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

DENISSON AUGUSTO BASTOS LEAL

Ensembles **Dinâmicos para Detecção de** ***Concept Drift*** **em Séries Temporais**

Recife

2022

DENISSON AUGUSTO BASTOS LEAL

***Ensembles* Dinâmicos para Detecção de *Concept Drift* em Séries Temporais**

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Inteligência Computacional

Orientador (a): Adriano Lorena Inacio de Oliveira

Coorientador (a): Ricardo de Andrade Araújo

Recife

2022

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

L435e Leal, Denisson Augusto Bastos
Ensembles dinâmicos para detecção de *concept drift* em séries temporais / Denisson Augusto Bastos Leal. – 2022.
68 f.: il., fig., tab.

Orientador: Adriano Lorena Inácio de Oliveira.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.

Inclui referências e anexo.

1. Inteligência computacional. 2. Séries temporais. I. Oliveira, Adriano Lorena Inácio de (orientador). II. Título.

006.31 CDD (23. ed.) UFPE - CCEN 2022-134

Denisson Augusto Bastos Leal

“Ensembles Dinâmicos para Detecção de Concept Drift em Séries Temporais”

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

Aprovado em: 03/03/2022.

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto
Centro de Informática/UFPE

_____

Prof. Dr. Jarley Palmeira Nóbrega
Centro de Tecnologias Estratégicas do Nordeste/MCTI

_____

Prof.Dr. . Adriano Lorena Inácio de Oliveira
Centro de Informática/UFPE
(**Orientador**)

AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer à minha família, em especial minha mãe Arnalva Bastos, pelo apoio dado durante todos esses anos.

Gostaria de agradecer também ao meu orientador Dr. Adriano Lorena, meu coorientador Dr. Ricardo Araújo e a Gustavo Oliveira que me ajudaram em toda a jornada deste trabalho. Aos meus colegas de curso Raphael Crespo, Dr. Kellyton Brito e Elisa Andrade pelo apoio nos momentos difíceis.

À Universidade Federal de Pernambuco (UFPE), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e ao Programa de Pós-Graduação do Centro de Informática (CIn), que tornaram essa pesquisa possível.

RESUMO

Séries temporais são medições realizadas em um intervalo fixo de tempo, que podem ser usadas em diversos tipos de problemas. Elas são estudadas por diversas áreas a fim de compreender as características de sua geração. A área mais específica de previsão de séries temporais busca encontrar padrões dos movimentos em situações que já ocorreram para prever a próxima observação da série. Porém, ao longo do tempo podem acontecer alguns eventos que mudam todo o contexto da série, o *concept drift*, e o conhecimento armazenado da série pode não refletir mais a distribuição da série após esse evento. Então, quando um modelo de aprendizado de máquina é treinado para realizar previsões e um *concept drift* acontece, esse modelo passa a ficar defasado e para atualizar o modelo, alguns pontos do novo conceito precisam ser capturados e armazenados, até serem suficientes para um novo treinamento. Durante esse período de coleta de dados os modelos aumentam o erro bruscamente, afetando o desempenho geral do sistema de previsão e dependendo da frequência em que o *concept drift* aconteça, pode inviabilizar o seu uso. O objetivo desse trabalho é propor um método para previsão de séries temporais na presença de *concept drift*, minimizando o impacto da redução do desempenho preditivo durante o processo de adaptação ao novo conceito. Para isso foram propostos três métodos que usam o conceito antigo para melhorar o desempenho nessa fase de adaptação. Os métodos usam *Particle Swarm Optimization* (PSO) para otimização do treinamento de partículas *Extreme Learning Machine* (ELM), que são usadas na previsão e como sensores para detecção *concept drift*. O primeiro método usa um *ensemble* com todas as partículas treinadas. O segundo faz uma combinação, usando o algoritmo guloso, das melhores partículas quando o *concept drift* é detectado até a sua adaptação. E o terceiro usa a melhor combinação das partículas desde o início e atualiza a combinação depois da detecção de um *concept drift*. Todos os métodos propostos fazem adaptação para o novo conceito depois de ter dados suficientes para o treinamento. Nos experimentos foram usadas sete séries, sendo elas quatro geradas sinteticamente com *concept drift* conhecidos e três séries reais de índices do mercado financeiro com *concept drift* desconhecidos. Os resultados obtidos foram comparados com métodos da literatura e dois métodos propostos conseguiram resultados melhores com significância estatística. Mostrando que, o período de adaptação do método ao novo conceito é relevante no erro geral da previsão e que o treinamento anterior pode ajudar a reduzir esse erro.

