variáveis de folga ε_i é usada comumente na forma linear, e recebe o nome

de L1-SVM. Em algumas situações, utiliza-se como forma de regularizar o modelo, ao invés de uma soma linear, o quadrado da soma das variáveis de folga, chegando assim ao chamado L2-SVM.

$$\min_{\bar{w},b,\bar{\varepsilon}} \frac{1}{2}\bar{w}\cdot\bar{w} + C\sum_{i=1}^{L}\varepsilon_i^2 \tag{2.12}$$

ainda sujeito a mesma restrição antes vista

$$y_i(\bar{w}\cdot\bar{x}_i - b) \geq 1 - \varepsilon_i$$

com $i = 1, \cdots, L$, $C > 0$. Introduzindo os multiplicadores de Lagrange α_i, assim como no L1-SVM, nós obtemos o problema de otimização *dual* para o L2-SVM dado por:

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L}\alpha_i\alpha_j y_i y_j \left(\bar{x}_i\cdot\bar{x}_j + \frac{\delta_{ij}}{C} \right) \right) \tag{2.13}$$

sujeito as restrições

$$\sum_{i=1}^{L}\alpha_i y_i = 0,$$

$$\alpha_i \geq 0$$

para $i = 1, \cdots, L$, onde δ_{ij} é a função delta de Kronecker que assume valor igual a 1 quando $i = j$ e valor igual a 0 quando $i \neq j$. Assim como para o L1-SVM, a função de decisão para o classificador L2-SVM é dada por:

$$f(\bar{x}) = sinal\left(\sum_{i=1}^{L}\alpha_i^* y_i \bar{x}_i\cdot\bar{x} - b^* \right). \tag{2.14}$$

Vale ressaltar que é possível ainda realizar a substituição do produto escalar $\bar{x}_i\cdot\bar{x}_j$ por uma função kernel $k(\bar{x}_i, \bar{x}_j)$, permitindo assim tornar o L2-SVM em um modelo não linear.

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L}\alpha_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L}\alpha_i\alpha_j y_i y_j \left(k(\bar{x}_i, \bar{x}_j) + \frac{\delta_{ij}}{C} \right) \right)$$

sujeito as restrições

$$\sum_{i=1}^{L}\alpha_i y_i = 0,$$

$$\alpha_i \geq 0.$$

2.3 *MINIMUM ENCLOSING BALL* (MEB)

Dado um conjunto $S = \{x_1, \cdots, x_n\}$ com $x_i \in \mathbb{R}^n$, o *minimum enclosing ball* ou *minimal enclosing ball* de S, que será denotado como $MEB(S)$, é um problema da geometria computacional que busca a bola de menor raio r e centro c que contém todos os pontos de S, sendo expressado como:

$$\min_{r,c} r^2 \tag{2.15}$$

sujeito as restrições:

$$||c - \bar{x}_i||^2 \leq r^2, \quad i = 1, \ldots, n.$$

O problema de otimização definido por 2.15 é um problema padrão de programação quadrática que pode ser resolvido pelo método de Lagrange (STRACK, 2013). Dessa forma, temos nosso Lagrangiano dado por:

$$L(\bar{\alpha}, c, r) = r^2 - \sum_{i=1}^{L} \alpha_i (r^2 - ||c - \bar{x}_i||^2) \tag{2.16}$$

onde $\alpha_i \geq 0$ com $i = 1, \ldots, L$ representa os multiplicadores de Lagrange. Partindo de 2.16 e tomando suas derivadas parciais, obtemos:

$$c^* = \sum_{i=1}^{L} \alpha_i \bar{x}_i \quad \text{e} \quad \sum_{i=1}^{L} \alpha_i = 1. \tag{2.17}$$

Substituindo 2.17 em 2.16, temos:

$$L(\bar{\alpha}, w^*, r^*) = \sum_{i=1}^{L} \alpha_i ||\bar{x}_i \cdot \bar{x}_i|| - \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j \bar{x}_i \cdot \bar{x}_j.$$

Consequentemente, nosso problema de otimização para o MEB é definido como:

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L} \alpha_i ||\bar{x}_i \cdot \bar{x}_i|| - \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j \bar{x}_i \cdot \bar{x}_j. \right) \tag{2.18}$$

sujeito as restrições

$$\sum_{i=1}^{L} \alpha_i = 1 \tag{2.19}$$

$$\alpha_i \geq 0 \tag{2.20}$$

com $i = 1, \cdots, L$.

Se desejarmos, podemos generalizar o nosso problema MEB através da utilização do truque do *kernel*, construindo assim o chamado *kernel-MEB*. Assim, seja $k(\bar{x}_i \cdot \bar{x}_j) = \phi(\bar{x}_i) \cdot \phi(\bar{x}_j)$

uma função *kernel* qualquer, podemos reescrever a equação 2.18 como:

$$\max_{\bar{\alpha}} \left(\sum_{i=1}^{L} \alpha_i ||k(\bar{x}_i \cdot \bar{x}_i)|| - \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j k(\bar{x}_i \cdot \bar{x}_j). \right) \tag{2.21}$$

ainda sujeito as mesma restrições definidas em 2.19 e 2.20, com $i = 1, \cdots, L$.

Se considerarmos funções *kernel* normalizadas, ou seja, funções *kernel* $k(\bar{x}, \bar{y})$ que dependem apenas de $\bar{x} - \bar{y}$, de forma que $k(\bar{x}, \bar{x})$ será constante, a restrição de igualdade (2.19) implicará que o termo linear da equação 2.21 será constante. Dessa forma, nosso problema de otimização pode ser escrito apenas como:

$$\max_{\bar{\alpha}} \left(- \sum_{i=1}^{L} \sum_{j=1}^{L} \alpha_i \alpha_j k(\bar{x}_i \cdot \bar{x}_j). \right) \tag{2.22}$$

dado que com a função *kernel* constante, $\sum_{i=1}^{L} \alpha_i ||k(\bar{x}_i \cdot \bar{x}_i)||$ não será afetado pelo processo de otimização.

2.4 MÉTRICAS DE AVALIAÇÃO

2.4.1 Matriz de confusão

A matriz de confusão é uma tabela cruzada com os valores reais e os valores preditos pelo modelo, que ajuda a visualizar o comportamento do algoritmo de uma forma geral (BALDI et al., 2000). Ela é construída partindo da ideia de classificação binária, podendo ser expandida para casos com mais classes. Para o caso binário, a tabela é construída a partir de 4 valores. Originalmente positivos e classificados como positivos (VP - Verdadeiro Positivo), originalmente negativos e classificados como positivos (FP - Falso Positivo), originalmente positivos e classificados como negativos (FN - Falso Negativo) e originalmente negativos e classificados como negativos (VN - Verdadeiro Negativo). Um exemplo da matriz de confusão é apresentada na Tabela 5.

Tabela 5 – Matriz de Confusão

	Predito Positivo	Predito Negativo
Real Positivo	VP	FN
Real Negativo	FP	VN

Fonte: Autor.

2.4.2 Acurácia

Apesar da simples visualização da matriz de confusão, é sempre preferível a utilização de métricas que possam resumi-la. Com essa finalidade temos a acurácia (ACC) que apresenta a taxa de acertos de um modelo avaliado e possui a seguinte fórmula:

$$ACC = \frac{VP + VN}{VP + FN + FP + VN}$$

Por se tratar de um valor entre 0 e 1, quanto mais próximo de 1, melhor será a capacidade de predição do modelo.

2.4.3 Sensibilidade

Outra métrica bastante conhecida é a sensibilidade ou *recall*. A Sensibilidade é utilizada para avaliar a capacidade do modelo de classificar corretamente a classe positiva. Em outras palavras, a sensibilidade mede o número de classificações positivas corretas dentre todas as observações que eram originalmente pertencentes a classe positiva.

$$SEN = \frac{VP}{VP + FN}$$

Assim como a acurácia varia entre 0 e 1, quanto mais próximo de 1, melhor será a capacidade de predição do modelo para as classes positivas.

2.4.4 Precisão

A precisão por sua vez é utilizada para quantificar o número de vezes que um modelo classifica corretamente como positivo dentre todas as classificações como positivo. Quando aumentamos o valor da precisão, diminuímos o valor da sensibilidade, e vice versa.

$$PREC = \frac{VP}{VP + FP}$$

Assim como as demais varia entre 0 e 1 e quanto mais próximo de 1 melhor a capacidade do modelo.

2.4.5 *F1-Score*

Por fim, temos a métrica chamada de *F1-Score* ou *F1 Measure*. O *F1-Score* é definida como a média harmônica entre as medidas de precisão e sensibilidade. Em comparação com a medida de acurácia, a *F1-Score* é considerada mais justa mesmo quando utilizada para bases de dados desbalanceados.

$$F1 = \frac{2 \times PREC \times SEN}{PREC + SEN}$$

Apresenta valores entre 0 e 1, sendo quanto mais próximo de 1, melhora a capacidade do modelo.

2.5 FORMAS DE TRATAMENTO PARA DADOS DESBALANCEADOS

Assim como dados linearmente separáveis, no mundo real não são muitas as bases de dados que apresentam o perfeito balanceamento, de maneira que é necessária a utilização de técnicas para torna-los balanceados, ou mudar a forma de abordagem. Frente a esse problema, uma solução bastante comum é a inserção ou remoção de observações aos dados originais, como forma de tornar a base de dados balanceada, os métodos de *undersampling* e *oversampling*.

2.5.1 *Undersampling*

Quando trabalhamos com uma base de dados desbalanceada, a classe que possui um maior número de observações é chamada de classe majoritária, enquanto que as demais observações pertencem a chamada classe minoritária. Partindo deste princípio, a técnica conhecida como *undersampling* consiste na redução de observações da classe majoritária, de forma que o número de observações seja altamente reduzido (LIU, 2004).

Existem diferentes formas de equiparar as classes via *undersampling* na literatura. Entretanto, a escolhida para ser aqui utilizada consiste na simples subamostragem aleatória. Basicamente a ideia consiste na eliminação aleatória de observações da classe majoritária até que a proporção de observações entre as classes minoritária e majoritária estejam tão próximas quanto o desejado.

O principal problema desta técnica consiste no fato de não ser possível exercer o controle sobre quais observações serão eliminadas, de forma que informações importantes sobre o limite

de decisão entre as classes majoritária e minoritária podem ser eliminadas (LIU, 2004).

2.5.2 *Oversampling*

Por sua vez, a técnica conhecida como *oversampling* desempenha o processo oposto ao *undersampling*, sendo assim, consiste em aumentar o número de observações pertencentes a classe minoritária até que a proporção de observações entre as classes majoritária e minoritária atinjam a proporção desejada.

A vantagem da utilização desta técnica é que nenhuma observação do conjunto de dados é perdida, dado que todas as observações das duas classes são mantidas. Entretanto, a base de dados pode ser ampliada de forma exagerada (LIU, 2004).

Assim como para a técnica *undersampling*, existem várias formas de aumentar o tamanho da base de dados expandindo a classe minoritária. Novamente, a forma mais simples e também a escolhida para ser utilizada será a sobreamostragem aleatória. As observações da classe minoritária são selecionadas de forma aleatória e com reposição, sendo então duplicados e adicionadas ao novo conjunto de dados (LIU, 2004).

3 NOVAS PROPOSTAS PARA DADOS MASSIVOS

Em seguida a apresentação de diversos conceitos introdutórios necessários, neste capítulo será abordado o problema existente na utilização dos modelos de Máquinas de Vetores de Suporte para dados massivos. Além do mais, serão também comentadas técnicas comumente utilizadas para tratar tal problema, como também será proposto a utilização de dois novos modelos baseados na junção dos *Weak SVMs* e a construção de *coresets*.

O problema a ser abordado e discutido, como anteriormente comentado, se refere a utilização do modelo para grandes bases de dados. Este problema é gerado a partir da forma de implementação do modelo que precisa resolver um problema de otimização quadrática, o que o torna com um custo computacional diretamente proporcional ao tamanho da base que precisa ser classificada.

Dada a sua importância, várias formas diferentes foram propostas para ser possível a implementação do SVM no contexto de grandes bases de dados. Começando por abordagens mais recentes, podemos citar o chamado SVM multinível (SCHLAG; SCHMITT; SCHULZ, 2021), novo caminho de pesquisa promissor e amplamente utilizado em problemas de particionamento de grafos. Quando tratamos um problema utilizando o *framework* multinível, o primeiro passo é a construção de uma hierarquia do problema, onde cada nível da hierarquia é um problema que diminui de tamanho, mas reflete a estrutura do problema original.

Na sequência, o SVM é treinado para o 'maior' problema, e este treinamento é projetado para o topo da hierarquia, ou seja, o modelo vai sendo refinado para melhor se adequar ao problema original, enquanto aumenta o tamanho do conjunto de dados utilizado para o treinamento. De forma geral, observou-se que o SVM multinível reduz o tempo de computação enquanto é comparável e muitas vezes melhor do que abordagens não hierárquicas em termos de qualidade de previsão em grandes conjuntos de dados.

Outro caminho possível é a utilização da programação em paralelo conjuntamente com o modelo SVM, sendo um exemplo disso o *parallel SVM* (SINGH et al., 2021). Ao utilizá-lo, o SVM é treinado por meio de subSVMs, onde grandes escalas de dados podem ser divididas em grupos menores e independentes. Os vetores de suporte dos antigos subSVMs são usados como entrada para os subSVMs posteriores, sendo o processo repetido até que um único conjunto de vetores seja alcançado. O tamanho do conjunto de treinamento vai sendo reduzido sequencialmente em cada estágio. Paralelizar reduz o número de conjuntos de dados para o

modelo, acelerando assim o processo de teste e tornando-o mais eficiente nas previsões.

Outro modelo um tanto quanto recente é o *Quantum Support Vector Machine* (REBENTROST; MOHSENI; LLOYD, 2014), que performa através de uma rápida avaliação quântica de produtos internos e uma solução quântica através de um algoritmo de inversão de matrizes, algoritmo esse que pode ser utilizado após o modelo ser reescrito como um problema aproximado de mínimos quadrados. Além disso, o método ao utilizar-se de um técnica de exponenciação de matrizes esparsas, pode revelar de forma eficiente e quântica, os maiores autovalores e autovetores correspondentes das matrizes de sobreposição de dados de treinamento e covariância.

Voltando ainda mais no tempo para técnicas mais 'tradicionais' e conhecidas, chegamos em dois grupos de técnicas utilizadas para grandes bases de dados. O primeiro grupo trás técnicas que procuram selecionar conjuntos de dados de treinamento menores, onde estejam presentes os chamados vetores de suporte. Por sua vez, o segundo grupo de técnicas que buscam reduzir a complexidade do problema de otimização, tornando-o mais eficiente.

Começando pela seleção de amostras, uma das formas de se realizar esta seleção de observações é por meio de técnicas de *clustering*. Quando trabalhamos com classificação e SVM, os vetores de suporte são apenas um pequeno subconjunto dos vetores originais presentes nos dados, sendo esse conjunto diminuído durante o treinamento, eliminando os vetores desnecessários. A técnica SVM-KM (ALMEIDA; BRAGA; BRAGA, 2000), utiliza do algoritmo *K-means* para realizar esta seleção ao organizar os dados de treinamento em pequenos *clusters*. Após formados, os grupos mais distantes em relação ao hiperplano são substituídos pelo seu centroide, e os grupos mais próximos não são afetados. Desta maneira, o número de vetores será reduzido, acelerando o treinamento dos SVMs e exigindo menos esforço computacional sem comprometer a capacidade de generalização do SVM.

Outra técnica baseada na análise de *clusters* é o chamado *Clustering-Based SVM* (CB-SVM) (YU; YANG; HAN, 2003), que através da utilização de amostragem seletiva conjuntamente com um algoritmo hierárquico de *microclustering*, varre todo o conjunto de dados de forma única construindo uma amostra de alta qualidade que resume os dados maximizando o aprendizado da técnica SVM. O CB-SVM é escalável em termos de eficiência de treinamento, enquanto maximiza o desempenho dos SVMs.

Por sua vez, (WANG et al., 2014) buscou por meio da utilização dos *Weak SVMs*, que são modelos ajustados com base em pequenas bases de dados amostrados do conjunto de treinamento original, realizar a seleção de observações que tenham uma maior possibilidade de serem os vetores de suporte. Ao realizar a seleção destas observações, espera-se que estas

observações estejam mais próximas das margens, tendo assim uma maior dificuldade de serem preditas, o que indicam que as mesmas possuem uma maior chance de serem os ditos vetores de suporte. Os resultados alcançados por este método são comparáveis a outros métodos como o PEGASOS (SHALEV-SHWARTZ; SINGER; SREBRO, 2007), LIBLINEAR (FAN et al., 2008) e o RSVM (LEE; MANGASARIAN, 2001).

Passando para o grupo de técnicas onde se busca reduzir a complexidade do problema de otimização, podemos citar a técnica *Sequential Minimal Optimization (SMO)* (ZENG et al., 2008), onde o problema de otimização quadrática é dividido em uma série de problemas menores que são resolvidos de forma analitica. A quantidade de memória necessária para o SMO é linear ao tamanho do conjunto de treinamento, o que permite ao SMO lidar com conjuntos de treinamento muito grandes.

Por fim, dentre as mais diversas técnicas existentes, chegamos as técnicas a serem abordadas e tratadas neste trabalho, *Core Vector Machine (CVM)* (TSANG et al., 2005), *Ball Vector Machine (BVM)* (TSANG; KOCSOR; KWOK, 2007) e *Sphere Support Vector Machine* (STRACK et al., 2013). As três técnicas se baseiam na utilização dos chamados *coresets*, que são nada mais do que subconjuntos de pontos que são utilizados para resolver problemas de otimização, apresentando uma solução aproximada a obtida ao se utilizar todos os pontos presentes, onde se tem como principal vantagem a redução do custo computacional (STRACK, 2013). Além disso, elas partem do ponto em que o modelo SVM pode ser escrito como o problema de geometria computacional *Minimum Enclosing Ball (MEB)*, e a partir dele, ao construir um modelo aproximado, o problema de otimização do modelo SVM é solucionado.

Baseando-se nessa ideia, conjuntamente com o apresentado anteriormente ao se falar dos *Weak SVMs*, nas próximas subseções serão apresentadas de forma mais detalhada as técnicas CVM, BVM e *Sphere SVM*, como também o modelo proposto que permite a estimação do modelo SVM sem perda da capacidade preditiva, ou com perda mínima de capacidade preditiva, e com um ganho computacional médio superior a 60% do tempo total.

3.1 *CORE VECTOR MACHINE*

Ao observar diversas implementações práticas do SVM, TSANG et al. percebeu que as soluções obtidas eram na verdade aproximadas, de forma que estas soluções quase ótimas, geralmente são boas o suficiente para as aplicações práticas. Notou-se ainda que todo esse poder de 'aproximação' visto para o modelo SVM, nunca foi totalmente explorado em suas

implementações, tendo então como objetivo a construção de um algoritmo aproximado para os modelos SVM.

Como anteriormente comentado, o *Core Vector Machine* parte de dois conceitos, a utilização de *coresets* e a escrita do modelo SVM como um MEB. O problema *Minimun Enclosing Ball (MEB)* foi apresentado na seção 2.3, restando agora apresentarmos de modo formal o conceito de *coresets*.

Em alguns exemplos de problemas de otimização, é comum se trabalhar com um subconjunto de observações ou pontos, $Q \subset S$, de forma que, o resultado obtido ao se trabalhar com Q é o mesmo, ou bastante próximo, ao resultado obtido ao se trabalhar com S. Nesta situação, o conjunto Q é o que chamamos de *coreset*. Os *coresets* têm como principal vantagem o ganho computacional obtido com a sua utilização, onde o algoritmo executado utilizando o conjunto *coreset* retornará um resultado mais rapidamente do que o algoritmo executado para todo o conjunto de dados (STRACK, 2013).

Com o conceito de *coresets* apresentado, iremos agora apresentar o modelo L2-SVM modificado por TSANG et al. para que possa ser visto como um problema MEB.