Palavras-chaves: séries temporais; concept drift; aprendizado de máquina; ensemble.

ABSTRACT

Time series are measurements performed over a fixed time interval, which can be used in different types of problems. They are studied by many disciplines seeking to understand the characteristics of their generation. There is a more specific area of time series forecasting that seeks to find patterns of movement in situations that happened in the past to predict the next observation in the series. However, over time, events known as concept drift can occur that change the context of the series and the knowledge stored in the series is no longer the distribution of the series after this event. So, when a machine learning model is trained to perform predictions and concept drift occurs, this model becomes outdated and for the model update, some points of the new concept need to be captured and stored, until they are sufficient for a new training. During this period of data collection, the models errors increase exponentially, affecting the overall performance of the forecasting system and depending on the frequency in which the concept drift happens, which can make its use unfeasible. The objective of this work is to propose a time series forecasting method in the presence of concept deviation, minimizing its impact during the adaptation process to the new concept. For this, three methods were proposed that use the old concept to improve performance in this adaptation phase. The methods use Particle Swarm Optimization (PSO) to optimize the training of Extreme Learning Machine (ELM) models, which are used in prediction and as sensors for concept drift detection. The first method uses an ensemble with all models trained. The second uses a combination, using the greedy algorithm, of the best models when the concept drift is detected until its adaptation. And the third uses the best combination of models from the start and updates the combination after a concept drift is detected. All proposed methods adapt to the new concept after having enough data for training. In the experiments, seven series were used, four synthetically generated with known concept drift and three real series of financial market indices with unknown concept drift. The results obtained were compared with methods in the literature and two proposed methods achieved better results with statistical significance. Showing that the period of adaptation of the method to the new concept is relevant in the general error of the forecast and that previous training can help to reduce this error.

Keywords: time series; concept drift; machine learning; ensemble.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AR	Autoregressivo
BOLE	Boosting-like Online Learning Ensemble
CD	Critical Difference
DDD	Diversity for Dealing with Drifts
DDM	Drift Detector Mechanism
DJIA	Dow Jones Industrial Average
ECDD	Exponentially Weighted Moving Average para Concept Drift Detection
EMAD	Exponential Moving Average Detector
ELM	Extreme Learning Machine
FL	Federated Learning
GRU	Gated Recurrent Units
ICM	Inductive Conformal Martingales
KOS-ELM	Kernel Online Sequential ELM
LFR	Linear Four Rates
LSTM	Long Short-Term Memory
LFR	Linear Four Rates
MSE	Mean Square Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
OABM1	Online AdaBoost-based M1
OR-ELM	Online Recurrent ELM
OzaBag	Oza and Russell's Online Bagging
PSO	Particle Swarm Optimization
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SMOTE	Synthetic Minority Oversampling TEchnique
ZSD	Z-Score Detector

SUMÁRIO

1 INTRODUÇÃO

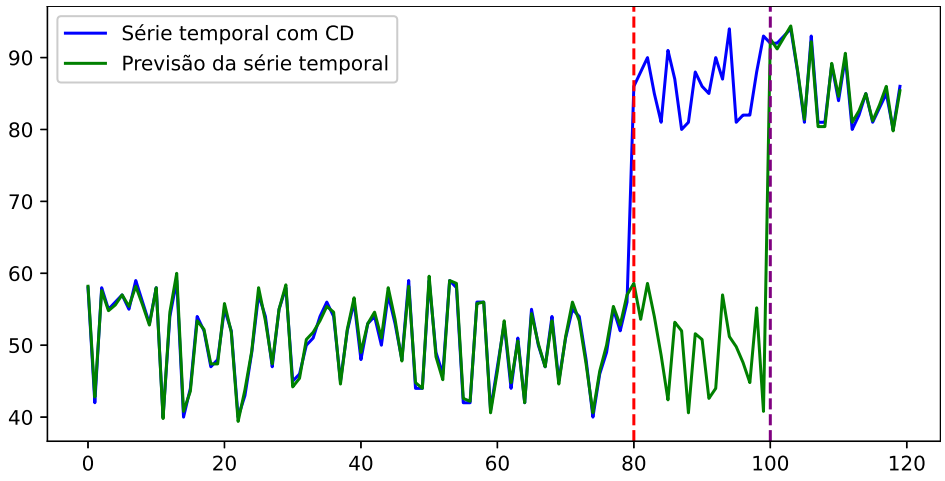
Séries temporais são, por definição, medições realizadas em um intervalo fixo de tempo, que podem ser usadas para modelar diversos tipos de problemas (COWPERTWAIT; ANDREW, 2009). Exemplos disso estão em toda parte, como crescimento populacional, número de voos diários para um determinado destino, área da floresta amazônica ao longo dos anos, quantidade diária de novos casos de COVID-19, entre outros.