Assim, o problema de otimização inicial para o L2-SVM é definido como:

$$\min_{\bar{w},b,\bar{\varepsilon}} \frac{1}{2}\bar{w}\cdot\bar{w} + \frac{b}{2} - \rho + \frac{C}{2}\sum_{i=1}^{L}\varepsilon_i^2 \tag{3.1}$$

sujeito a restrição

$$y_i(\bar{w}\cdot\bar{x}_i + b) \geq \rho - \varepsilon_i, \quad i = 1, \ldots, m.$$

Quando comparamos as equações 2.12 e 3.1, é vista a adição do termo de viés $\frac{b}{2}$ e da variável ρ. A adição do viés $\frac{b}{2}$ seria como adicionar um *feature* com todos os valores iguais a 1 ao espaço de características, ou seja, a dimensão original dos dados, fazendo

$$\phi'(x) = \begin{bmatrix} \phi(x) \\ 1 \end{bmatrix}.$$

Já para a variável ρ, sua adição substitui a restrição $\sum_{i=1}^{L}\alpha_i y_i = 0$ na equação 2.13 por $\sum_{i=1}^{L}\alpha_i = 1$. Assim, o termo $\sum_{i=1}^{L}\alpha_i$ em 2.13 pode ser removido por se tratar de uma constante. Modificações como essas que adicionam novas variáveis, podem ser também encontradas no trabalho desenvolvido por SCHÖLKOPF et al., onde utilizou-se a nova variável para limitar o número de vetores de suporte que seriam utilizados pelo ν-SVM. Por sua vez, a variável ρ é usada para facilitar a transformação do L2-SVM no problema MEB (STRACK, 2013).

Escrevendo a equação 3.1 na forma do Lagrangiano, temos:

$$L(\bar{\alpha}, \bar{w}, b, \rho, \varepsilon) = \frac{1}{2}\bar{w} \cdot \bar{w} + \frac{b^2}{2} - \rho + \frac{C}{2}\sum_{i=1}^{L} \varepsilon_i^2 - \sum_{i=1}^{L} \alpha_i (y_i(\bar{w} \cdot \bar{x}_i - b) - \rho + \varepsilon_i) \tag{3.2}$$

com $\alpha_i \geq 0$, $i = 1, \ldots, m$. Ao derivarmos 3.2 parcialmente, temos que:

$$\bar{w}^* = \sum_{i=1}^{L} \alpha_i y_i \bar{x}_i, \tag{3.3}$$

$$b^* = \sum_{i=1}^{L} \alpha_i y_i, \tag{3.4}$$

$$\sum_{i=1}^{L} \alpha_i = 1, \tag{3.5}$$

$$\varepsilon_i^* = \frac{\alpha_i}{C}. \tag{3.6}$$

Substituindo as equações (3.3-3.6) no Lagrangiano dado em 3.2, temos que:

$$L(\bar{\alpha}, \bar{w}, b, \rho, \varepsilon) = -\frac{1}{2}\left(\sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j (\bar{x}_i \cdot \bar{x}_j) + \sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j + \frac{1}{C}\sum_{i=1}^{L} \alpha_i^2\right).$$

Dessa forma, nosso problema de otimização é reescrito como:

$$\min_{\alpha}\left(\sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j (\bar{x}_i \cdot \bar{x}_j) + \sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j + \frac{1}{C}\sum_{i=1}^{L} \alpha_i^2\right)$$

Sujeito às restrições:

$$\sum_{i=1}^{L} \alpha_i = 1,$$

$$\alpha_i \geq 0,$$

com $i = 1, \ldots, L$. Com isso, nossa função de decisão é escrita como:

$$f(\bar{x}) = sinal\left(\sum_{i=1}^{L} \alpha_i^* y_i \bar{x}_i \cdot \bar{x} - b^*\right). \tag{3.7}$$

Observa-se que a função de decisão para o L2-SVM original (2.14) e a função de decisão para o L2-SVM modificado (3.7) são idênticas, existindo apenas uma diferença no valor do viés estimado b^*.

Com a substituição do produto escalar por uma função kernel, nosso problema de otimização é escrito como:

$$\min_{\alpha}\left(\sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j k(\bar{x}_i, \bar{x}_j) + \sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j y_i y_j + \frac{1}{C}\sum_{i=1}^{L} \alpha_i^2\right) \tag{3.8}$$

sujeito às restrições:

$$\sum_{i=1}^{L} \alpha_i = 1,$$
$$\alpha_i \geq 0,$$

com $i = 1, \ldots, L$. Por sua vez, nossa função de decisão é escrita como:

$$f(\bar{x}) = sinal\left(\sum_{i=1}^{L} \alpha_i^* y_i k(\bar{x}_i, \bar{x}) + b^*\right).$$

Se definirmos uma nova função *kernel* $\tilde{k}(\bar{x}_i, \bar{x}_j)$, com o objetivo de melhor captar e impedir a perda de informações dos dados, dada por:

$$\tilde{k}(\bar{x}_i, \bar{x}_j) = y_i y_j k(\bar{x}_i, \bar{x}_j) + y_i y_j + \frac{1}{c}\delta_{ij}$$

onde $k(\bar{x}_i, \bar{x}_j)$ é o *kernel* utilizado originalmente no SVM e δ_{ij} é o chamado delta de *Kronecker* (0 se $i \neq j$ e 1 se $i = j$), podemos reescrever a equação 3.8, da seguinte forma:

$$\min_{\alpha}\left(\sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j \tilde{k}(\bar{x}_i, \bar{x}_j) - \sum_{i=1}^{L} \alpha_i \tilde{k}(\bar{x}_i, \bar{x}_j)\right),$$

sujeito às restrições:

$$\sum_{i=1}^{L} \alpha_i = 1,$$
$$\alpha_i \geq 0,$$

com $i = 1, \ldots, L$. Seja $k(\bar{x}_i, \bar{x}_j)$ uma função *kernel* constante ou normalizada, temos que $\tilde{k}(\bar{x}_i, \bar{x}_j)$ também será um *kernel* constante. Dessa forma, com o *kernel* constante, $\sum_{i=1}^{L} \alpha_i \tilde{k}(\bar{x}_i \cdot \bar{x}_i)$ não será afetado pelo processo de otimização, e o problema será então escrito como:

$$\min_{\alpha}\left(\sum_{i=1}^{L}\sum_{j=1}^{L} \alpha_i \alpha_j \tilde{k}(\bar{x}_i, \bar{x}_j)\right). \tag{3.9}$$

Isto significa que para funções *kernel* normalizadas, como por exemplo o *kernel* gaussiano, o L2-SVM (3.1) pode ser tratado como um problema MEB (2.22), e dessa forma, a solução obtida resolvendo o problema MEB no espaço de características e a solução do L2-SVM modificado são iguais (STRACK, 2013).

O algoritmo

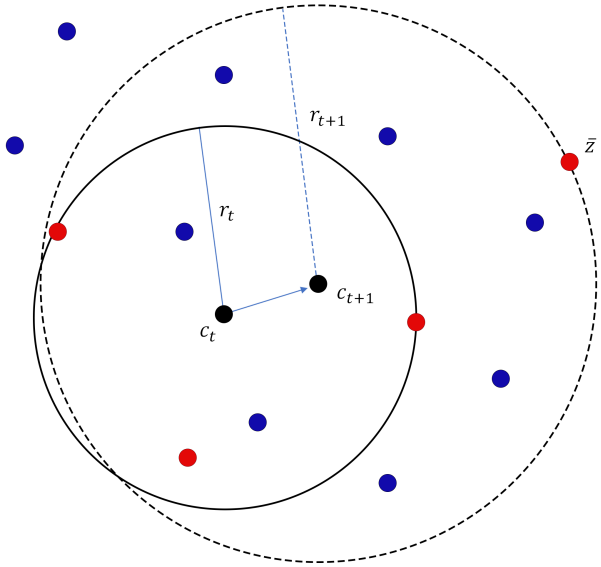
Após a formulação do modelo SVM como um MEB, foi obtida uma nova função *kernel* transformada $\tilde{k}$, e agora, para a solução deste problema, foi utilizado por TSANG et al. um algoritmo aproximado apresentado por (BĂDOIU; CLARKSON, 2008). A ideia baseia-se em ir expandindo a bola ao incluir o ponto mais distante do centro atual (TSANG et al., 2005), e para isso, definiu-se o *coreset*, o centro e o raio da bola na t-ésima iteração como Q_t, c_t e r_t, respectivamente. Assim, para um $\eta > 0$, o *Core Vector Machine* irá funcionar da seguinte forma:

1. Inicializa-se Q_0, c_0 e r_0;
2. O algoritmo finaliza se não for encontrado nenhum ponto $\bar{z}$ tal que $\tilde{\phi}(\bar{z})$ esteja fora da bola $B(c_t, (1+\eta)r_t)$;
3. Encontra-se $\bar{z}$ com $\tilde{\phi}(\bar{z})$ mais distante de c_t, e faz $Q_{t+1} = Q_t \cup \{\bar{z}\}$;
4. Encontra-se o novo $MEB(Q_{t+1})$, onde $c_{t+1} = c_{MEB(Q_{t+1})}$ e $r_{t+1} = r_{MEB(Q_{t+1})}$;
5. Incrementa t em 1 e retorna ao segundo passo.

Os valores inicias são definidos e com base nos valores de centro e raio (c_0 e r_0) é executado o algoritmo MEB e encontra-se a menor bola $B(c_t, (1+\eta)r_t)$. Verifique a existência ou não de pontos de treinamento fora da menor bola calculada, e se nenhum ponto for encontrado fora da bola, encerre o algoritmo. Caso contrário, siga em frente. Encontre o ponto $\bar{z}$ que esteja mais distante do centro da bola inicialmente calculada e adicione este ponto ao seu conjunto *coreset* fazendo $Q_{t+1} = Q_t \cup \{\bar{z}\}$. Com base neste *coreset*, encontre novamente a menor bola, porém, desta vez tendo como base a relação entre o treinamento de um modelo SVM e o MEB, e então defina um novo centro e um novo raio para próxima interação fazendo $c_{t+1} = c_{MEB}$ e $r_{t+1} = r_{MEB}$.

Por fim, incremente t em 1 e repita o processo interativo. Todos os pontos que são adicionados ao conjunto *coreset* serão chamados de *core vectors*. Um passo do algoritmo descrito pode ser visto na Figura 9. Maiores detalhes da implementação podem ser encontradas em (TSANG et al., 2005).

Figura 9 – Uma etapa do algoritmo CVM. A linha continua representa a etapa t do CVM, enquanto que a linha tracejada representa a etapa $t+1$. O ponto $\bar{z}$ representa o ponto mais distante do centro na etapa t. Os pontos em vermelho representam aqueles que já foram definidos como *core vectors*.



Fonte: Adaptado de STRACK, 2013

3.2 *BALL VECTOR MACHINE*

O *Core Vector Machine* apresentou experimentalmente ótimos resultados para classificação, entretanto, apesar de apresentar uma ideia simples, em cada iteração de seu algoritmo, ainda está envolvido um subproblema de programação quadrática no momento de construção do *coreset*, sendo assim ainda necessário um sofisticado solucionador numérico para eficiente implementação (TSANG; KOCSOR; KWOK, 2007).

Pensando nisso, TSANG; KOCSOR; KWOK propuseram uma nova técnica construída a partir do CVM, onde ao fixar o valor do raio da bola, não será necessária a utilização de métodos numéricos, sendo chamada de *Simpler Enclosing Ball (EB)* ou ainda *Ball Vector Machine (BVM)*.

A utilização do *Ball Vector Machine* produz resultados próximos ao *Core Vector Machine*, pois para um valor conveniente de η, o valor do centro encontrado para o EB é próximo ao valor de centro encontrado para o MEB. Além disso, para bases de dados cada vez maiores, os resultados vistos para o BVM conseguem ser superiores aos resultados obtidos com a utilização do CVM, dado que o tamanho do *coreset* construído durante a execução do *Core Vector Machine* acaba também possuindo um grande tamanho para bases de dados muito

grandes.

O algoritmo

Por se tratar de um caso especifico do CVM, onde o raio da esfera será fixo, a formulação do modelo SVM como um MEB obtida anteriormente ainda é valida. Entretanto, existem algumas pequenas diferenças no algoritmo utilizado para resolver agora o problema EB. São definidos então o centro da bola na t-ésima iteração e o raio fixo como c_t e r, respectivamente. Assim como para o CVM, para uma $\eta > 0$, o *Ball Vector Machine* irá funcionar da seguinte forma:

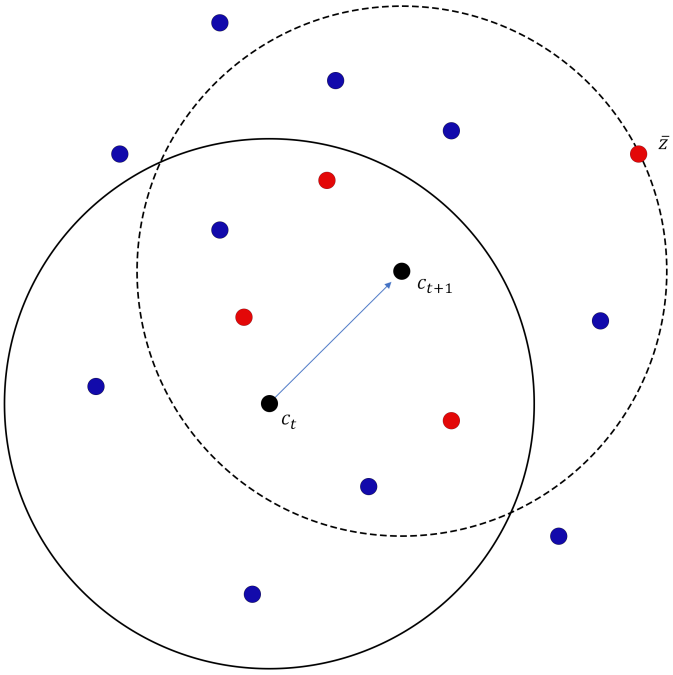
1. Inicialize c_0 e r;
2. Crie uma bola $B(c_t, (1+\eta)r)$, verifique a existência de pontos fora da bola e encerre se nenhum ponto for encontrado. Continue caso contrario;
3. Encontre o melhor valor para o centro de $B(c_{t+1}, r)$, de forma que um ponto $\bar{z}$ fora de $B(c_t, (1+\eta)r)$ toque a margem de $B(c_{t+1}, r)$;
4. Pare quando $c_t^2 \leq (2\eta + \eta^2)r^2$, e inclua $\bar{z}$ no *coreset* S;
5. Faça $r = \sqrt{(1+\eta)^2 r^2 - c_t^2}$ e retorne para o passo 2.

Assim como para o CVM, os valores iniciais são previamente definidos, porém, não precisamos definir os primeiros pontos que pertencerão ao primeiro *coreset* S, pois diferentemente do CVM, não é necessário captar o ponto mais distante do centro da bola, então, a primeira bola encontrada com raio fixo terá como base todos os pontos do conjunto de dados.

Após definir os valores, é então encontrada a primeira bola e se avalia a existência de pontos fora dela, parando o método interativo caso nenhuma esteja. Escolhe-se um ponto $\bar{z}$ fora da bola, o mais distante do centro, de forma que o novo centro para a próxima interação seja o melhor possível, garantindo ainda que o ponto selecionado esteja na margem de uma nova bola $B(c_{t+1}, r)$. O novo centro para a próxima interação é definido como descrito anteriormente, e atualiza-se também o valor do raio fixo.

Os pontos $\bar{z}$ selecionados formarão o *coreset*, e o método encerra quando todos os pontos estiverem dentro da última bola. Um passo do algoritmo do *Ball Vector Machine* pode ser visto na Figura 10.

Figura 10 – Uma etapa do algoritmo BVM. A esfera de raio fixo r é transladada até que um ponto $\tilde{z}$ fora da esfera toque a margem da nova esfera. A linha continua representa a etapa t do CVM, enquanto que a linha tracejada representa a etapa $t+1$. O ponto $\tilde{z}$ representa o ponto mais distante do centro na etapa t. Os pontos em vermelho representam aqueles que já foram definidos como *core vectors*.



Fonte: Adaptado de STRACK, 2013.

3.3 *SPHERE SUPPORT VECTOR MACHINE*

Por sua vez, o *Sphere SVM* (STRACK et al., 2013), é uma nova reformulação do *Ball Vector Machine*. Por isso, algumas partes dos dois algoritmos coincidem, como por exemplo o procedimento de inicialização e o critério de parada. Entretanto, apresenta também algumas diferenças, como a forma como o valor do centro é atualizada. O foco do *Sphere SVM* é direcionado para a eliminação de vetores de suporte dentro da esfera que os envolve, ao invés de encontrar amostras de dados periféricas. Para solucionar o problema MEB definido anteriormente, foi adotado pelos autores o algoritmo MDM (MITCHELL; DEM'YANOV; MALOZEMOV, 1974).

O algoritmo

O algoritmo base para o *Sphere SVM* é apresentado a seguir:

1. Inicialize $\alpha = 0$, $\alpha_0 = 1$, um valor fixo para o raio r, o centro c do MEB e $\eta = \frac{1}{2}$;

2. Selecione uma amostra X_r de X, e busque o primeiro vetor violador $\psi(\bar{x}_v)$, vetor mais distante do centro c;

3. Ainda com base em X_r, encontre o segundo vetor violador $\psi(\bar{x}_u)$, vetor mais próximo do centro c;

4. Calcule τ e β, valores utilizados para a atualização de c;

5. Atualize o valor de c e retorne ao passo 2.

Inicialmente se escolhe aleatoriamente um vetor de suporte e faz-se seu peso igual a 1, e então o raio $\hat{r} = \sqrt{k_{ii} + 1 + \frac{1}{C}}$ da esfera é calculado, onde k_{ii} é a norma quadrada dos vetores $\bar{x}_i$ no espaço de características induzido pelo *kernel* $k_{ij} = \phi(\bar{x}_i) \cdot \phi(\bar{x}_j)$. Apesar de $\hat{r} \geq r$, para conjuntos de dados com alta dimensionalidade e espaço de características grande, essa diferença é insignificante. Em cada interação dois vetores de suporte, que são chamados de vetores violadores, serão selecionados.

Primeiro, um subconjunto de todos os pontos X_r é sorteado, e então um vetor $\psi(x_v)$ dos vetores $\phi(\bar{x}_i) \in X_r$ é escolhido, de forma que a distância de $\psi(x_v)$ para o centro c da bola é maior que $(1 + \eta)\hat{r}$. Se este vetor não for encontrado dentro do subconjunto X_r, então outro subconjunto é sorteado, e se procura novamente por $\psi(x_v)$. Esse processo pode ser repetido n vezes, com n definido de forma arbitrária. Se após n tentativas nenhum vetor $\psi(x_v)$ for encontrado, então diminui-se o valor de η.

Após se encontrar $\psi(x_v)$, inicia-se a busca pelo segundo vetor violador, $\psi(x_u)$. Enquanto $\psi(x_v)$ representa um vetor de suporte fora da bola, o vetor $\psi(x_u)$ será o vetor mais próximo do centro c, ou seja, $||c - \phi(\bar{x}_i)||^2|$ é mínimo. Com estes dois vetores em mãos, o próximo passo é atualizar o centro da esfera, de forma que o novo centro é deslocado através da linha que conecta os dois vetores violadores (Vide Figura 11):

$$c' = c + \beta(\psi(\bar{x}_v) - \psi(\bar{x}_u)).$$

O coeficiente β é selecionado de forma que a nova esfera centrada em c' possua o vetor violador $\psi(\bar{x}_v)$ na fronteira. O coeficiente β é escrito como:

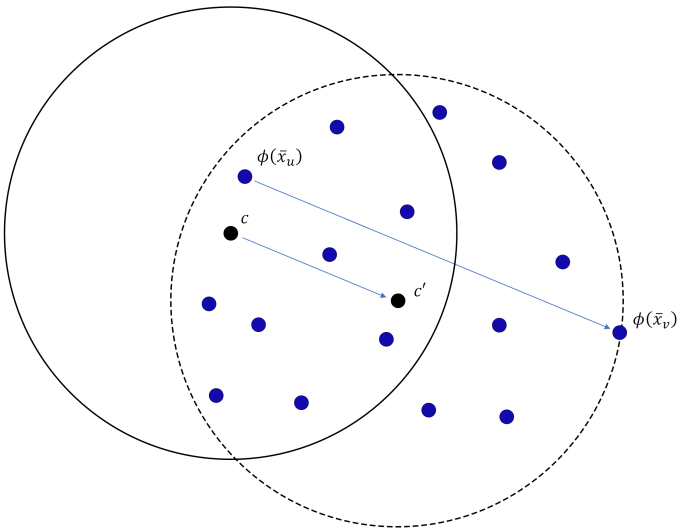
$$\beta = \tau - \sqrt{\tau^2 - \frac{||\psi(\bar{x}_v) - c||^2 - r^2}{||\psi(\bar{x}_v) - \psi(\bar{x}_u)||^2}}$$

onde τ é dado por:

$$\tau = \frac{(\psi(\bar{x}_v) - \psi(\bar{x}_u)) \cdot (\psi(\bar{x}_v) - c)}{||\psi(\bar{x}_v) - \psi(\bar{x}_u)||^2}.$$

Pensando no espaço *dual*, a atualização do centro é equivalente a aumentar o valor do multiplicador de Lagrange correspondente α_v por β e reduzir o valor do multiplicador de Lagrange correspondente α_u por β. É importante manter as condições dos multiplicadores de Lagrange satisfeita, com atenção especial a condição de não negatividade. Portanto, é preciso que $\beta \leq 1 - \alpha_v$ e $\beta \leq \alpha_u$. Possuindo o valor β, é possível atualizar o centro da esfera e retomar o algoritmo verificando o critério de parada e procurando outros vetores.

Figura 11 – Uma etapa do algoritmo *Sphere SVM*. O centro c é deslocado ao longo do vetor $\psi(\bar{x}_v) - \psi(\bar{x}_u)$ para uma nova posição c'. A linha continua representa a etapa t do CVM, enquanto que a linha tracejada representa a etapa $t+1$. O ponto $\bar{z}$ representa o ponto mais distante do centro na etapa t.



Fonte: Adaptado de STRACK et al., 2013.

Seguindo a ideia de *coresets*, o *coreset* final é formado pelo conjunto de vetores violadores $\psi(\bar{x}_v)$. Uma etapa do algoritmo utilizado para o *Sphere Support Vector Machine* pode ser visualizada na Figura 11.

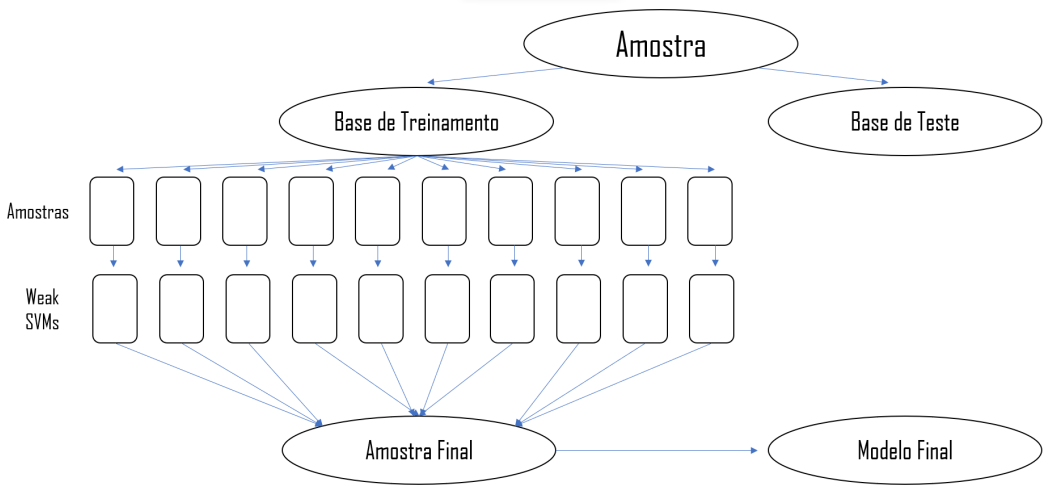
3.4 SPEED UP SVM PARA CLASSIFICAÇÃO

Proposto inicialmente por (WANG et al., 2014), o modelo *Speed Up SVM* é uma dentre muitas técnicas que optam pela seleção de amostras de dados, como forma de aceleração do modelo SVM, sendo essa seleção possível a partir da utilização dos *weak SVMs*. A utilização dos *Weak SVMs* parte da ideia de que por serem modelos construídos utilizando menos observações, em comparação com a base de treinamento completa ($n << N$), os modelos seriam

mais fracos, por possuírem menos informações para basear a sua previsão.

Com isso em mente, o modelo foi adaptado e utilizado no contexto da regressão por (PIMENTEL; OSPINA; ARA, 2021) para aplicação numa base de dados de educação onde o objetivo consistia na utilização do modelo SVM para uma previsão *online*. Porém, com o objetivo de utiliza-lo para a classificação de observações, o modelo será aqui descrito com essa intenção. A ideia completa do modelo será descrita a seguir, sendo o processo completo descrito na Figura 12.

Figura 12 – Fluxograma da descrição do modelo *Speed Up SVM* para classificação com a apresentação dos passos necessários para a seleção de observações para a amostra final.



Fonte: Autor.

Amostragem Inicial e construção dos Weak

Partindo da totalidade dos dados que serão utilizados para o treinamento do modelo, são selecionadas K amostras de tamanho n sem reposição. Na sequência, para cada uma das K amostras, será construído um modelo SVM, sendo estes modelos chamados de *Weak SVMs*.

Seleção das observações

Com os K *Weak SVMs* construídos, eles serão utilizados para prever todas as observações que pertencem a nossa base de treinamento original, construindo assim um conjunto de K previsões para cada observação presente na base de treinamento. A partir destas previsões, será calculada então a variância para cada observação, de maneira a ordenar as observações

pertencentes a base de treinamento pela variabilidade das previsões.

Vale ser ressaltado que a ideia por trás da utilização do *Speed Up SVM* é a detecção das observações com maior chance de serem os vetores de suporte. Com isso em mente, espera-se que as observações com maior variância durante as K previsões sejam essas candidatas. Uma alta variância durante as previsões significa dizer que as observações foram classificadas de forma diferente durante os K modelos fracos construídos, indicando assim sua proximidade ao hiperplano e margens responsáveis pela classificação do modelo, e sugerindo uma maior chance de ser um vetor de suporte. Observações com baixa variabilidade de previsão estão localizadas distante do hiperplano, possuindo assim baixa probabilidade de ser um vetor de suporte.

Construção da amostra final

A partir da base de treinamento ordenada de acordo com a variância existente durante as K previsões realizadas pelos *Weak SVMs*, serão então selecionadas as observações a fazerem parte da amostra final que será utilizada para estimar o modelo SVM, utilizando-se assim de menos observações.

Para esta seleção, tomou-se como ponto de corte o terceiro quantil das variâncias calculadas para todas as observações, mantendo assim na base de dados as observações com as 25% maiores variâncias. Com esta base construída, ela é utilizada para estimar o modelo SVM.

3.5 FUSION SUPPORT VECTOR MACHINE

Na literatura, dentre os métodos existentes, o modelo descrito na seção 3.4 apresenta características interessantes que podem ser exploradas a partir de outras abordagens, sendo uma delas a utilização de SVMs fracos, os *Weak SVMs*.

Além disso, durante as primeiras simulações e experimentos utilizando a teoria de *coresets*, observou-se que os mesmos são construídos de maneira rápida independente do tamanho da base de dados, além do que, sua teoria garante que mesmo pequenos conjuntos podem apresentar boas previsões, superiores aos *Weak SVMs*, sendo assim modelos mais 'fortes'.

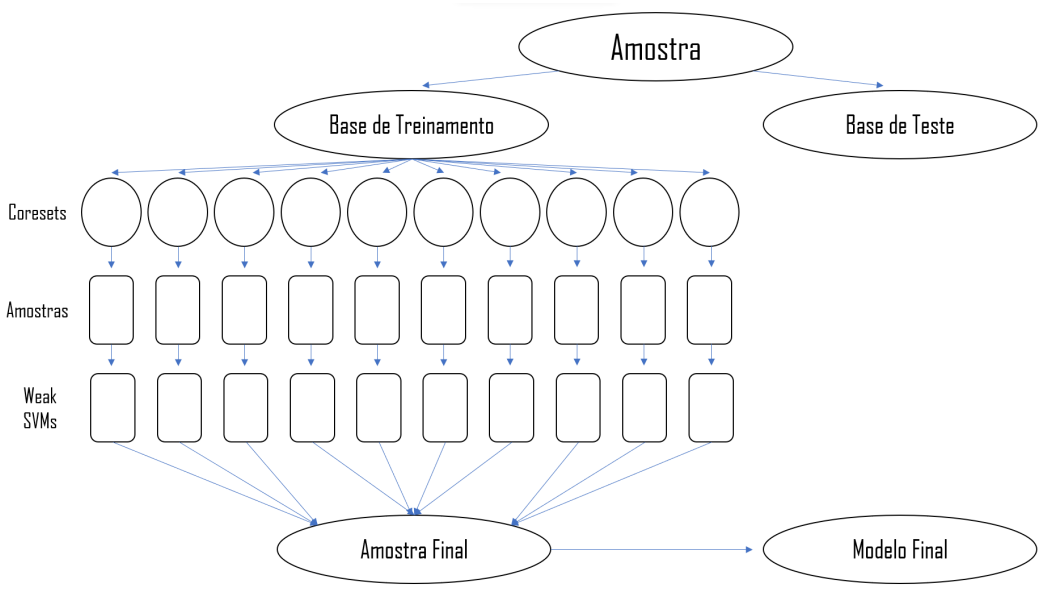
Por consequência, foi pensada a utilização de SVMs fracos e esféricos, unindo os *Weak SVMs* com os *coresets*, para criar assim o modelo *Fusion SVM*. Nas próximas subseções serão apresentados os dois modelos propostos a partir da união do modelo *Speed Up SVM*

(PIMENTEL; OSPINA; ARA, 2021) e do modelo anteriormente apresentado *Sphere SVM* (STRACK et al., 2013).

3.5.1 Fusion W SVM

A ideia base por trás do modelo proposto, consiste na utilização de *coresets* anteriormente a fase de seleção de amostras durante o processo do modelo *Speed Up SVM*. A ideia completa do modelo será descrita a seguir, sendo o processo completo descrito na Figura 13.

Figura 13 – Fluxograma da descrição do modelo *Fusion W SVM* com a apresentação dos passos necessários para a seleção de observações para a amostra final.



Fonte: Autor.

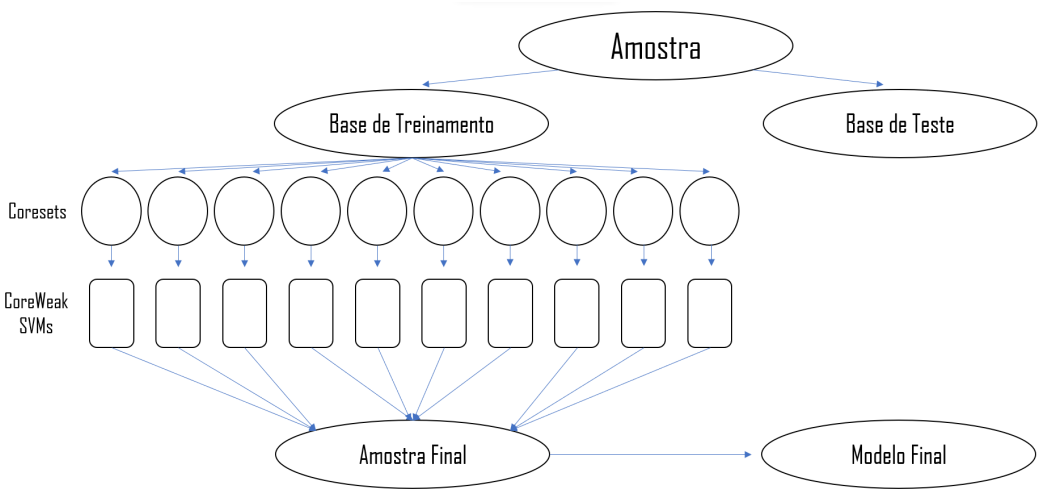
Assim como para o modelo *Speed Up SVM*, partimos da totalidade dos dados que serão utilizados para o treinamento do modelo, entretanto, diferentemente do modelo *Speed Up SVM*, são primeiro construídos K *coresets* de tamanho h. Na sequência, para cada um dos K *coresets* construídos, será tomada uma amostra de tamanho n, e então por fim serão estimados os modelos *Weak SVM*.

Na sequência, são repetidos os mesmos passos definidos na seção 3.4, com a previsão de todas as observações da base de treinamento original, cálculo da variância para cada observação, e seleção das observações que farão parte da amostra final, que será utilizada para estimação do modelo final.

3.5.2 Fusion CW SVM

Seguindo a ideia recém apresentada, foi pensado ainda uma pequena variação no modelo *Fusion SVM*, de forma que o modelo possa se tornar ainda mais rápido. Novamente a ideia deste segundo modelo proposto é descrito na Figura 14.

Figura 14 – Fluxograma da descrição do modelo *Fusion CW SVM* com a apresentação dos passos necessários para a seleção de observações para a amostra final.



Fonte: Autor.

A sutil mudança foi realizada durante o primeiro processo de criação de amostras, substituindo a construção de *coresets* e depois seleção de amostras, pela simples construção de *coresets* do tamanho da amostra que seria tomada. Ao eliminar esta etapa, os *Weak SVMs* serão construídos diretamente a partir dos *coresets*, passando então a serem chamados de *CoreWeak*. Dessa maneira, o primeiro modelo será chamado de *Fusion W SVM* dada a utilização de *Weak*, e o segundo modelo proposto será chamado de *Fusion CW SVM* dada a utilização dos *CoreWeak*.

4 ESTUDO DE SIMULAÇÃO

Neste capitulo serão apresentados os resultados obtidos através de uma série de experimentos realizados em bases de dados simuladas. Os modelos a serem aqui comparados são:

1. **SVM:** Implementação padrão do modelo SVM;
2. **SPEED UP SVM:** Implementação em R do modelo adaptado *Speed Up SVM*;
3. **SPHERE SVM:** Implementação em R do modelo *Sphere SVM*;
4. **FUSION W SVM:** Implementação em R do modelo proposto *Fusion W SVM*;
5. **FUSION CW SVM:** Implementação em R do modelo proposto *Fusion CW SVM*.

O SVM corresponde ao modelo LIBSVM altamente utilizado pela comunidade científica quando trabalha-se com as Máquinas de Vetores de Suporte (CHANG; LIN, 2011). A implementação do modelo *Speed Up SVM* corresponde ao modelo apresentado na seção 3.4, uma adaptação para a classificação do modelo apresentado por (PIMENTEL; OSPINA; ARA, 2021) para o contexto da regressão. A implementação *Sphere SVM* retrata o modelo apresentado na seção 3.3 proposto por (STRACK et al., 2013), com alterações na forma de definir os valores de centro e raio da esfera inicial. As duas últimas implementações, *Fusion W* e *Fusion CW* correspondem ao modelo proposto na seção 3.5 em suas duas versões, com a utilização de *Weak SVMs* na primeira versão e *CoreWeak SVMs* na segunda versão. Todos os parâmetros utilizados durante os experimentos para as diferentes bases de dados são encontrados na Tabela 6.

Tabela 6 – Parâmetros padrões utilizados durante os experimentos com dados simulados

Modelos	Parâmetros
SVM	Padrão KERNLAB
Speed Up SVM	$K = 10$, $n = 1000$
Sphere SVM	$m = 50$, h: Variável, $\eta = 0.5$
Fusion W SVM	$m = 50$, $K = 10$, $h = 2500$, $n = 1000$, $\eta = 0.5$
Fusion CW SVM	$m = 50$, $K = 10$, $h = 1000$, $\eta = 0.5$

Fonte: Autor.

K: N° de *Weak SVMs*. n: Tamanho da amostra do *Weak SVM*.
m: Amostra inicial. h: Tamanho do *coreset*.

Para o modelo SVM, foram utilizados os paramêtros padrão do pacote KERNLAB (KARATZOGLOU et al., 2004). Para o modelo *Speed Up SVM*, foram utilizados os mesmos paramêtros

apresentados por (PIMENTEL; OSPINA; ARA, 2021), sendo o valor de n extendido ainda para o modelo *Fusion W SVM*. Para os modelos *Sphere SVM*, *Fusion W* e *Fusion CW*, foi fixado o valor de η definido por (STRACK et al., 2013). Em contrapartida, o valor da amostra inicial m foi obtido após a realização de testes iniciais que definiram o valor mais apropriado. Por fim, o tamanho do *coreset* foi fixado em $h = 1000$ para o modelo *Fusion CW SVM*, de forma a preservar o mesmo tamanho das amostras utilizadas pelos modelos 1 e 4, e fixado em $h = 2500$ para o modelo *Fusion W SVM* por questões computacionais.

Na tabela 7 são apresentadas as características gerais de cada conjunto de dados utilizados.

Tabela 7 – Conjuntos de dados simulados utilizados

Conjunto de dados	Total de observações	Variáveis
Base de dados simulados de forma independente	1,000,000	4
Base de dados simulados com dependência	1,000,000	5
Base de dados aumentada simulados com dependência	1,000,000	25
Checkerboard	3,308,761	3

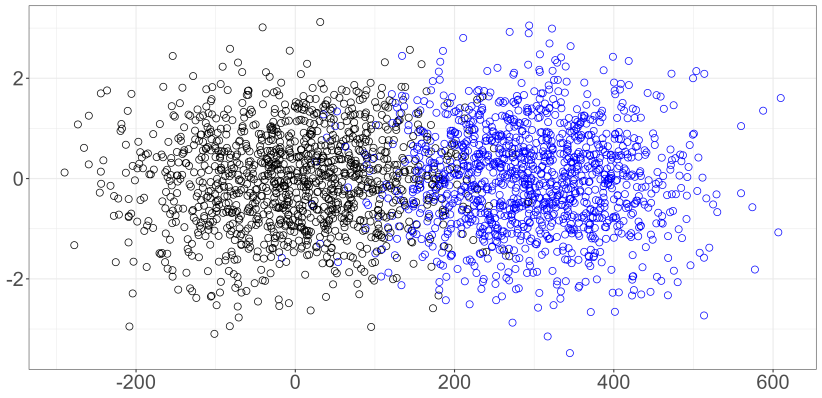
Fonte: Autor.

Todos os experimentos foram realizados utilizando a linguagem R (versão 4.2.2) (TEAM et al., 2013) em um computador pessoal com processador `2.50GHz Intel Core i5-10300H`, `8GB RAM` de memória e um Sistema operacional `Windows 11 de 64 bits`. Para um resultado mais sólido, em cada base de dados serão utilizadas diferentes tamanhos de população (100 mil, 300 mil, 500 mil, 1 milhão e 2 milhões), sendo utilizado ainda o *handout* repetido 10 vezes com uma proporção de $80/20$ para as bases de tamanho 100 e 300 mil, e $90/10$ para os demais tamanhos de população.

4.1 BASE DE DADOS INDEPENDENTES

Dando inicio aos experimentos, a primeira base de dados utilizada foi gerada a partir de 3 variáveis aleatórias com distribuição normal, sendo $X_2, X_3 \sim N(0, 1)$, e a variável aleatória $X_1 \sim N(0, 100)$ para a primeira classe $(Y = 1)$ e $X_1 \sim N(300, 100)$ para a segunda classe $(Y = 2)$, sendo Y a variável a ser classificada. Uma representação destes dados é apresentada na Figura 15. Foram geradas artificialmente um total de 1 milhão de observações, e durante os experimentos, os modelos foram avaliados para 4 diferentes tamanhos de população, 100 mil, 300 mil, 500 mil e 1 milhão de observações.

Figura 15 – Representação dos dados simulados de forma independente.