As séries temporais são estudadas por diversas áreas a fim de compreender as características de seu fenômeno gerador, possibilitando estimá-lo no tempo futuro (KIRCHGÄSSNER; WOLTERS; HASSLER, 2012). Durante os anos de 2020 e 2021 foi muito comum ver em jornais e na televisão sobre as tendências de aumento ou diminuição dos casos de COVID-19, de forma a dar suporte a tomada de decisão por chefes de estado para adoção de medidas de quarentena e distanciamento social, visando retardar a disseminação acelerada da mesma e para garantir o nivelamento da demanda por leitos nos hospitais (HE et al., 2020). Nesse contexto, existe a área de previsão de séries temporais que é mais específica e não captura somente a tendência, mas tenta encontrar um padrão desse movimento em situações que já ocorreram para prever a próxima observação da série, sendo esta geralmente conhecida como previsão de um passo à frente. Pois, quanto mais passos na frente são previstos, maior o erro acumulado e a incerteza (LIM; ZOHREN, 2021).

Porém ao longo do tempo podem acontecer alguns eventos que mudam todo o contexto da série, esses eventos são chamados de *concept drift* e trás consigo uma mudança da distribuição da mesma, então a probabilidade de uma determinada saída com uma entrada especifica é diferente antes e depois do acontecimento desse evento (IWASHITA; PAPA, 2018). Usualmente o conhecimento armazenado na série não reflete a realidade pós *concept drift*. Logo, a detecção desses eventos torna-se fundamental para gerar uma previsão mais confiável, com uma menor taxa de erro (CASTELLANI; SCHMITT; HAMMER, 2021). Depois da detecção é necessário que o modelo se adapte à nova realidade e, para que isso aconteça, é necessário que dados desse novo conceito sejam coletados e apresentados novamente ao modelo. Vale mencionar que a quantidade de dados necessários para uma correta adaptação ao evento depende da complexidade do modelo empregado (SATO et al., 2021). Logo, é possível observar que sem a inclusão do processo de adaptação, o erro de previsão aumenta, implicando em uma diminuição do desempenho global do modelo (HIDALGO; SANTOS; BARROS, 2021).

A Figura 1 apresenta uma série com *concept drift* e a previsão de um modelo para essa série, onde as linhas tracejadas vermelhas e roxas representam o momento que acontece o *concept drift* e o momento da adaptação, respectivamente. A Figura 1 também mostra que durante o período pré *concept drift*, antes da linha vermelha, e pós adaptação, depois da linha roxa, a previsão possui um erro mínimo em relação a série real, porém no período entre as duas linhas tracejadas o erro na previsão cresce bruscamente, caracterizado pelo baixo desempenho do modelo de previsão, levando muitas vezes a ser o principal fator que leve o modelo de previsão a um ruim desempenho geral. Por esse motivo é importante estudar técnicas de detecção e adaptação do *concept drift*.

Figura 1 – Ilustração da adaptação para o *concept drift*.



Fonte: O autor

O surgimento desses eventos são difíceis de prever, porém existem algumas formas de identificar o acontecimento pós-fato baseado no comportamento da própria série (HIDALGO; SANTOS; BARROS, 2021). Alguns desses eventos não são claramente explicáveis. No entanto, outros eventos tais como a crise imobiliária de 2008, o Impeachment da ex-presidente Dilma Rousseff e a pandemia da COVID-19 tiveram impactos visuais na série temporal, que possibilitam a sua identificação de forma clara, como a mudança do comportamento de compra de consumidores, dos valores dos bens negociados no mercado de ações, no câmbio de moedas, entre outros, durante esse período. Em ambos os casos, esses períodos de adaptação tornam-se um problema para métodos que realizam previsões, tornando eles, muitas vezes, inoperantes durante esse período (TIAN, 2020).

1.1 OBJETIVOS E PERGUNTAS DE PESQUISA

O objetivo geral deste trabalho é propor um método para previsão de séries temporais na presença de *concept drift*, que minimize o impacto da redução do desempenho preditivo durante o processo de adaptação ao novo conceito. Para atingir esse objetivo os seguintes objetivos específicos devem ser alcançados:

1. Desenvolvimento de um método capaz de prever séries temporais;
2. Desenvolvimento de um detector capaz de detectar *concept drift*;
3. Identificação de quais eventos do passado podem ser utilizados para melhorar a previsão da série temporal após a ocorrência do *concept drift*;
4. Verificação do desempenho preditivo de modelos de *ensemble* de partículas em ambientes com presença de *concept drift*;
5. Desenvolvimento de um *ensemble* otimizado para solucionar o problema de previsão de séries temporais com *concept drift*;
6. Realização de uma análise experimental com o modelo proposto utilizando medidas de desempenho consolidadas na literatura e testes estatísticos para confirmação dos resultados obtidos superiores aos trabalhos recentemente propostos na literatura.

Esses objetivos serviram para orientar o andamento do trabalho e durante a leitura servirá para facilitar o entendimento. Além disso, esse trabalho responde às seguintes questões de pesquisa:

Questão 1: Combinar os melhores modelos do PSO oferece vantagens ao uso apenas do melhor? Sabendo que as partículas de um PSO tendem a convergir para uma mesma solução, combinar modelos de um PSO vai realmente reduzir a quantidade de variações nas previsões e com isso diminuir o erro?

Questão 2: Tendo em vista que ao acontecer o *concept drift* teoricamente os modelos ficam obsoletos, é possível usar modelos treinados em conceitos passados para acelerar na adaptação do novo conceito?

1.2 ESTRUTURA DO DOCUMENTO

A organização desse trabalho foi dividida em seis capítulos: Introdução, Fundamentação Teórica, Trabalhos Relacionados, Método Proposto, Experimentos e Conclusão. A Introdução, no Capítulo 1, apresenta a contextualização, a motivação, os objetivos, as questões de pesquisa e a estrutura dos capítulos subsequentes deste trabalho.

Na Fundamentação Teórica, no Capítulo 2, serão apresentados os principais conceitos teóricos necessário para o entendimento do método proposto. Bem como define séries temporais, *concept drift*, aprendizagem de máquina, *ensembles*, processos de otimização e testes estatísticos.

Os Trabalhos Relacionados, no Capítulo 3, apresenta os principais e mais recentes trabalhos apresentados na literatura para solucionar o problema de previsão de séries temporais com *concept drift*. Serão discutidos os resultados, vantagens e desvantagens de cada um dos trabalhos investigados.

O Método Proposto, no Capítulo 4, trata do detalhamento do que é proposto neste trabalho para resolução do problema. O método foi feito visando atingir os objetivos e responder às questões de pesquisa. Ele foi construído de forma incremental, então será passado por cada uma das etapas e explicitando suas diferenças.

Nos Experimentos, Capítulo 5, serão apresentados a configuração experimental, as séries utilizadas e como foram obtidas, os resultados dos experimentos, os testes estatísticos e uma discussão sobre esses resultados relacionando com as questões de pesquisa. Também, será realizada uma análise comparativa dos resultados obtidos com o modelo proposto e os modelos recentemente apresentados na literatura.

Por fim, a Conclusão, no Capítulo 6, são apresentadas as considerações finais deste trabalho. Bem como é feita uma discussão sobre o modelo desenvolvido, suas limitações e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo trata dos principais fundamentos para entender o método proposto construído. Inicialmente fala sobre o que é uma série temporal e qual sua composição, seguindo pelo problema abordado, que é o *concept drift*, então é explicado o que é, quais os tipos e como pode ser detectado. Posteriormente, aborda as ferramentas utilizadas, como o modelo de aprendizado de máquina ELM, seguindo por uma forma de otimização do treinamento, o PSO, formas de fazer combinações ou *ensemble*, um algoritmo guloso de otimização em buscas e finaliza pela avaliação dos resultados através de medidas de desempenho e testes estatísticos.

2.1 SÉRIES TEMPORAIS

De acordo com Box, Jenkins e Reinsel (1994), uma série temporal representa uma sequência de observações sobre um determinado fenômeno ou evento que evolui com o tempo. Tais observações são pontos temporais discretos ou contínuos, sendo estes usualmente equidistantes. Portanto, uma série temporal pode ser formalmente definida por (BOX; JENKINS; REINSEL, 1994):

$$\mathbf{X} = \{x_t \in \mathbb{R} \mid t = 1, 2, 3 \ldots N\}, \tag{1}$$

onde o termo t representa um índice temporal (tempo), que determina a granularidade das observações, x_t é o valor medido ou observado e o termo N representa o número de observações ou pontos da série temporal.