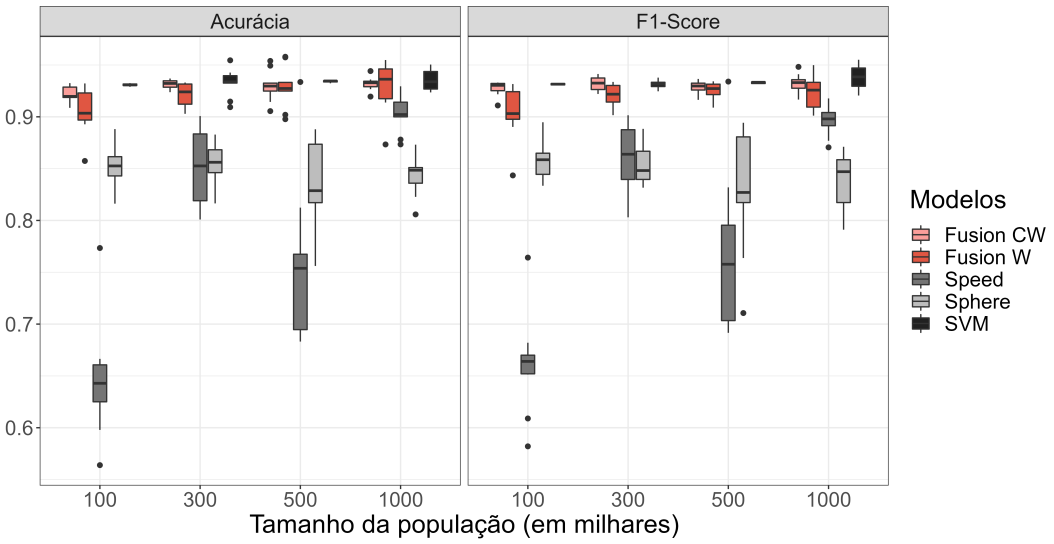


Fonte: Autor.

Inicialmente avaliou-se a capacidade preditiva dos modelos utilizados, sempre comparando com o modelo SVM completo.

Com base na Figura 16, nota-se que o comportamento dos modelos é bastante próximo para os diferentes tamanhos de população, com exceção ao modelo *Speed Up SVM* que apresentou métricas bem abaixo do modelo SVM para as populações de tamanho 100 e 500 mil, uma alta variação para 300 mil observações, se aproximando mais dos resultados dos demais modelos ao se utilizar 1 milhão de observações.

Figura 16 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados via simulação em dados independentes.

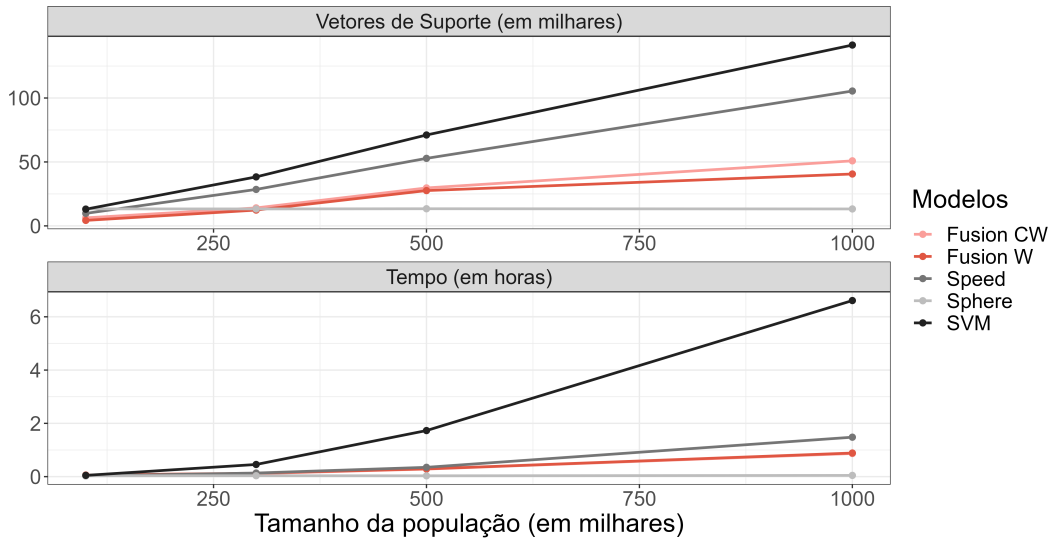


Fonte: Autor.

Por sua vez, quando avaliamos o tempo médio necessário para a estimação dos modelos propostos frente ao modelo SVM (Vide Figura 17), podemos destacar o modelo *Sphere SVM*

que apresenta um tempo constante, com minímas alterações com o aumentar do tamanho da população, ao mesmo tempo que consegue estimar o modelo utilizando menos vetores de suporte. Da mesma forma, observa-se que os modelos *Fusion CW SVM* e *Fusion W SVM* conseguem reduzir ainda mais o tempo médio utilizado para a estimação quando comparado com o modelo *Speed Up SVM*, assim como estimar o modelo com menos vetores de suporte.

Figura 17 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados via simulação em dados independentes.



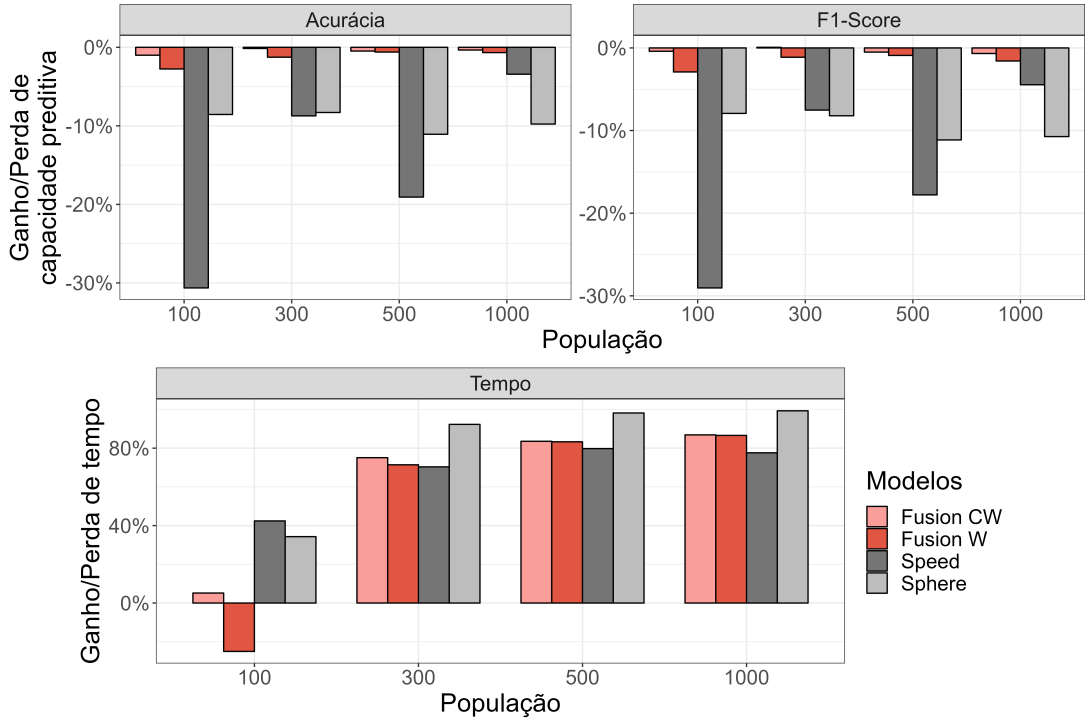
Fonte: Autor.

Por fim, com base na Figura 18, podemos visualizar e avaliar o ganho de tempo médio e capacidade preditiva dos modelos. Como já evidenciado anteriormente, o modelo *Sphere SVM* é aquele que apresentou o maior ganho de tempo durante os experimentos, ganho esse porém que não foi acompanhado pela capacidade preditiva que apresenta redução próxima aos 10% durante os experimentos.

Por sua vez, com o aumentar do tamanho da base de dados, é possível observar que o melhor comportamento é alcançando pelos modelos *Fusion CW SVM* e *Fusion W SVM*, dada a perda mínima de capacidade preditiva e elevado ganho de tempo computacional durante a estimação do modelo.

Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 9 no Apêndice A.

Figura 18 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados via simulação em dados independentes.



Fonte: Autor.

4.2 BASE DE DADOS DEPENDENTES

Nossa segunda base foi gerada de forma semelhante a primeira utilizada, porém, inserindo alguma dependência nos dados (BREIMAN, 1996). Desta vez, foi gerada uma variável aleatória normal multivariada com média $\mu_1 = (0, \cdots, 0)_{1\times p}$ e matriz de covariância dada por

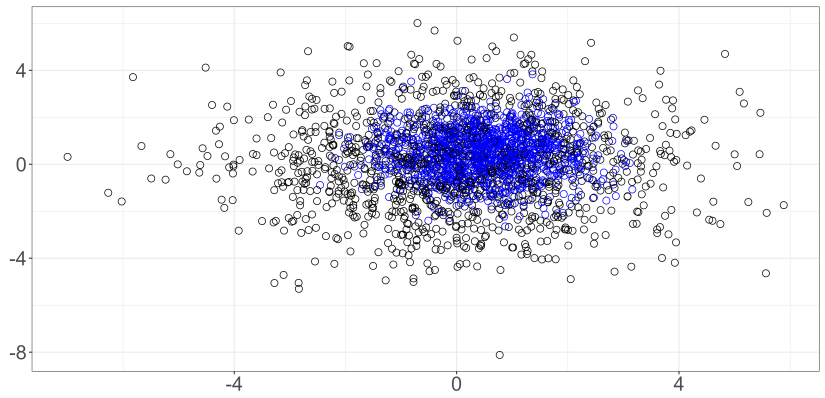
$$\Sigma_1 = \begin{pmatrix} 4 & 1 & \cdots & 0 & 0 \\ 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 4 & 1 \\ 0 & 0 & \cdots & 1 & 4 \end{pmatrix}_{p\times p}$$

para a primeira classe $(Y = 1)$ e uma variável aleatória normal multivariada com $\mu_2 = (\frac{1}{\sqrt{p}}, \cdots, \frac{1}{\sqrt{p}})_{1\times p}$ e matriz de covariância dada por

$$\Sigma_2 = \begin{pmatrix} 1 & 1 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & \cdots & 1 & 1 \end{pmatrix}_{p \times p}$$

para a segunda classe ($Y = 2$), sendo Y novamente a variável a ser classificada, sendo utilizado para esta simulação $p = 4$. Uma representação destes dados é apresentada na Figura 19.

Figura 19 – Representação dos dados simulados de forma dependente.



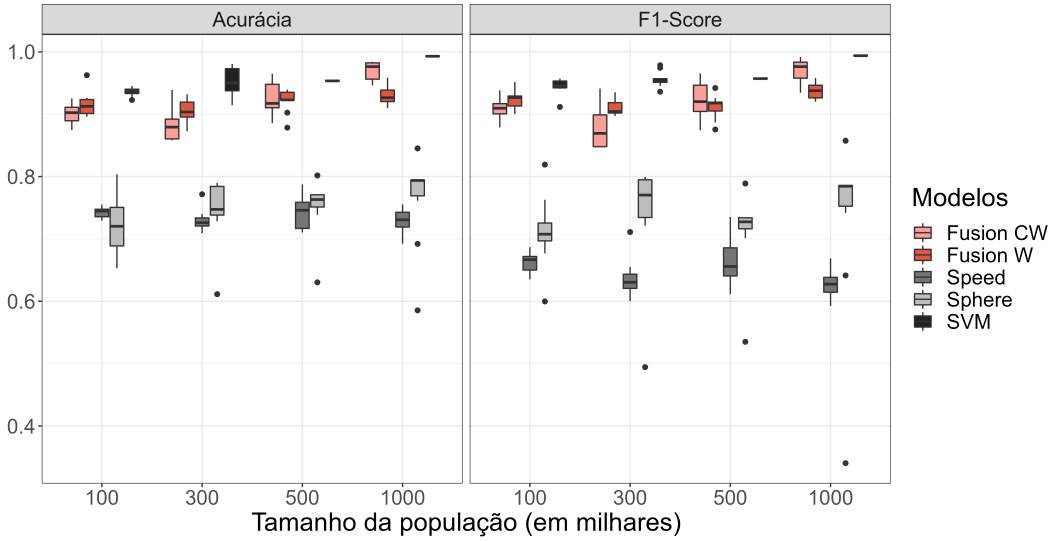
Fonte: Autor.

Foram geradas artificialmente um total de 1 milhão de observações, e durante os experimentos, os modelos foram avaliados para 4 diferentes tamanhos de população, 100 mil, 300 mil, 500 mil e 1 milhão de observações.

Começando a análise pela capacidade preditiva dos 5 modelos, com base na Figura 20 observa-se que apenas os dois modelos *Fusion* mantiveram desempenho próximo ao desempenho do modelo original, tendo os modelos *Sphere* e *Speed* alcançado valores de acurácia e *F1-Score* até 0.2 abaixo do atingido pelo modelo SVM.

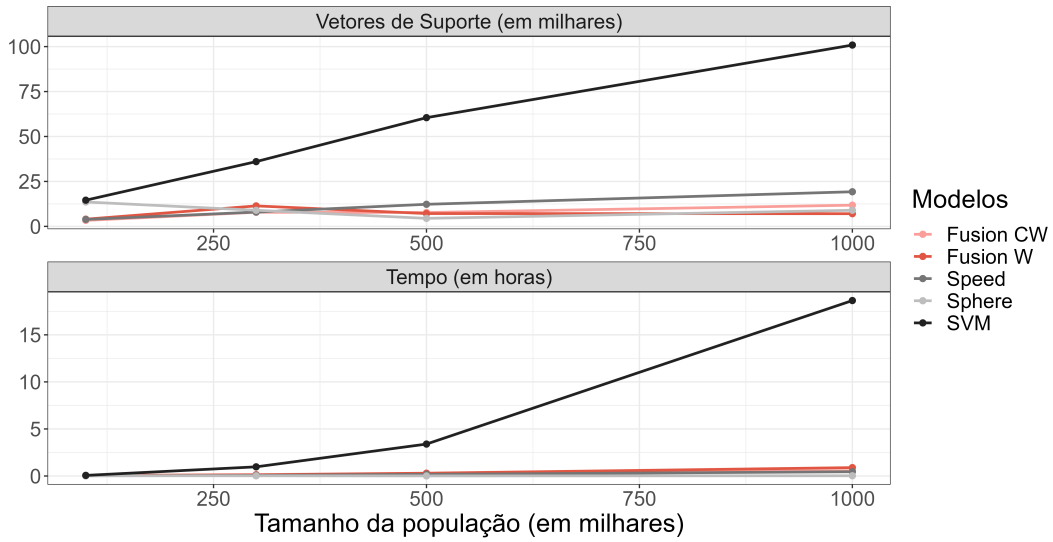
Avaliando o tempo médio necessário para a estimação do modelo (Vide Figura 21), nota-se que todos os 4 modelos implementados foram executados em tempo inferior ao modelo SVM e semelhantes entre si, apresentado assim quanto ao tempo 4 ótimas opções. Quando avalia-se o número de vetores de suporte é possível observar que apesar de reduzido para os 4 modelos, para o modelo *Speed Up SVM* foi utilizado quase duas vezes o número de vetores necessário para a estimação do *Fusion W SVM*.

Figura 20 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados via simulação em dados dependentes.



Fonte: Autor.

Figura 21 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados via simulação em dados dependentes.

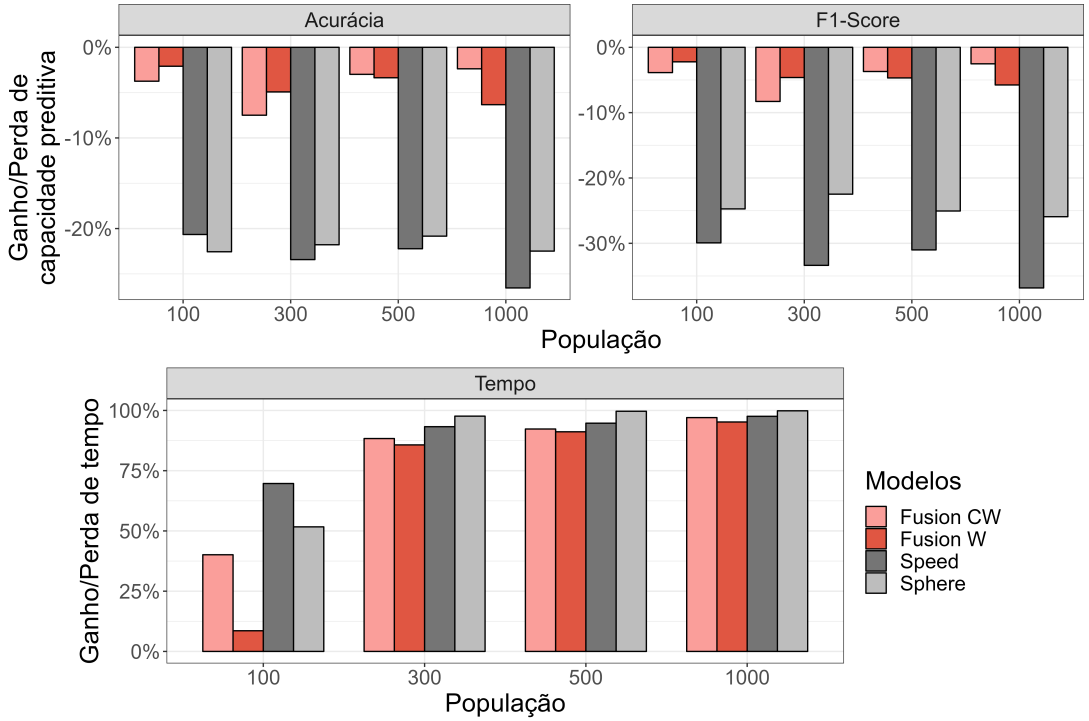


Fonte: Autor.

Por fim, atráves da Figura 22 é possível observar que os modelos *Fusion W SVM* e *Fusion CW SVM* apresentam o melhor custo beneficio dentre os 4 modelos avaliados, ao apresentar alto ganho de tempo (bastante próximo aos demais modelos) e uma perda de capacidade preditiva bastante reduzida se comparada aos demais modelos.

Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 10 no Apêndice A.

Figura 22 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados via simulação em dados dependentes.



Fonte: Autor.

4.3 BASE DE DADOS DEPENDENTES AUMENTADA

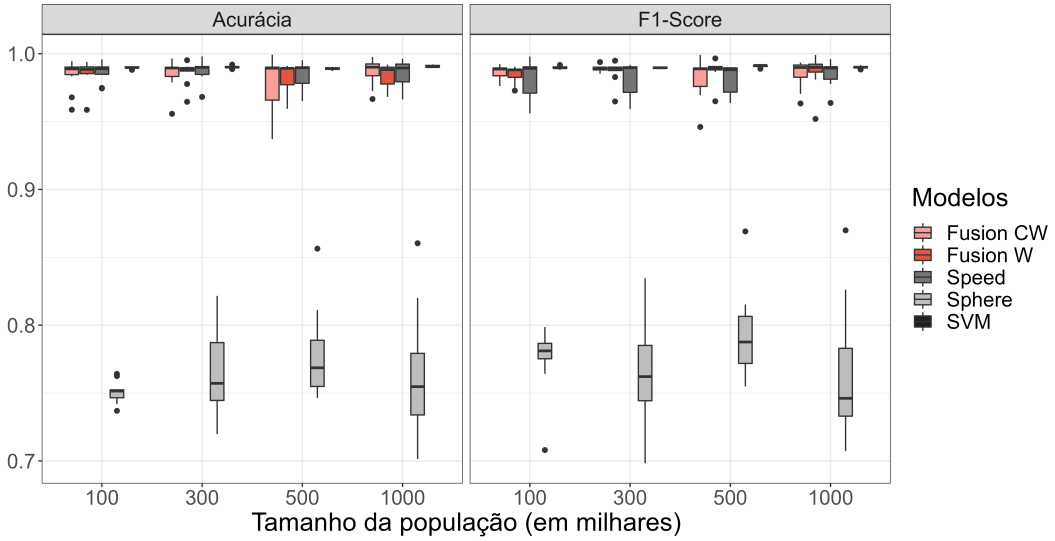
A terceira base de dados utilizada segue os moldes da base anterior, porém com a utilização de um número maior covariáveis, para assim avaliar o comportamento dos modelos implementados em situações com muitas variáveis explicativas. Para a base utilizada na seção 4.2, os dados foram gerados com um total de $p = 4$ covariáveis, enquanto que a base a ser aqui utilizada possui um total de $p = 24$ covariáveis.

Foram geradas artificialmente um total de 1 milhões de observações, e durante os experimentos, os modelos foram avaliados para 4 diferentes tamanhos de população, 100 mil, 300 mil, 500 mil e 1 milhão de observações.

Avaliando a capacidade preditiva dos 5 modelos, diferentemente do observado para apenas 4 covariáveis, o comportamento dos modelos é muito próximo ao comportamento do modelo SVM, com exceção ao modelo *Sphere* que apresenta uma capacidade preditiva inferior (Vide Figura 23).

Ao avaliar o tempo médio e número de vetores de suporte, foi observado um comportamento diferente ao ocorrido nos dois experimentos anteriores (Vide Figura 24). Quanto ao

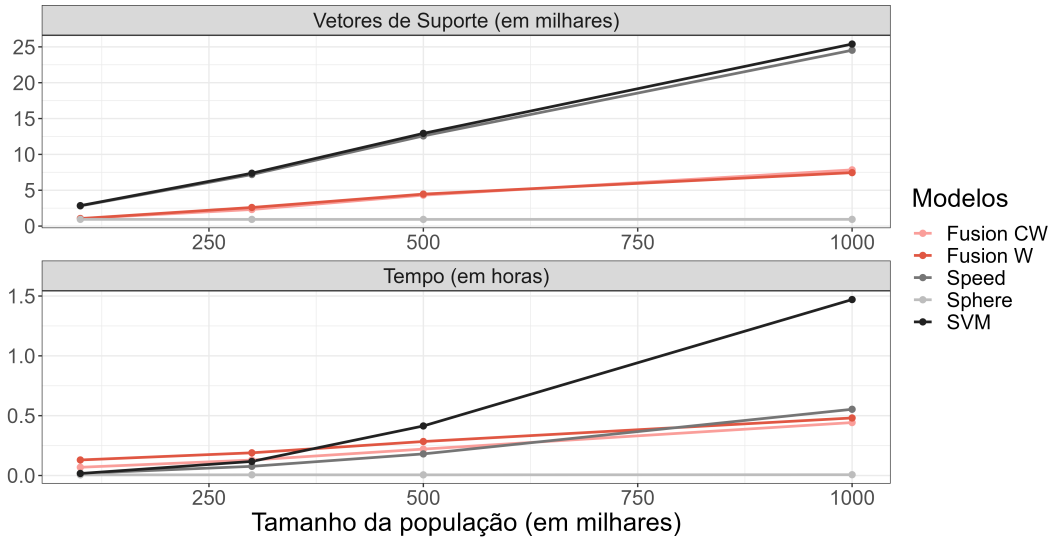
Figura 23 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados via simulação em dados dependentes com um maior número de covariáveis.



Fonte: Autor.

tempo, apenas quando as bases superaram 500 mil observações, que se mostrava interessante a utilização dos 4 modelos implementados. Já para o número de vetores de suporte, ainda se observou um valor reduzido para 3 modelos, entretanto, o modelo *Speed Up SVM* acompanhou de forma próxima o número de vetores de suporte do modelo SVM.

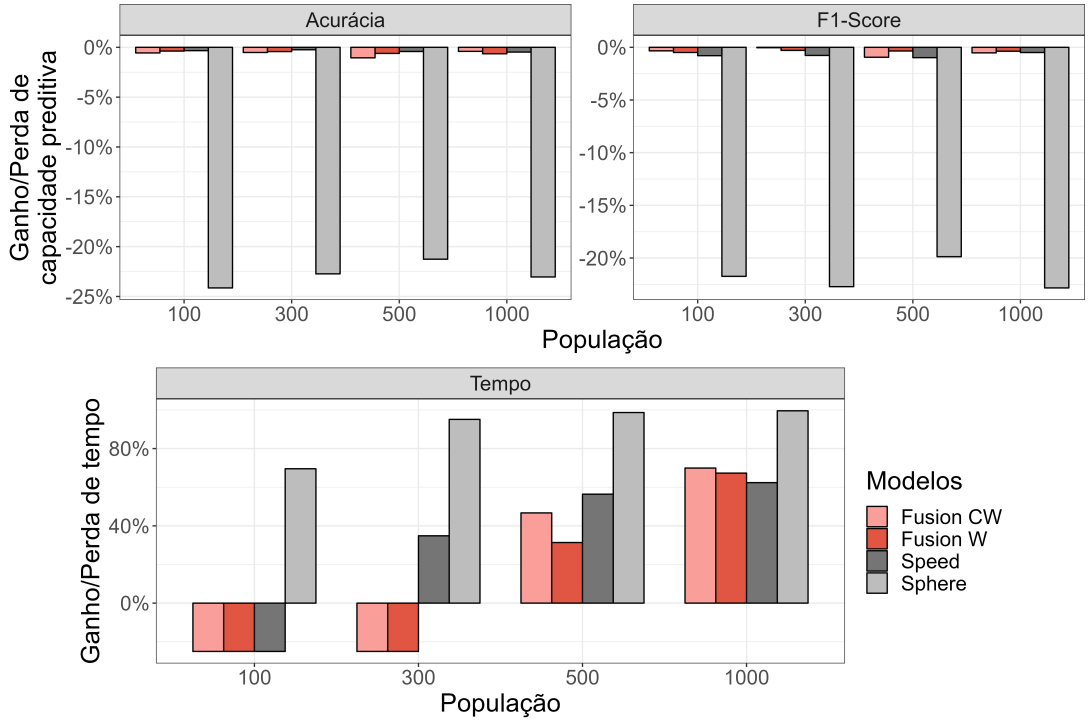
Figura 24 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados via simulação em dados dependentes com um maior número de covariáveis.



Fonte: Autor.

Para finalizar, ao observar o ganho/perda de tempo médio e capacidade preditiva dos modelos implementados, com base na Figura 25, é possível observar que o modelo *Sphere*

Figura 25 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados via simulação em dados dependentes com um maior número de covariáveis.



Fonte: Autor.

SVM é o único a apresentar elevada perda de capacidade preditiva, superando sempre em média os 20%. Quando observa-se o tempo, é possível notar um melhor comportamento do modelo *Speed Up SVM*, sendo superado pelos modelos *Fusion* apenas para a maior base de dados, 1 milhão de observações.

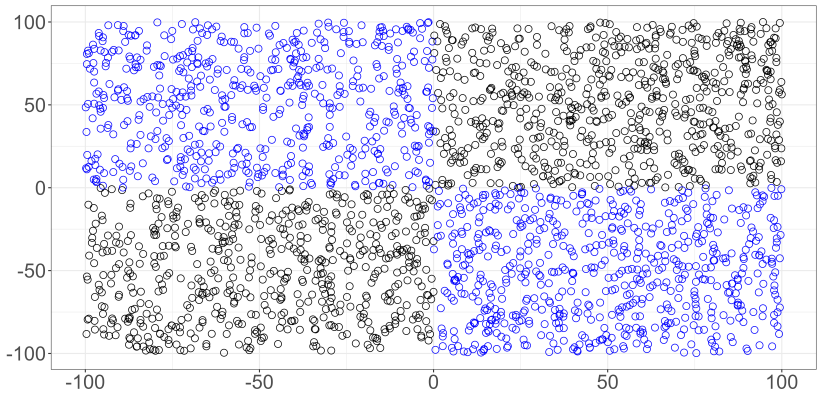
Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 11 no Apêndice A.

4.4 *CHECKERBOARD* 2X2

A quarta e última base de dados simulada a ser utilizada durante os experimentos, é a famosa base de dados *checkerboard* (TSANG et al., 2005; WU; CHANG, 2003), que como o próprio nome sugere, apresenta dados simulados que formam graficamente um tabuleiro de damas (Vide Figura 26).

Para este último teste com dados simulados, foram gerados artificialmente um pouco mais de 3 milhões de observações, valor este muito acima do necessário para uma boa classificação com estes dados, porém, necessário para que fosse possível avaliar a qualidade dos modelos

Figura 26 – Representação da base de dados simulada *Checkerboard* $2x2$.



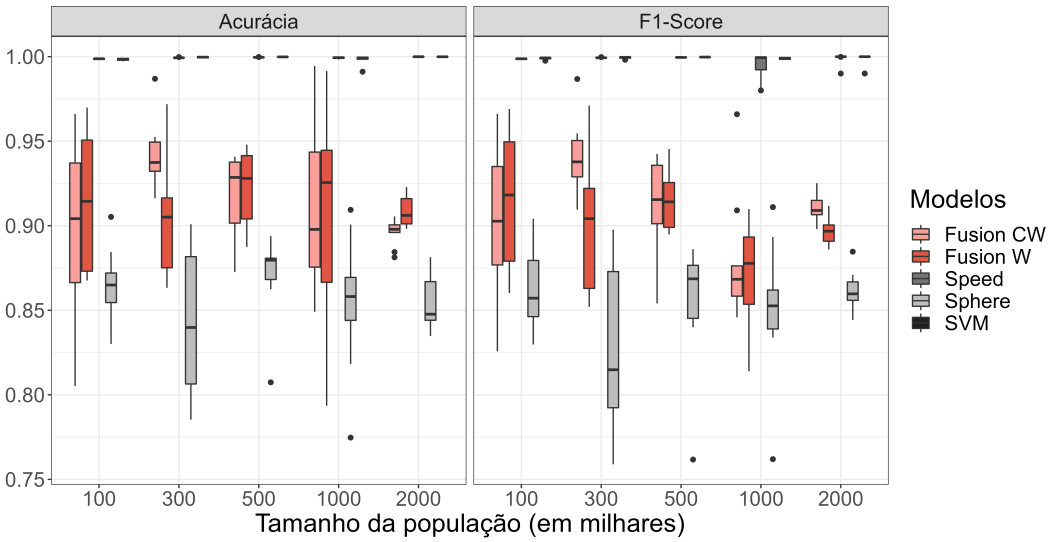
Fonte: Autor.

apresentados para grandes bases de dados, o principal objetivo deste estudo.

Os modelos foram avaliados para 5 diferentes tamanhos de população, 100 mil, 300 mil, 500 mil, 1 milhão e 2 milhões de observações. Por motivo de limitação computacional e do tempo necessário, não foi possível a estimação do modelo SVM para a totalidade de observações geradas.

Com base na Figura 27 é possível observar que para esta última base de dados, os modelos SVM e *Speed Up SVM* atingiram uma classificação quase perfeita. Logo atrás vem os modelos *Fusion*, com uma maior variação nas predições, porém, com ainda ótima capacidade preditiva.

Figura 27 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados utilizando a base de dados simulada *Checkerboard* $2x2$.

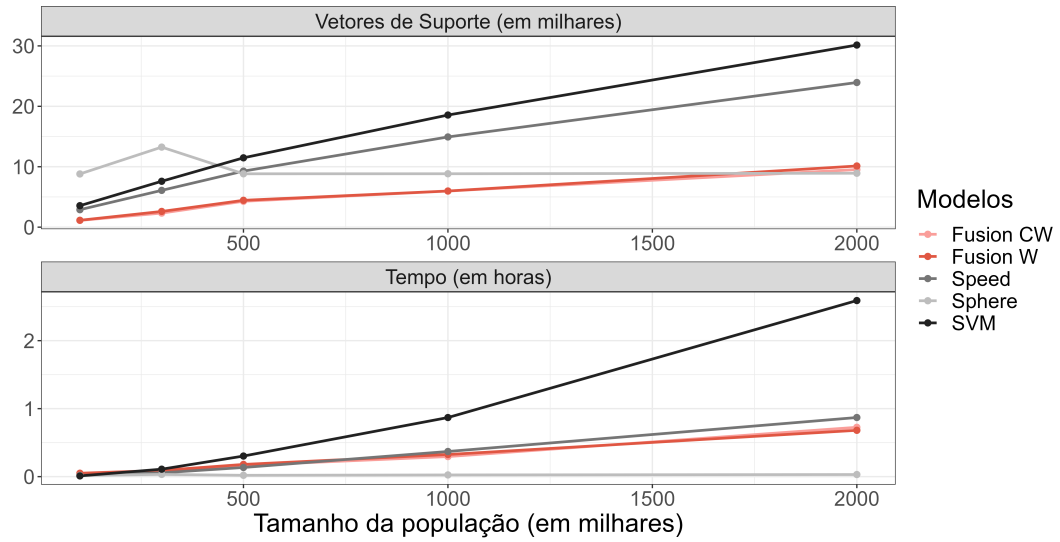


Fonte: Autor.

Avaliando o tempo médio e número de vetores de suporte é encontrado comportamento

semelhante ao visto para a base anterior. O modelo *Speed Up SVM* mais uma vez apresenta um número de vetores de suporte próximos ao do modelo SVM, e quando avaliado ao tempo médio, se mantem ao longo dos diferentes tamanhos de bases de dados próximo aos modelos *Fusion*, sendo superado quando a base de dados supera o total de 1 milhão de observações (Vide Figura 28).

Figura 28 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados utilizando a base de dados simulada *Checkerboard* $2x2$.

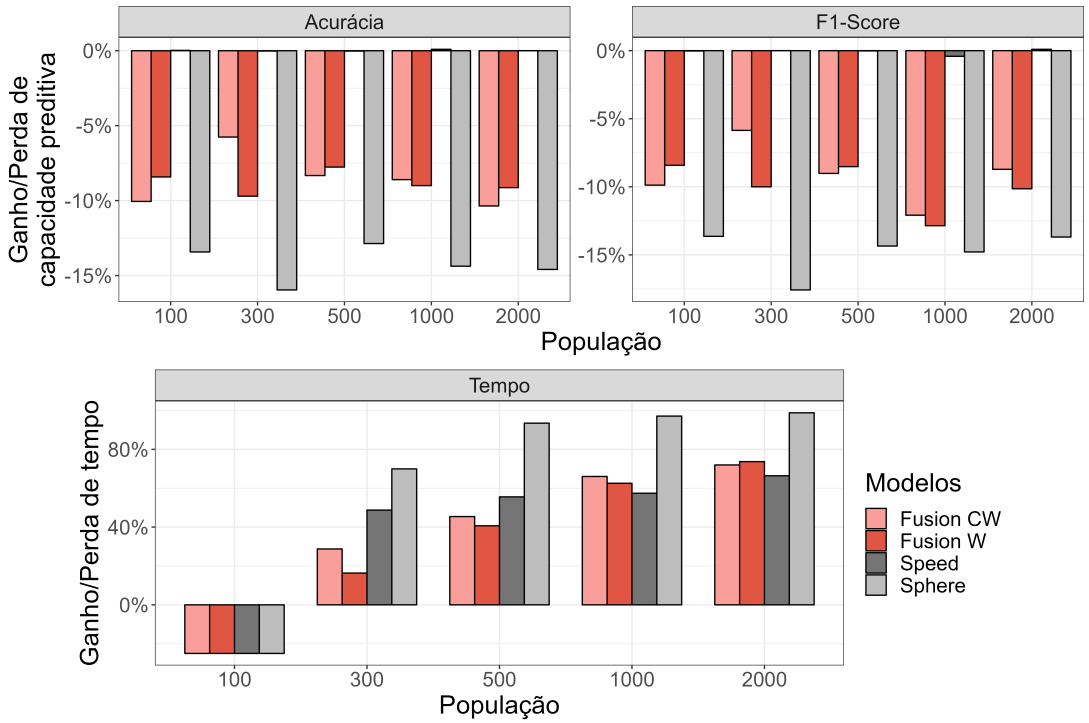


Fonte: Autor.

Para finalizar, assim como já era possível observar nos demais gráficos, o modelo *Speed Up SVM* é o único a não apresentar perda de capacidade preditiva em comparação ao modelo SVM, e apresenta bom ganho de tempo médio para bases de dados 'menores', porém, sendo ultrapassado pelos modelos *Fusion W SVM* e *Fusion CW SVM* para bases de dados cada vez maiores.

Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 12 no Apêndice A

Figura 29 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados utilizando a base de dados simulada *Checkerboard* $2x2$.



Fonte: Autor.

5 EXPERIMENTOS EM DADOS REAIS

Após uma completa análise utilizando-se apenas de bases de dados simuladas, para uma melhor avaliação dos modelos, serão também também utilizadas 4 bases de dados reais, sendo todas elas coletadas do site *Kaggle* - <https://www.kaggle.com/>. Algumas informações sobre as bases a serem utilizadas estão presentes na Tabela 8.

Tabela 8 – Conjuntos de dados reais utilizados durante experimentos

Conjunto de dados	Total de observações	Variáveis
Car Sales Russia	1,159,274	6
Heart Disease Health	500,000	22
KDDCUP-99 *Intrusion Detection*	1,945,562	17
Targeted Marketing	2,000,000	18

Fonte: Autor.

Assim como durante o processo de simulação, os modelos a serem comparados são apresentados logo abaixo:

1. **SVM:** Implementação padrão do modelo SVM;
2. **SPEED UP SVM:** Implementação em R do modelo adaptado *Speed Up SVM*;
3. **SPHERE SVM:** Implementação em R do modelo *Sphere SVM*;
4. **FUSION W SVM:** Implementação em R do modelo proposto *Fusion W SVM*;
5. **FUSION CW SVM:** Implementação em R do modelo proposto *Fusion CW SVM*.

Os paramêtros a serem utilizados durante os experimentos com bases de dados reais são os mesmos utilizados com as bases de dados simuladas, e podem ser vistos na Tabela 6. Todos os experimentos foram realizados utilizando a linguagem R (versão 4.2.2) (TEAM et al., 2013) em um computador pessoal com processador `2.50GHz Intel Core i5-10300H, 8GB RAM` de memória e um Sistema operacional `Windows 11 de 64 bits`.

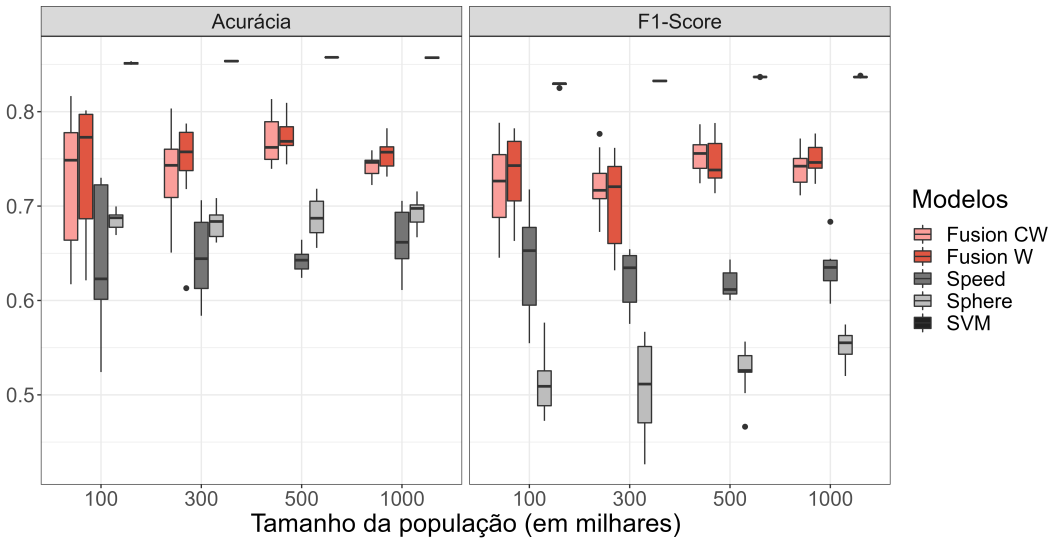
5.1 *CAR SALES RUSSIA*

A primeira base real utilizada, trata-se da base de dados *Car Sales*. Este conjunto de dados apresenta informações sobre vendas de carros em regiões específicas da Rússia, tendo sido

coletados originalmente de um popular site com anúncios de vendas de carros, representando informações sobre vendas de carros na região russa de *Primórski krai*. Originalmente, a base de dados era composta por um total de 17 variáveis e pouco mais de 1.5 milhão de observações, restando após um tratamento prévio, pouco mais de 1.1 milhão de observações e 6 variáveis. A variável resposta utilizada na construção do modelo, *bodytype*, representa o tipo de carroceria que cada veiculo pertencente a base de dados possui, sendo o *bodytype* do veiculo uma das formas mais simples de se categorizar um veiculo.