Durante a coleta de dados de uma série temporal, a forma que o valor é coletado pode variar de acordo com o que fizer mais sentido para o problema. Esses valores podem ser amostrados onde, por exemplo, pode ser visto na bolsa de valores onde o dado coletado é normalmente o preço de fechamento dos ativos, seja a frequência diária, a cada uma hora ou a cada cinco minutos. O outro caso comum é os dados serem agregados, onde, usando o mesmo exemplo da bolsa de valores, o volume das negociações, que é a soma de todo o dinheiro usado em negociações de um determinado ativo. Dentre outras formas, é possível também usar a média dos valores, como por exemplo, a série usada para acompanhar a temperatura do planeta com frequência anual (COWPERTWAIT; ANDREW, 2009).

Uma série temporal, seja de qualquer natureza, é composta em termos de abordagem

aditiva ou multiplicativa, por três componentes ou características principais, e determinada, respectivamente por (MORETTIN; TOLOI, 2006):

$$x_t = T_t + C_t + R_t, \tag{2}$$

e

$$x_t = T_t \cdot C_t + R_t, \tag{3}$$

onde T_t é uma componente de tendência (incremento ou decremento gradual sistemático das observações do fenômeno temporal), C_t é uma componente cíclica (movimentações cíclicas na observação do fenômeno temporal sem uma definição clara de sua frequência) ou sazonal (movimentações cíclicas na observação do fenômeno temporal com flutuações de caráter sazonal, isto é, com frequência bem definida), e R_t é a componente aleatória ou ruído. Desta forma, a análise de uma série temporal consiste na identificação e estimativa do grau de contribuição de cada uma destas componentes (características) para que se tenha uma aproximação do fenômeno gerador da série temporal (MORETTIN; TOLOI, 2006).

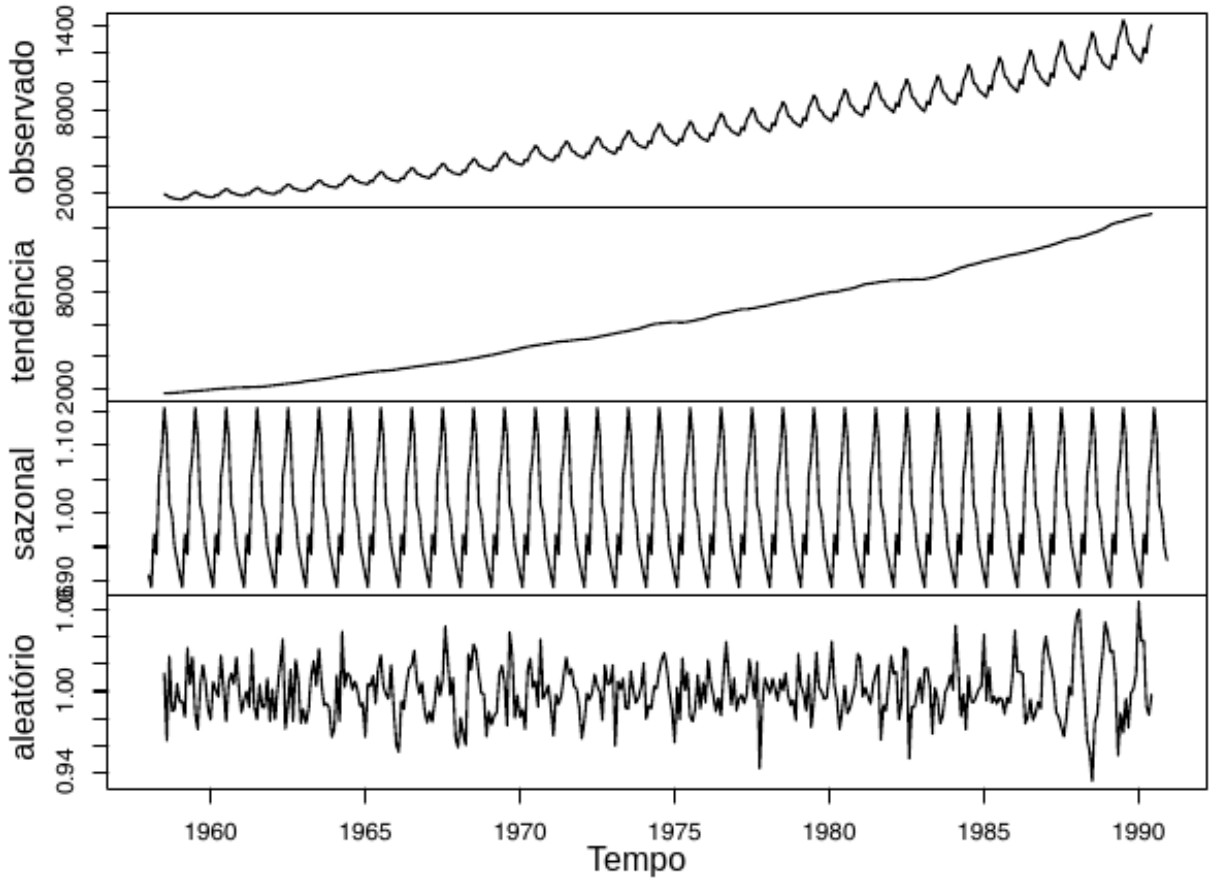
A Figura 2 mostra um exemplo da decomposição de uma série, mostrando na primeira parte do gráfico a série original de produção de eletricidade, seguida por sua tendência, depois pela sazonalidade e por fim a parte aleatória da série. Com a decomposição é possível identificar quais componentes fazem parte da série em questão, porém, além disso, é importante saber como essas partes se relacionam, já que para séries mais simples esses modelos de combinação das séries podem ser usados até mesmo para realizar previsões (COWPERTWAIT; ANDREW, 2009).

2.2 CONCEPT DRIFT

O *concept drift* é caracterizado como uma mudança no fenômeno gerador da série. Logo, em um contexto onde um modelo aprendeu o comportamento da série, depois do surgimento de um *concept drift*, o modelo torna-se defasado (IWASHITA; PAPA, 2018). Matematicamente, o *concept drift* pode ser definido como, dado o par entrada e saída de uma série:

$$\{(x_0, y_0), ..., (x_i, y_i), ..., (x_t, y_t)\} \tag{4}$$

Figura 2 – Decomposição de série temporal da produção de eletricidade.



Fonte: Adaptado de Cowpertwait e Andrew (2009)

onde, x_t é a entrada no tempo t e y_t é a saída correspondente no tempo t. No contexto atual os dados seguem uma distribuição $p_t(x, y)$, supondo que ocorra um *concept drift* no tempo $t + 1$, a probabilidade $p_{t+1}(x, y)$ muda (LEE; PARK, 2022). Logo:

$$p_t(x, y) \neq p_{t+1}(x, y) \tag{5}$$

O *concept drift* pode ser categorizado em pelo menos três grupos. O *Class prior concept drift* em que refere-se a mudanças na probabilidade a priori (LEE; PARK, 2022). onde:

$$p_t(y) \neq p_{t+1}(y) \tag{6}$$

O *Virtual concept drift* em que a mudança é na probabilidade condicional (LEE; PARK, 2022):

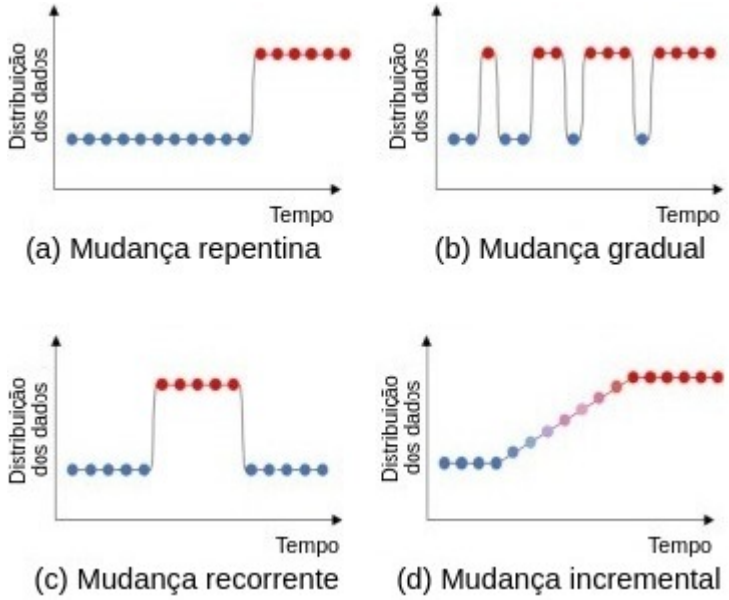
$$p_t(x|y) \neq p_{t+1}(x|y) \tag{7}$$

E o *Real concept drift* em que a mudança é na probabilidade a posteriori (LEE; PARK, 2022):