Assim como para as bases de dados simuladas, o comportamento do modelo foi avaliado para 4 diferentes tamanhos de população, 100 mil, 300 mil, 500 mil e todas as observações existentes na base de após o tratamento.

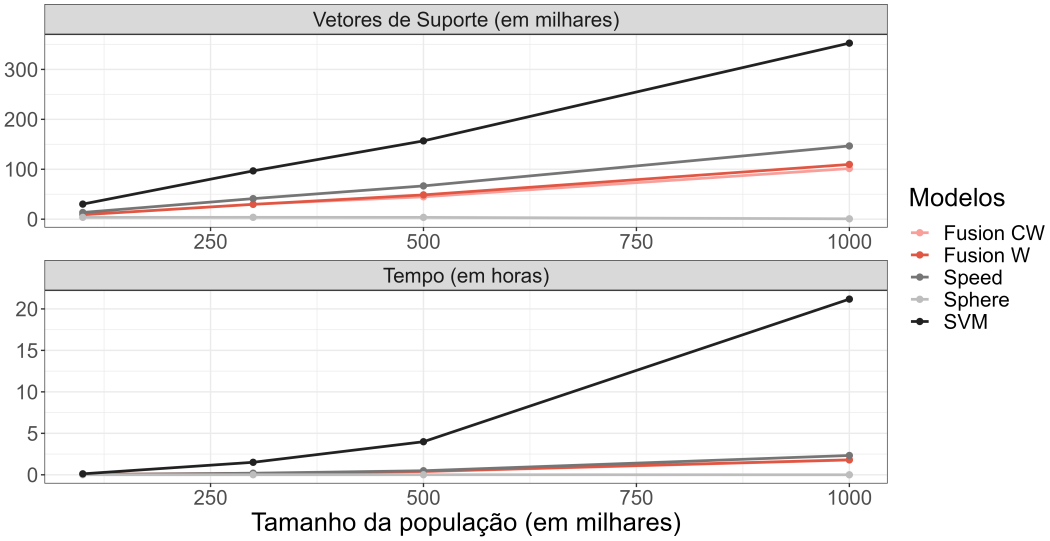
Figura 30 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados utilizando a base de dados *Car Sales*.



Iniciando a analise pela capacidade preditiva dos modelos, a partir da Figura 30, é possível observar um melhor desempenho dos modelos *Fusion W SVM* e *Fusion CW SVM*. Quando observamos o modelo *Sphere SVM*, nota-se que o modelo apresenta grande perda de capacidade preditiva, além de ser desbalanceado, ou seja, alta capacidade preditiva para uma categoria em detrimento das demais.

Do ponto de vista de número de vetores de suporte, observa-se um crescimento linear para todos os modelos, com exceção do modelo *Sphere SVM* que se mantém com um poucos vetores de suporte independente do tamanho da população (Vide Figura 31). Observando o tempo médio necessário para a estimação dos modelos, é visto um comportamento bem próximo para todos os modelos, com baixo crescimento ao aumentar o tamanho da população, e alto ganho em relação ao modelo SVM.

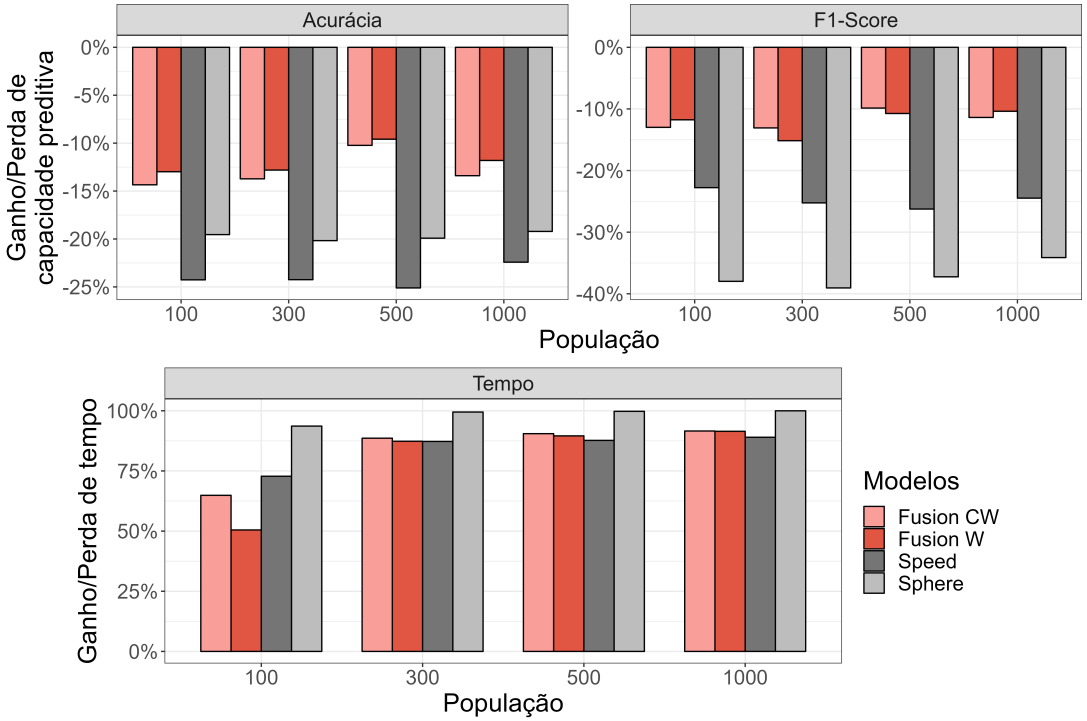
Figura 31 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados utilizando a base de dados *Car Sales*.



Fonte: Autor.

Por fim, com base na Figura 32, analisando de forma direta o ganho/perda de tempo médio e capacidade preditiva, fica evidente a superioridade dos modelos *Fusion*, ao apresentar ganho de tempo médio acima dos 70% e uma menor perca de capacidade preditiva.

Figura 32 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados utilizando a base de dados *Car Sales*.



Fonte: Autor.

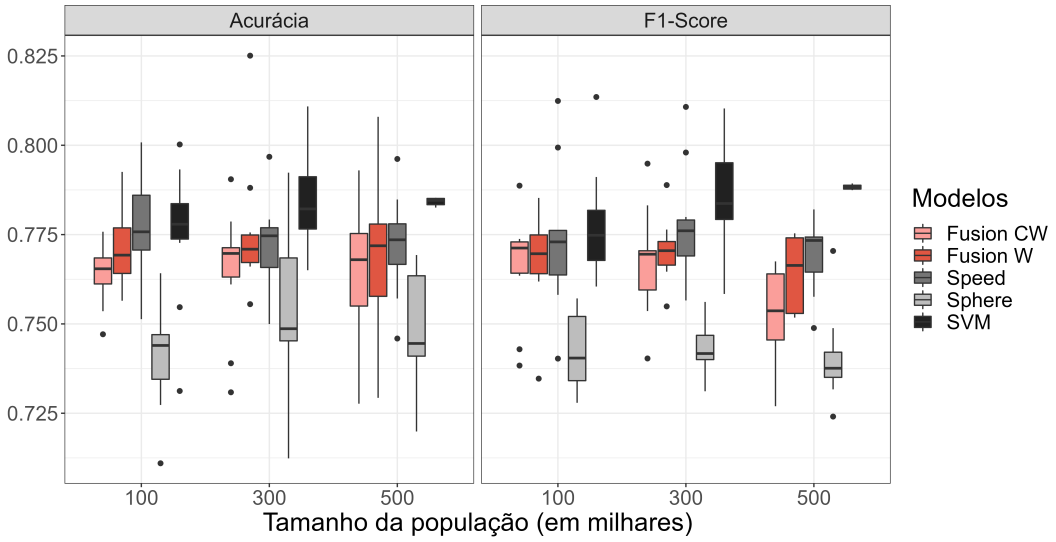
Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 13 no Apêndice B.

5.2 *HEART DISEASE HEALTH*

A segunda base de dados real a ser utilizada, é proveniente originalmente do Sistema de Vigilância de Fatores de Risco Comportamentais (BRFSS - do inglês *Behavioral Risk Factor Surveillance System*), pesquisa realizada por telefone anualmente pelo Centro de Controle e Prevenção de Doenças (CDC - do inglês *Center for Disease Control and Prevention*) dos EUA, com perguntas relacionada a saúde da população. A partir desta pesquisa, realizada desde 1984 com mais de 400 mil pessoas, esta base foi construída para ser usada para a classificação binária de doenças cardíacas, resultando após tratamento em pouco mais de 250 mil observações e 22 variáveis.

Por se tratar de uma base de dados fortemente desbalanceada (90% de entrevistados sem histórico de doenças cardíacas e 10% de entrevistados com histórico de doenças cardíacas), foi necessário a utilização da técnica chamada de *oversampling*, que basicamente aumenta o número de observações da classe minoritária, de forma a balancear a base de dados. Após a utilização do *oversampling*, a base de dados final era composta de 500 mil observações e 22 variáveis. Para se avaliar o modelo para diferentes tamanhos de bases de dados, foram utilizados 3 diferentes tamanhos de população, 100 mil, 300 mil e 500 mil observações.

Figura 33 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados utilizando a base de dados *Heart Disease*.

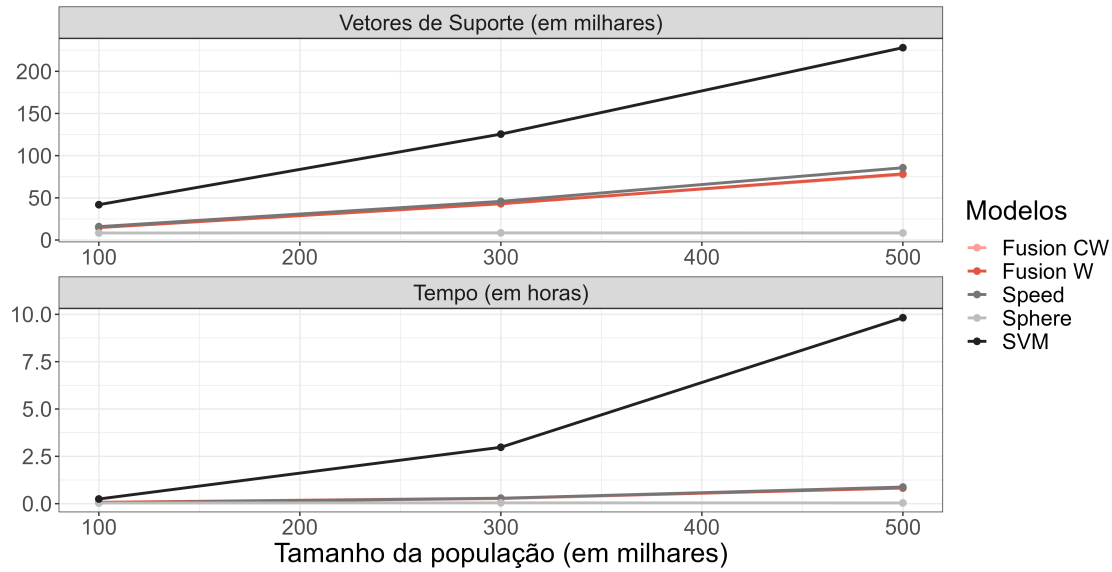


Fonte: Autor.

Diferentemente da base de dados *Car Sales* utilizada na seção anterior, todos os modelos aqui implementados apresentaram comportamentos semelhantes (Vide Figura 33), tendo o *Sphere SVM* o pior desempenho dentre todos.

Quando analisado o número de vetores de suporte dos modelos, é possível notar um comportamento bastante semelhante entre os modelos *Fusion W SVM*, *Fusion CW SVM* e *Speed Up SVM*, tendo todos eles sido executados com menos da metade dos vetores utilizados pelo modelo SVM (Vide Figura 34). O mesmo comportamento é visto quanto ao tempo computacional médio gasto por cada modelo, onde os modelos praticamente se sobrepõem, sendo executados em 10% do tempo do modelo SVM.

Figura 34 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados utilizando a base de dados *Heart Disease*.

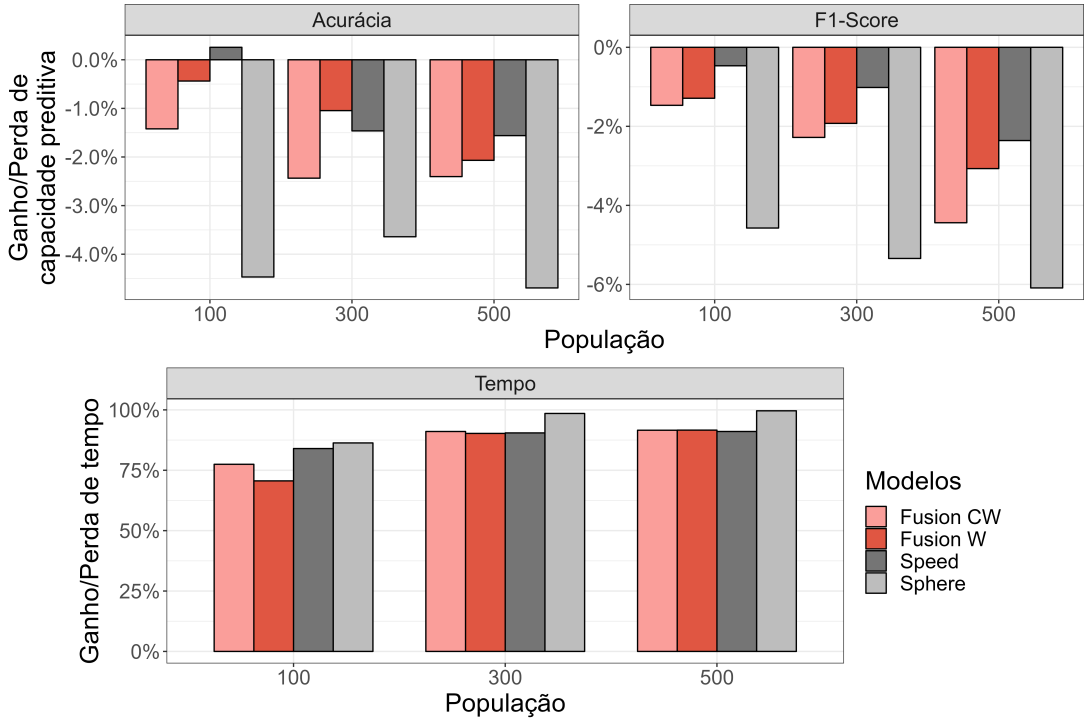


Fonte: Autor.

Por fim, com base na Figura 35 é possível visualizar de forma mais clara o ganho/perda de tempo médio e capacidade preditiva para todos os modelos. Dessa forma, observa-se a diferença mínima existente dentre os modelos *Fusion W SVM*, *Fusion CW SVM* e *Speed Up SVM*, diferença essa que permite a utilização de quaisquer destes modelos com a garantia de alta redução no tempo computacional necessário para a estimação.

Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 14 no Apêndice B.

Figura 35 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados utilizando a base de dados *Heart Disease*.



Fonte: Autor.

5.3 KDDCUP-99 *INTRUSION DETECTION*

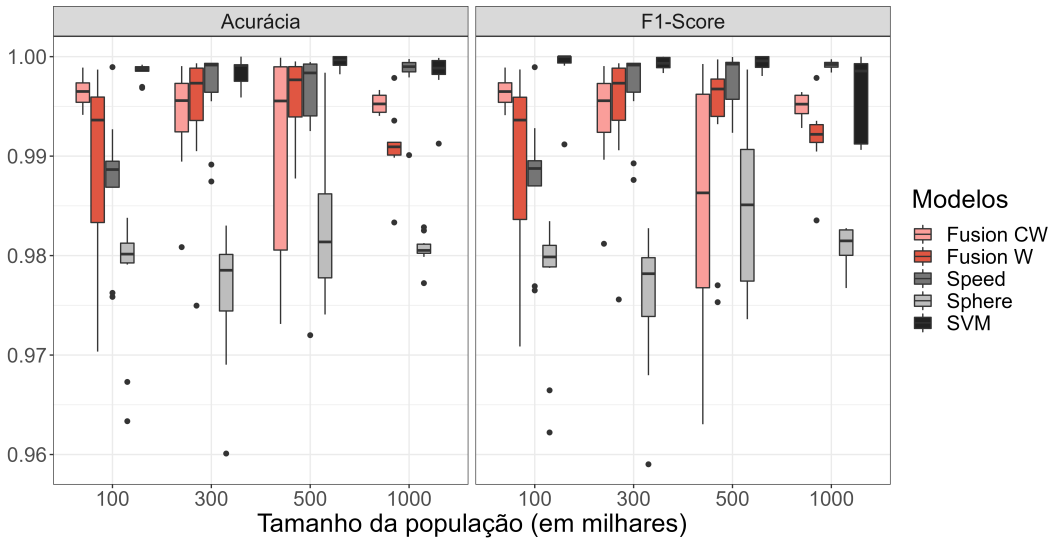
A terceira base de dados a ser utilizada é encontrada originalmente no repositório de dados UCI, e foi utilizada durante a *Third International Knowledge Discovery and Data Mining Tools Competition*, realizada em conjunto com KDD-99. O objetivo da competição era construir um modelo capaz de classificar invasões na rede, ou seja, distinguir entre as conexões ruins (ataques ou intrusões) e conexões boas (normais). Inicialmente, o conjunto de dados era composto por um total 42 variáveis e mais de 4.8 milhões de observações, porém, após a realização de um tratamento na base de dados, com a remoção de variáveis altamente correlacionadas e com alta concentração de observações em uma única categoria, além da utilização da técnica de *undersampling* para reduzir o número de observações da categoria maioritária, dado que a base de dados era desbalanceada, a base de dados ficou com aproximadamente 2 milhões de observações e 17 variáveis.

Para melhor comparação e avaliação do comportamento dos modelos para diferentes tamanhos de populações, foram utilizados 4 diferentes tamanhos de população, 100 mil, 300 mil, 500 mil e 1 milhão de observações. Por motivo de limitação computacional e do tempo

necessário, não foi possível a estimação do modelo SVM para a totalidade de observações geradas.

A base de dados *Intrusion Detection* permite a comparação dos modelos em seu máximo, dado que todos os modelos obtiveram ótimos valores de acurácia e *F1-Score*, como pode ser visto na Figura 36.

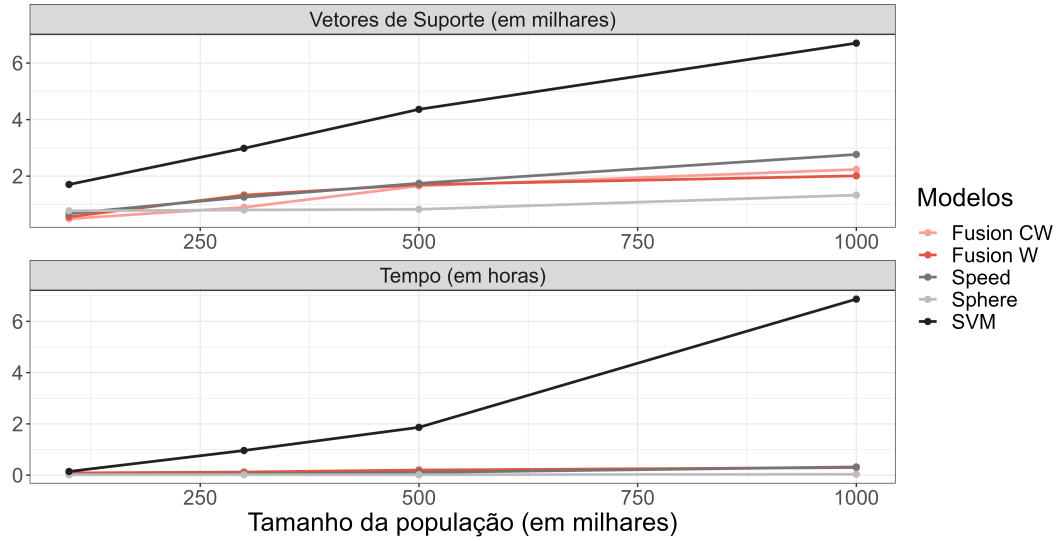
Figura 36 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados utilizando a base de dados KDDCUP-99 *Intrusion Detection*.