$$p_t(y|x) \neq p_{t+1}(y|x) \tag{8}$$

Quando ocorre um *real concept drift* podemos dizer que o conceito de destino para uma entrada fixa mudou com o tempo (LEE; PARK, 2022). O *real concept drift* pode ocorrer de diversas formas, as mais comuns estão ilustradas na Figura 3, onde, iniciando pela mudança repentina ela ocorre quando o novo conceito substitui abruptamente o antigo. A mudança gradual é caracterizada pela mudança ocorrer durante um tempo que pode ser considerado longo e os conceitos antigos e novos coexistem durante esse tempo. A mudança recorrente ocorre quando o conceito anterior volta depois de um tempo através de um novo *concept drift*. Já o mudança incremental ocorre por pequenas mudanças ao longo do tempo (LEE; PARK, 2022).

Figura 3 – Formas de ocorrer o *real concept drift*.



Fonte: Adaptado de Lee e Park (2022)

Existem diversos métodos para detectar *concept drift*, alguns dos presentes na literatura são o *Drift Detector Mechanism* (DDM) (GAMA et al., 2004), o *Feature Extraction for Explicit Concept Drift Detection* (FEDD) (CAVALCANTE; MINKU; OLIVEIRA, 2016), o *Exponential Moving Average Detector* (EMAD) (CASTELLANI; SCHMITT; HAMMER, 2021), o *Linear Four Rates* (LFR) (WANG; ABRAHAM, 2015), o *Exponentially Weighted Moving Average* para *Concept Drift*

Detection (ECDD) (ROSS et al., 2012) e o *Z-Score Detector* (ZSD) (CASTELLANI; SCHMITT; HAMMER, 2021) (CASTELLANI; SCHMITT; HAMMER, 2021; LIU et al., 2020; LIN et al., 2019). Este trabalho da prioridade para o método baseado em sensores proposto por Oliveira et al. (2017), que foi o detector usado nos experimentos, pois o mesmo demonstrou em experimentos realizados pelo autor que possui maior acurácia quando usando em um ambiente que possui diversos modelos previamente treinados no conceito anterior, como o usado nesse trabalho (OLIVEIRA et al., 2017).

Os sensores são modelos de aprendizado de máquina usados para detecção de *concept drift*. Então, dado um conjunto de modelos treinados para previsão de séries temporais, os sensores são formados por um subconjunto formado pelos melhores modelos. Logo, formalmente, o *concept drift* é identificado quando a condição da Equação (9) for satisfeita (OLIVEIRA et al., 2017).

$$Z_t > \mu_s + c \cdot \sigma_{Z_t}, \quad \forall \quad s \notin S \tag{9}$$

sendo que:

$$Z_t = (1 - \lambda) Z_{t-1} + \lambda \cdot e_t^s \tag{10}$$

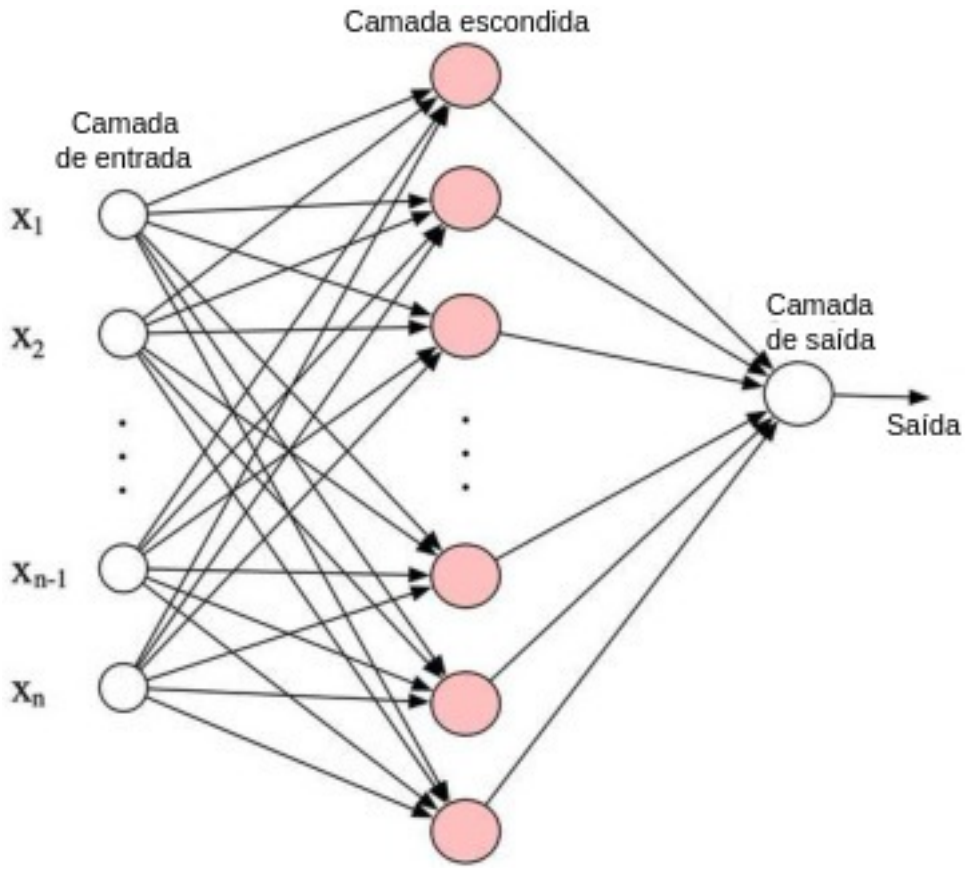
$$\sigma_{Z_t} = \sqrt{\frac{\lambda}{2 - \lambda}(1 - (1 - \lambda)^{2t})\sigma_s} \tag{11}$$