Fonte: Autor.

Quando avaliamos o número de vetores de suporte necessários para a estimação do modelo (Vide Figura 37), é possível confirmar que o modelo é facilmente classificado, de forma que mesmo para uma base de dados com 1 milhão de observações, foram necessários apenas aproximadamente 7 mil vetores de suporte para estimar o modelo SVM.

Figura 37 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados utilizando a base de dados KDDCUP-99 *Intrusion Detection*.



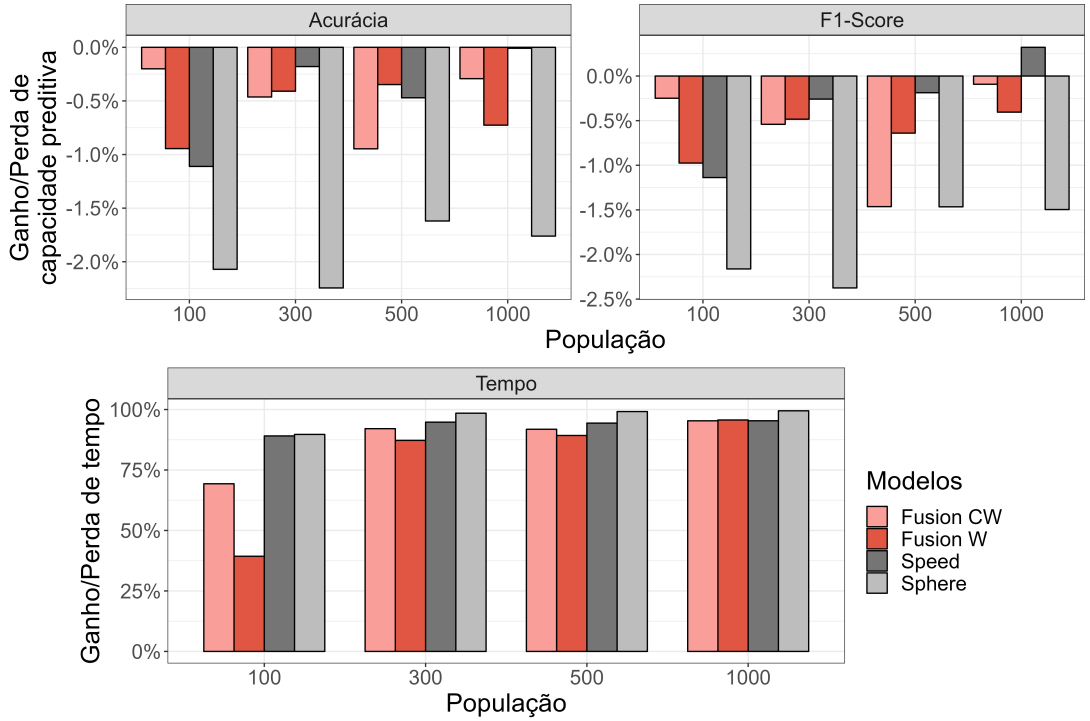
Fonte: Autor.

Esse comportamento permite ainda a execução do modelo em poucas horas, mesmo para o modelo SVM e com grandes bases de dados, O que torna a utilização dos modelos propostos necessária apenas para bases de dados maiores do que as primeiras testadas. Mesmo assim, a utilização dos modelos se mostrou eficaz para todos os tamanhos avaliados, de forma a alcançar uma estimação no melhor dos casos em 5% do tempo médio do modelo SVM.

Para finalizar, ao avaliarmos diretamente o ganho/perda de tempo médio computacional e capacidade preditiva, fica explicíto a qualidade dos modelos propostos para esta base de dados, de forma que o 'pior' modelo apresentou perda de apenas 2%, enquanto que o melhor modelo pode executar o modelo SVM em 5% do tempo.

Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 15 no Apêndice B.

Figura 38 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados utilizando a base de dados KDDCUP-99 *Intrusion Detection*.



Fonte: Autor.

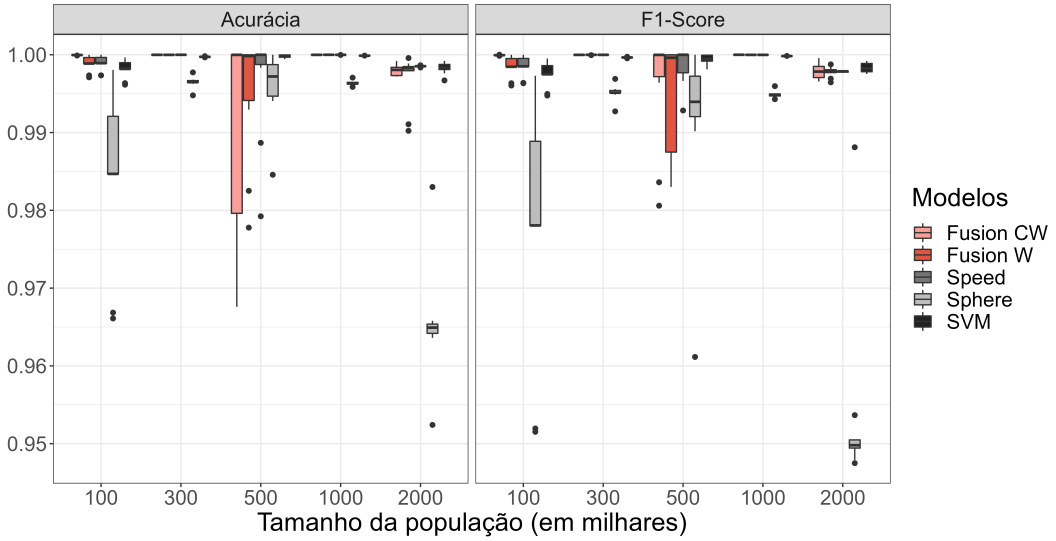
5.4 *TARGETED MARKETING*

Por fim, a nossa última base de dados é fornecido pelo *AI lab of Criteo* e busca avaliar o efeito de campanhas de *marketing* com base em características dos usuários dos sites acompanhados durante o período de testes de 2 semanas. Originalmente, a base de dados possuía 13 milhões de observações e 17 variáveis. Após tratamento, e utilização novamente da técnica *undersampling* para balancear a base de dados, foram mantidas as 17 variáveis, porém, agora com 2 milhões de observações. A variável resposta a ser classificada corresponde ao efeito do tratamento sobre o usuário, onde $Y = 0$ representa que independente da campanha o usuário não acessa o site, e $Y = 1$ representa que o usuário acessou o site.

Assim como nas demais 7 bases de dados, foram utilizados diferentes tamanhos de população para avaliar os modelos existentes, totalizando 5, 100 mil, 300 mil, 500 mil, 1 milhão e 2 milhões de observações.

De maneira mais acentuada ao ocorrido para a base anterior que era facilmente classificada, a base de dados sobre *Targeted Marketing* possui resultados semelhantes quanto a capacidade preditiva dos modelos aplicados, porém ainda melhores, com variação quase nula, como pode

Figura 39 – Boxplots comparativos das métricas de avaliação da capacidade preditiva dos modelos avaliados utilizando a base de dados *Targeted Marketing*.

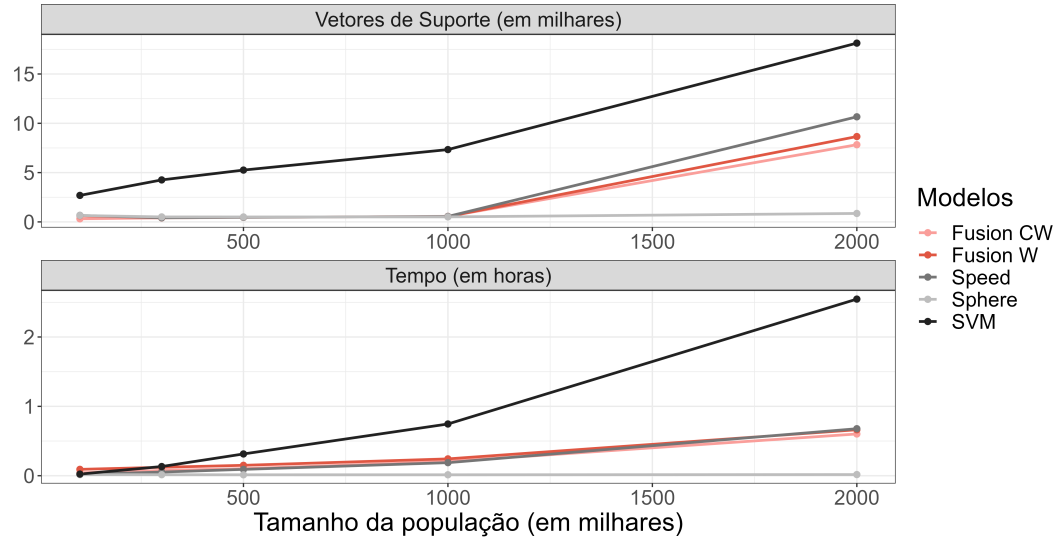


Fonte: Autor.

ser visto na Figura 39.

Quando observamos o número de vetores de suporte, apesar de elevação para o maior tamanho de base de dados, para todos os modelos, inclusive o modelo padrão, foram necessários poucos vetores de suporte, um número irrisório se comparado com a quantidade de dados disponíveis (Vide Figura 40). Quanto ao tempo computacional, apesar de pequeno o tempo gasto pelo modelo SVM, principalmente se comparado com os demais testes, a utilização dos modelos propostos significou ainda reduções maiores no tempo necessário para a sua estimação.

Figura 40 – Análise comparativa do tempo médio de execução e total de vetores de suporte necessários para a estimação dos modelos avaliados utilizando a base de dados *Targeted Marketing*.

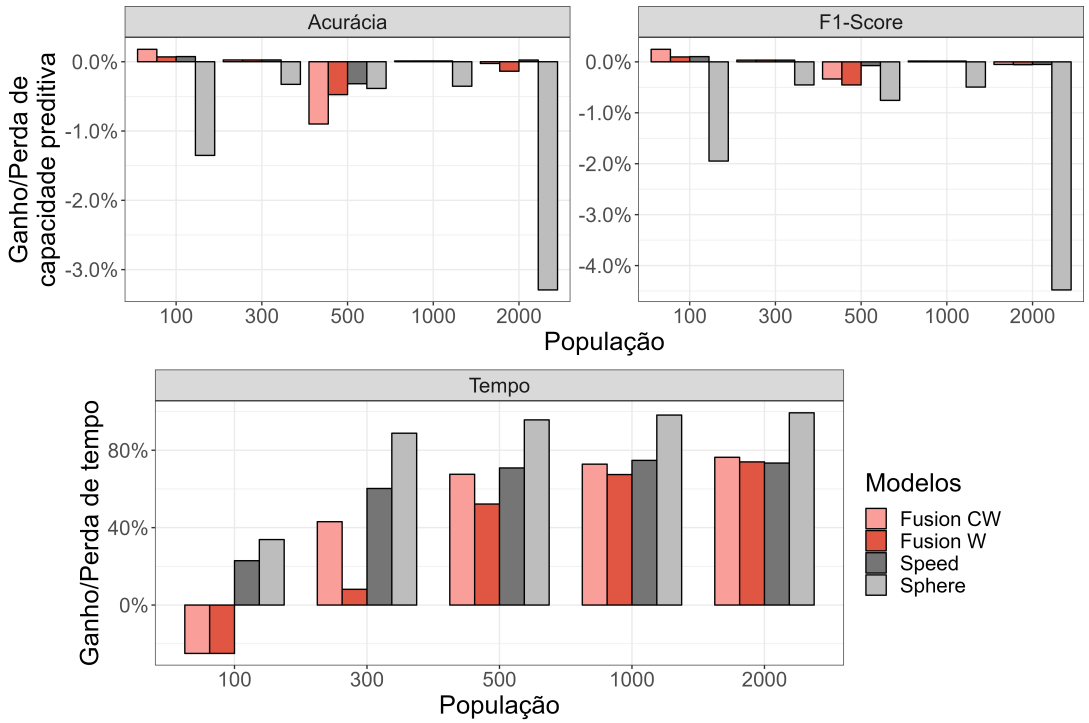


Fonte: Autor.

Por fim, ao avaliar estritamente o ganho/perda de tempo médio computacional e capacidade preditiva, é possível observar de forma mais direta a perda minima de capacidade preditiva para todos os modelos. Da mesma forma, é possível observar ainda o ganho de tempo computacional, principalmente para as maiores bases testadas, desta maneira, apresentando o poder de redução no tempo computacional de todos os modelos avaliados.

Para maiores detalhes dos resultados obtidos para esta base de dados, eles são apresentados na Tabela 16 no Apêndice B.

Figura 41 – Análise comparativa do ganho/perda de tempo médio e capacidade preditiva dos modelos avaliados utilizando a base de dados *Targeted Marketing*.



Fonte: Autor.

6 COMENTÁRIOS FINAIS

Nesta dissertação, foram abordados conceitos sobre a teoria das Máquinas de Vetores de Suporte, e os problemas práticos da utilização de tal modelo para dados massivos. Frente a isso, foram apresentados modelos existentes e conhecidos da literatura para tratar tal problema, como também, foram propostos dois modelos com a finalidade de enfrentar a dificuldade vista. Os modelos utilizados durante todos os experimentos para enfrentar o problema existente para dados massivos, foram todos baseados nas ideias de SVMs fracos (*Weak SVMs*) e SVMs esféricos a partir da utilização de *Coresets*.

Durante os experimentos foram utilizados diversos conjuntos de dados simulados e reais, sempre variando o tamanho das populações, com o objetivo de avaliar o comportamento do modelo para diferentes tamanhos de bases de dados, e em conjuntos com caracteristicas diferentes.

Iniciando as análises a partir das bases de dados simuladas, foi constatado nas duas primeiras bases de dados que os modelos *Speed Up SVM* e *Sphere SVM* apesar de reduzirem o tempo computacional, ambos apresentaram valores mais elevados de perda de capacidade preditiva. Para as outras duas bases de dados simuladas, o comportamento se manteve para o modelo *Sphere SVM*, e apresentou melhora para o modelo *Speed Up SVM*. Já para os modelos propostos, *Fusion W SVM* e *Fusion CW SVM*, ainda foi verificado um comportamento de redução do tempo computacional de estimação dos modelos, estimando em um tempo que representou entre 15% e 40% do tempo gasto pelo modelo SVM, conseguindo além disso, manter a estimação do modelo com uma perda de capacidade preditiva reduzida, perda que superou 5% apenas em 1 das 4 bases de dados avaliadas.

Passando para as bases de dados reais, com exceção da primeira base de dados onde os modelos *Speed Up SVM* e *Sphere SVM* mais uma vez apresentaram maior perda de capacidade preditiva, todos os modelos se comportaram bem diante da avaliação do poder de predição. Avaliando apenas as três bases finais, foi vista de forma geral uma perda de capacidade preditiva de no máximo 5%, permitindo assim uma comparação melhor focando apenas no tempo de estimação do modelo. Do ponto de vista do custo computacional, todos os modelos se mostraram eficazes quando comparado com o modelo SVM, executando suas previsões em até um quarto do tempo total utilizado pelo modelo SVM, mesmo que em algumas bases de dados, essa diferença a favor dos modelos apresentados tenham sido vistas apenas para os

maiores tamanhos de bases de dados (500 mil, 1 milhão, 2 milhões).

Apesar disso, de forma geral, foi observado um comportamento superior dos dois modelos propostos, *Fusion W SVM* e *Fusion CW SVM*, igualando ou superando o modelo *Speed Up SVM* em diversas situações, e com uma tendência de redução cada vez maior do tempo computacional utilizado para a estimação do modelo.

De forma geral, os modelos propostos desempenharam bem seu papel, ao conseguir estimar o modelo completo em alguns momentos utilizando 10% do tempo do modelo SVM, e se mantendo sempre competitivo frente ao *Speed Up SVM* que parte da mesma ideia de ambos os modelos *Fusion W SVM* e *Fusion CW SVM*, reduzindo o tempo e mantendo capacidade preditiva.

Em trabalhos futuros, pretende-se maiores comparações com foco nos *Fusion W SVM* e *Fusion CW SVM*, podendo se realizar o teste de diferentes tamanhos de amostra (ou *coresets* no caso do *Fusion CW SVM*), como também a avaliação de diferentes funções *kernel*, como também a utilização das amostras *weak* em outros modelos de aprendizado de máquina.

REFERÊNCIAS

AL-JARRAH, O. Y.; YOO, P. D.; MUHAIDAT, S.; KARAGIANNIDIS, G. K.; TAHA, K. Efficient machine learning for big data: A review. *Big Data Research*, Elsevier, v. 2, n. 3, p. 87–93, 2015.

ALMEIDA, M. B. D.; BRAGA, A. de P.; BRAGA, J. P. Svm-km: speeding svms learning with a priori cluster selection and k-means. In: IEEE. *Proceedings. Vol. 1. Sixth Brazilian Symposium on Neural Networks*. [S.l.], 2000. p. 162–167.

BACK, Á. J. Aplicação de análise estatística para identificação de tendências climáticas. *Pesquisa agropecuária brasileira*, SciELO Brasil, v. 36, n. 5, p. 717–726, 2001.

BĂDOIU, M.; CLARKSON, K. L. Optimal core-sets for balls. *Computational Geometry*, Elsevier, v. 40, n. 1, p. 14–22, 2008.

BALDI, P.; BRUNAK, S.; CHAUVIN, Y.; ANDERSEN, C. A.; NIELSEN, H. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, Oxford University Press, v. 16, n. 5, p. 412–424, 2000.

BHAVAN, A.; CHAUHAN, P.; SHAH, R. R. et al. Bagged support vector machines for emotion recognition from speech. *Knowledge-Based Systems*, Elsevier, v. 184, p. 104886, 2019.

BREIMAN, L. *Arcing classifiers*. [S.l.], 1996.

CHANG, C.-C.; LIN, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, Acm New York, NY, USA, v. 2, n. 3, p. 1–27, 2011.

CHEN, W.; QIU, X.; CAI, T.; DAI, H.-N.; ZHENG, Z.; ZHANG, Y. Deep reinforcement learning for internet of things: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, IEEE, 2021.

DOKTORSKI, L. L2-svm: Dependence on the regularization parameter. *Pattern Recognition and Image Analysis*, Springer, v. 21, p. 254–257, 2011.

FAN, R.-E.; CHANG, K.-W.; HSIEH, C.-J.; WANG, X.-R.; LIN, C.-J. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, JMLR. org, v. 9, p. 1871–1874, 2008.

GLIELMO, A.; HUSIC, B. E.; RODRIGUEZ, A.; CLEMENTI, C.; NOÉ, F.; LAIO, A. Unsupervised learning methods for molecular simulation data. *Chemical Reviews*, ACS Publications, v. 121, n. 16, p. 9722–9758, 2021.

GURBINĂ, M.; LASCU, M.; LASCU, D. Tumor detection and classification of mri brain image using different wavelet transforms and support vector machines. In: IEEE. *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. [S.l.], 2019. p. 505–508.

HAMEL, L. H. *Knowledge discovery with support vector machines*. [S.l.]: John Wiley & Sons, 2011.

HASAN, B. M. S.; ABDULAZEEZ, A. M. A review of principal component analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining*, v. 2, n. 1, p. 20–30, 2021.

IWENDI, C.; BASHIR, A. K.; PESHKAR, A.; SUJATHA, R.; CHATTERJEE, J. M.; PASUPULETI, S.; MISHRA, R.; PILLAI, S.; JO, O. Covid-19 patient health prediction using boosted random forest algorithm. *Frontiers in public health*, Frontiers, v. 8, p. 357, 2020.

IZBICKI, R.; SANTOS, T. M. dos. *Aprendizado de máquina: uma abordagem estatística*. [S.l.]: Rafael Izbicki, 2020.

KARATZOGLOU, A.; SMOLA, A.; HORNIK, K.; ZEILEIS, A. kernlab-an s4 package for kernel methods in r. *Journal of statistical software*, American Statistical Association, v. 11, n. 9, p. 1–20, 2004.

KHAN, A. I.; SHAH, J. L.; BHAT, M. M. Coronet: A deep neural network for detection and diagnosis of covid-19 from chest x-ray images. *Computer Methods and Programs in Biomedicine*, Elsevier, v. 196, p. 105581, 2020.

LEE, Y.-J.; MANGASARIAN, O. L. Rsvm: Reduced support vector machines. In: SIAM. *Proceedings of the 2001 SIAM International Conference on Data Mining*. [S.l.], 2001. p. 1–17.

LI, P.; LUO, Y.; ZHANG, N.; CAO, Y. Heterospark: A heterogeneous cpu/gpu spark platform for machine learning algorithms. In: IEEE. *2015 IEEE International Conference on Networking, Architecture and Storage (NAS)*. [S.l.], 2015. p. 347–348.

LIU, A. Y.-c. *The effect of oversampling and undersampling on classifying imbalanced text datasets*. Tese (Doutorado) — Citeseer, 2004.

LU, S.; CHEN, Y.; ZHU, X.; OU, Y.; XIE, Y. Exploring support vector machines for big data analyses. *American Journal of Computer Science and Technology*, Science Publishing Group, v. 4, n. 3, p. 90, 2021.

MAIA, M.; PIMENTEL, J. S.; PEREIRA, I. S.; GONDIM, J.; BARRETO, M. E.; ARA, A. Convolutional support vector models: Prediction of coronavirus disease using chest x-rays. *Information*, Multidisciplinary Digital Publishing Institute, v. 11, n. 12, p. 548, 2020.

MANEVITZ, L. M.; YOUSEF, M. One-class svms for document classification. *Journal of machine Learning research*, v. 2, n. Dec, p. 139–154, 2001.

MITCHELL, B.; DEM'YANOV, V. F.; MALOZEMOV, V. Finding the point of a polyhedron closest to the origin. *SIAM Journal on Control*, SIAM, v. 12, n. 1, p. 19–26, 1974.

MUTHUKRISHNAN, S.; KRISHNASWAMY, H.; THANIKODI, S.; SUNDARESAN, D.; VENKATRAMAN, V. Support vector machine for modelling and simulation of heat exchangers. *Thermal Science*, v. 24, n. 1 Part B, p. 499–503, 2020.

NALEPA, J.; KAWULOK, M. Selecting training sets for support vector machines: a review. *Artificial Intelligence Review*, Springer, v. 52, n. 2, p. 857–900, 2019.

PANAHI, M.; SADHASIVAM, N.; POURGHASEMI, H. R.; REZAIE, F.; LEE, S. Spatial prediction of groundwater potential mapping based on convolutional neural network (cnn) and support vector regression (svr). *Journal of Hydrology*, Elsevier, v. 588, p. 125033, 2020.

PIMENTEL, J. S.; OSPINA, R.; ARA, A. Learning time acceleration in support vector regression: A case study in educational data mining. *Stats*, Multidisciplinary Digital Publishing Institute, v. 4, n. 3, p. 682–700, 2021.

REBENTROST, P.; MOHSENI, M.; LLOYD, S. Quantum support vector machine for big data classification. *Physical review letters*, APS, v. 113, n. 13, p. 130503, 2014.

SCHLAG, S.; SCHMITT, M.; SCHULZ, C. Faster support vector machines. *Journal of Experimental Algorithmics (JEA)*, ACM New York, NY, USA, v. 26, n. 1, p. 1–21, 2021.

SCHÖLKOPF, B.; SMOLA, A. J.; WILLIAMSON, R. C.; BARTLETT, P. L. New support vector algorithms. *Neural computation*, MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , v. 12, n. 5, p. 1207–1245, 2000.

SHALEV-SHWARTZ, S.; SINGER, Y.; SREBRO, N. Pegasos: Primal estimated sub-gradient solver for svm. In: *Proceedings of the 24th international conference on Machine learning*. [S.l.: s.n.], 2007. p. 807–814.

SHEYKHMOUSA, M.; MAHDIANPARI, M.; GHANBARI, H.; MOHAMMADIMANESH, F.; GHAMISI, P.; HOMAYOUNI, S. Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, IEEE, v. 13, p. 6308–6325, 2020.

SINGH, K. R.; NEETHU, K.; MADHUREKAA, K.; HARITA, A.; MOHAN, P. Parallel svm model for forest fire prediction. *Soft Computing Letters*, Elsevier, v. 3, p. 100014, 2021.

STRACK, R. Geometric approach to support vector machines learning for large datasets. 2013.

STRACK, R.; KECMAN, V.; STRACK, B.; LI, Q. Sphere support vector machines for large classification tasks. *Neurocomputing*, Elsevier, v. 101, p. 59–67, 2013.

STYAWATI, S.; HENDRASTUTY, N.; ISNAIN, A. R. Analisis sentimen masyarakat terhadap program kartu prakerja pada twitter dengan metode support vector machine. *Jurnal Informatika: Jurnal Pengembangan IT*, v. 6, n. 3, p. 150–155, 2021.

TEAM, R. C. et al. R: A language and environment for statistical computing. Vienna, Austria, 2013.

TSANG, I. W.; KOCSOR, A.; KWOK, J. T. Simpler core vector machines with enclosing balls. In: *Proceedings of the 24th international conference on Machine learning*. [S.l.: s.n.], 2007. p. 911–918.

TSANG, I. W.; KWOK, J. T.; CHEUNG, P.-M.; CRISTIANINI, N. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, v. 6, n. 4, 2005.

VAPNIK, V. N. The nature of statistical learning. *Theory*, springer, 1995.

VAPNIK, V. N. *Statistical Learning Theory*. [S.l.]: Wiley-Interscience, 1998.

WANG, S.; LI, Z.; LIU, C.; ZHANG, X.; ZHANG, H. Training data reduction to speed up svm training. *Applied intelligence*, Springer, v. 41, n. 2, p. 405–420, 2014.

WANG, W.; XU, Z.; LU, W.; ZHANG, X. Determination of the spread parameter in the gaussian kernel for classification and regression. *Neurocomputing*, Elsevier, v. 55, n. 3-4, p. 643–663, 2003.

WU, G.; CHANG, E. Y. Class-boundary alignment for imbalanced dataset learning. In: *ICML 2003 workshop on learning from imbalanced data sets II, Washington, DC*. [S.l.: s.n.], 2003. p. 49–56.

YU, H.; YANG, J.; HAN, J. Classifying large data sets using svms with hierarchical clusters. In: *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2003. p. 306–315.

ZENG, Z.-Q.; YU, H.-B.; XU, H.-R.; XIE, Y.-Q.; GAO, J. Fast training support vector machines using parallel sequential minimal optimization. In: IEEE. *2008 3rd international conference on intelligent system and knowledge engineering*. [S.l.], 2008. v. 1, p. 997–1001.

APÊNDICE A – RESULTADOS GERAIS - DADOS SIMULADOS

Tabela 9 – Comparação dos modelos via simulação em dados independentes

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9309	0.9315	80,000	13,049	2.87
Speed Up SVM	0.6457	0.6609	20,000	9,776	1.65
Sphere SVM	0.8513	0.8577	14,643	13,295	1.89
Fusion W SVM	0.9053	0.9044	20,000	4,289	3.60
Fusion CW SVM	0.9216	0.9277	20,000	6,299	2.72
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9330	0.9312	240,000	38,303	27.40
Speed Up SVM	0.8515	0.8612	60,000	28,509	8.12
Sphere SVM	0.8555	0.8548	14,646	13,250	2.11
Fusion W SVM	0.9212	0.9207	60,000	12,341	7.83
Fusion CW SVM	0.9316	0.9319	60,000	14,054	6.83
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9342	0.9330	450,000	71,082	103.78
Speed Up SVM	0.7561	0.7671	112,500	52,800	20.99
Sphere SVM	0.8308	0.8292	14,648	13,401	1.90
Fusion W SVM	0.9285	0.9246	112,500	27,600	17.35
Fusion CW SVM	0.9297	0.9283	110,000	29,675	17.07
Todas as observações geradas*					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9352	0.9385	900,000	141,420	396.58
Speed Up SVM	0.9032	0.8967	225,000	105,469	88.83
Sphere SVM	0.8438	0.8378	14,630	13,242	2.78
Fusion W SVM	0.9289	0.9237	225,000	40,576	53.19
Fusion CW SVM	0.9319	0.9322	225,000	50,866	52.09

Fonte: Autor.

*Total de observações disponível na Tabela 7.

Tabela 10 – Comparação dos modelos via simulação em dados dependentes

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9353	0.9447	80,000	14,572	4.05
Speed Up SVM	0.7421	0.6620	20,000	3,857	1.23
Sphere SVM	0.7243	0.7110	14,607	13,567	1.96
Fusion W SVM	0.9158	0.9237	20,000	4,013	3.70
Fusion CW SVM	0.9003	0.9081	20,000	3,329	2.43
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9529	0.9554	240,000	36,020	58.47
Speed Up SVM	0.7297	0.6366	60,000	7,955	3.94
Sphere SVM	0.7453	0.7407	9,740	8,965	1.38
Fusion W SVM	0.9060	0.9112	60,000	11,393	8.36
Fusion CW SVM	0.8815	0.8763	60,000	7,846	6.81
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9535	0.9572	450,000	60,471	203.58
Speed Up SVM	0.7415	0.6604	112,500	12.290	10.76
Sphere SVM	0.7548	0.7173	4,867	4.466	0.67
Fusion W SVM	0.9214	0.9123	112,500	7,133	18.02
Fusion CW SVM	0.9250	0.9217	112,500	7,719	15.68
Todas as observações geradas*					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9932	0.9940	900,000	34,647	1,118.52
Speed Up SVM	0.7295	0.6279	225,000	19,260	27.17
Sphere SVM	0.7940	0.7851	9,762	8,995	1.62
Fusion W SVM	0.9303	0.9367	225,000	7,051	53.53
Fusion CW SVM	0.9696	0.9689	225,000	11,800	33.17

Fonte: Autor.

*Total de observações disponível na Tabela 7.

Tabela 11 – Comparação dos modelos via simulação em dados dependentes com base aumentada

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9898	0.9897	80,000	2,853	1.01
Speed Up SVM	0.9864	0.9819	20,000	2,813	1.09
Sphere SVM	0.7508	0.7746	977	937	0.31
Fusion W SVM	0.9860	0.9849	20,000	1,050	7.80
Fusion CW SVM	0.9841	0.9864	20,000	1,062	4.13
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9901	0.9896	240,000	7,374	7.06
Speed Up SVM	0.9877	0.9820	60,000	7,187	4.60
Sphere SVM	0.7650	0.7648	979	933	0.35
Fusion W SVM	0.9859	0.9867	60,000	2,595	11.39
Fusion CW SVM	0.9850	0.9892	60,000	2,310	7.76
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9890	0.9910	450,000	12,932	24.85
Speed Up SVM	0.9848	0.9812	112,500	12,576	10.83
Sphere SVM	0.7787	0.7940	980	932	0.34
Fusion W SVM	0.9829	0.9875	112,500	4,460	17.05
Fusion CW SVM	0.9785	0.9816	112,500	4311	13.24
Todas as observações geradas*					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9907	0.9900	900,000	25,394	88.23
Speed Up SVM	0.9860	0.9851	225,000	24,530	33.21
Sphere SVM	0.7624	0.7639	980	939	0.40
Fusion W SVM	0.9843	0.9863	225,000	7,454	28.86
Fusion CW SVM	0.9867	0.9848	225,000	7,843	26.55

Fonte: Autor.

*Total de observações disponível na Tabela 7.

Tabela 12 – Comparação dos modelos via simulação através da base de dados *Checkerboard*

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9985	0.9989	80,000	3,555	0.65
Speed Up SVM	0.9987	0.9987	20,000	2,891	1.10
Sphere SVM	0.8644	0.8627	9,762	8,805	1.01
Fusion W SVM	0.9144	0.9148	20,000	1,138	3.02
Fusion CW SVM	0.8981	0.9002	20,000	1,118	2.01
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9996	0.9994	240,000	7,595	6.58
Speed Up SVM	0.9994	0.9994	60,000	6,087	3.37
Sphere SVM	0.8401	0.8237	14,644	13,249	1.98
Fusion W SVM	0.9026	0.8994	60,000	2,602	5.51
Fusion CW SVM	0.9421	0.9409	60,000	2,328	4.69
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9999	0.9997	450,000	11,475	18.17
Speed Up SVM	0.9996	0.9996	112,500	9,274	8.08
Sphere SVM	0.8712	0.8562	9,764	8,832	1.19
Fusion W SVM	0.9222	0.9146	112,500	4,443	10.78
Fusion CW SVM	0.9166	0.9096	112,500	4,281	9.92
1 milhão observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9984	0.9990	900,000	18,557	52.15
Speed Up SVM	0.9993	0.9948	225,000	14,928	22.20
Sphere SVM	0.8548	0.8513	9,758	8,838	1.51
Fusion W SVM	0.9085	0.8705	225,000	5,979	19.53
Fusion CW SVM	0.9125	0.8782	225,000	5,982	17.71
2 milhões de observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9999	0.9980	1,800,000	30,140	155.39
Speed Up SVM	0.9999	0.9990	450,000	23,941	52.18
Sphere SVM	0.8540	0.8613	9,755	8,924	1.86
Fusion W SVM	0.9002	0.8902	450,000	10,126	40.74
Fusion CW SVM	0.8974	0.9124	450,000	9,534	43.41

Fonte: Autor.

APÊNDICE B – RESULTADOS GERAIS - DADOS REAIS

Tabela 13 – Comparação dos modelos via dados reais através da base *Car Sales Russia*

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.8513	0.8291	80,000	30,213	7.03
Speed Up SVM	0.6447	0.6403	19,999	13,478	1.91
Sphere SVM	0.6849	0.5142	4,814	3,683	0.45
Fusion W SVM	0.7408	0.7316	19,997	8,865	3.48
Fusion CW SVM	0.7292	0.7214	19,999	8,955	2.47
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.8536	0.8326	270,000	96,760	90.38
Speed Up SVM	0.6467	0.6224	67,487	41,187	11.51
Sphere SVM	0.6814	0.5075	4,833	3,739	0.48
Fusion W SVM	0.7443	0.7064	67,497	29,686	11.45
Fusion CW SVM	0.7365	0.7237	67,494	29,917	10.30
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.8576	0.8368	450,000	156,854	239.29
Speed Up SVM	0.6423	0.6171	112,498	66,665	29.43
Sphere SVM	0.6868	0.5251	4,807	3,644	0.55
Fusion W SVM	0.7753	0.7469	112,489	48,678	24.95
Fusion CW SVM	0.7698	0.7542	112,498	44,814	22.75
Todas observações disponíveis*					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.8572	0.8368	1,043,346	352,655	1,270.95
Speed Up SVM	0.6650	0.6320	260,814	146,712	139.80
Sphere SVM	0.6925	0.5512	958	764	0.14
Fusion W SVM	0.7560	0.7499	260,832	109,692	108.39
Fusion CW SVM	0.7424	0.7415	260,829	101,459	106.80

Fonte: Autor.

*Total de observações disponível na Tabela 8.

Tabela 14 – Comparação dos modelos via dados reais através da base *Heart Disease Health*

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.7751	0.7776	80,000	41,876	14.83
Speed Up SVM	0.7771	0.7740	19,999	15,867	2.37
Sphere SVM	0.7402	0.7335	4,862	2,877	0.49
Fusion W SVM	0.7717	0.7676	20,000	14,754	4.36
Fusion CW SVM	0.7641	0.7662	19,999	14,710	3.34
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.7842	0.7856	240,000	125,524	178.74
Speed Up SVM	0.7728	0.7776	59,996	45,830	17.04
Sphere SVM	0.7557	0.7436	14,607	8,483	2.65
Fusion W SVM	0.7760	0.7704	60,000	42,929	17.42
Fusion CW SVM	0.7651	0.7677	59,998	43,090	15.99
Todas observações disponíveis*					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.7842	0.7883	450,000	228,012	589.30
Speed Up SVM	0.7720	0.7697	112,497	85,674	52.70
Sphere SVM	0.7474	0.7403	14,534	8,342	2.26
Fusion W SVM	0.7680	0.7641	112,493	78,374	49.38
Fusion CW SVM	0.7654	0.7533	112,496	77,511	49.64

Fonte: Autor.

*Total de observações disponível na Tabela 8.

Tabela 15 – Comparação dos modelos via dados reais através da base KDDCUP-99 *Intrusion Detection*

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9985	0.9989	80,000	1,704	8.58
Speed Up SVM	0.9874	0.9876	20,000	676	0.94
Sphere SVM	0.9778	0.9773	4,900	772	0.88
Fusion W SVM	0.9890	0.9892	18,216	559	5.21
Fusion CW SVM	0.9965	0.9965	18,605	490	2.63
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9985	0.9993	240,000	2,985	57.69
Speed Up SVM	0.9967	0.9967	59,998	1,256	3.01
Sphere SVM	0.9761	0.9756	4,900	801	0.88
Fusion W SVM	0.9944	0.9945	54,961	1,329	7.36
Fusion CW SVM	0.9939	0.9939	52,125	895	4.57
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9994	0.9994	450,000	4,363	111.87
Speed Up SVM	0.9947	0.9975	112,500	1,746	6.29
Sphere SVM	0.9832	0.9847	4,903	825	0.94
Fusion W SVM	0.9960	0.9930	105,287	1,697	12.01
Fusion CW SVM	0.9900	0.9848	101,503	1,654	9.13
1 milhão de observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9982	0.9959	900,000	6,706	412.09
Speed Up SVM	0.9981	0.9991	225,000	2,767	19.15
Sphere SVM	0.9759	0.9755	14,695	1,873	3.25
Fusion W SVM	0.9910	0.9919	191,980	2,010	17.80
Fusion CW SVM	0.9953	0.9950	217,721	2,236	19.21

Fonte: Autor.

Tabela 16 – Comparação dos modelos via dados reais através da base *Targeted Marketing*

100 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9982	0.9975	80,000	2,687	1.41
Speed Up SVM	0.9989	0.9985	20,000	664	1.09
Sphere SVM	0.9847	0.9781	4,901	650	0.93
Fusion W SVM	0.9989	0.9985	20,000	631	5.50
Fusion CW SVM	0.9999	0.9999	20,000	314	2.53
300 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9997	0.9996	240,000	4,258	7.95
Speed Up SVM	0.9999	0.9999	60,000	419	3.16
Sphere SVM	0.9965	0.9951	4,902	506	0.89
Fusion W SVM	0.9999	0.9999	60,000	411	7.30
Fusion CW SVM	0.9999	0.9999	60,000	411	4.52
500 mil observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9998	0.9994	450,000	5,255	18.83
Speed Up SVM	0.9966	0.9986	110,000	461	5.49
Sphere SVM	0.9959	0.9918	4,899	501	0.81
Fusion W SVM	0.9951	0.9948	110,000	469	9.00
Fusion CW SVM	0.9908	0.9960	102,000	455	6.10
1 milhão de observações					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9999	0.9998	900,000	7,334	44.74
Speed Up SVM	0.9999	0.9999	225,000	553	11.29
Sphere SVM	0.9963	0.9949	4,898	510	0.82
Fusion W SVM	0.9999	0.9999	225,000	547	14.57
Fusion CW SVM	0.9999	0.9999	225,000	564	12.16
Todas observações disponíveis*					
Modelo	Acurácia	F1-Score	Amostra de Treino	Vetores de Suporte	Tempo (minutos)
SVM	0.9983	0.9984	1,800,000	18,130	152.77
Speed Up SVM	0.9985	0.9978	450,000	10,658	40.66
Sphere SVM	0.9628	0.9578	4,902	885	0.94
Fusion W SVM	0.9969	0.9978	450,000	8,656	39.75
Fusion CW SVM	0.9980	0.9978	450,000	7,828	36.11

Fonte: Autor.

*Total de observações disponível na Tabela 8.