onde S é o conjunto de sensores, s é o sensor, μ_s é a média dos erros para um sensor, σ_s é o desvio padrão do conjunto de treinamento, o e_t^s é o erro de um determinado sensor no instante de tempo t, o λ indica o nível de importância dado aos erros de previsão mais recentes e a contante c regula o nível de tolerância para o erro de previsão. E, como descrito na Equação (9), a condição precisa ser atendida por todos os sensores (OLIVEIRA et al., 2017).

2.3 MODELO EXTREME LEARNING MACHINE

A *Extreme Learning Machine* (ELM) é uma rede neural que possui uma estrutura com apenas três camadas: camada de entrada, camada escondida e camada de saída. A Figura 4 ilustra a estrutura geral de uma ELM. Como principais vantagens, além da simplicidade, essa rede neural possui um rápido aprendizado e bom desempenho de generalização (PAN et al., 2020).

Figura 4 – Estrutura geral de uma ELM.



Fonte: Adaptado de Zhou et al. (2020)

O treinamento da ELM é feito de forma bem direta. Os pesos da camada de entrada e os *biases* da camada oculta são gerados aleatoriamente, portanto, muito processamento é poupado nessa fase do treinamento (ZHOU et al., 2020). Em seguida, os pesos da camada de saída (β) são obtidos usando:

$$\beta = H^{+}T \tag{12}$$

onde T é o vetor de saída esperado e H^{+} é a matriz inversa generalizada de Moore-Penrose, que é obtida por:

$$H^{+} = (H^{T}H)^{-1}H^{T}, \tag{13}$$

em que H é a matriz de saída da camada oculta, definida por:

$$H = \begin{bmatrix} g(a_1, b_1, x_1) & \cdots & g(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ g(a_1, b_1, x_N) & \cdots & g(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{14}$$

onde g é a função de ativação, a é o peso da entrada gerado aleatoriamente, b é a derivação do nó gerado aleatoriamente, x representa a entrada de treinamento, L é o número de camadas ocultas e N é o tamanho da amostra de treinamento (PAN et al., 2020). Essa forma de encontrar os valores para suas camadas deixa o treinamento do modelo eficiente e torna possível o uso em situações onde são necessários vários treinamentos.

2.4 PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) é uma técnica da computação evolutiva inspirada em comportamento de animais que vivem em bando (PRADHAN; BISOY; DAS, 2021). Foi inicialmente proposta por Kennedy e Eberhart (1995) ao observar o comportamento de pássaros ao encontrar comida e foi denominado *global PSO algorithm*.

O PSO, como um algoritmo de otimização, tenta encontrar a melhor solução possível, de forma iterativa, através do comportamento em grupo e compartilhamento de informações. O PSO é composto por um enxame de partículas, que nesse trabalho são modelos ELM, e a função objetivo, que vai medir se as partículas estão se aproximando ou se distanciando do alvo. É importante destacar que o algoritmo não garante encontrar a solução ótima, algumas vezes ele pode cair em um mínimo ou máximo local, mesmo possuindo mecanismos para evitar isso, e chegar nessa solução local como a final (PRADHAN; BISOY; DAS, 2021).

No início da busca as partículas são inicializadas em locais aleatórios e com velocidades aleatórias. A velocidade da partícula é um vetor que mostra a direção e intensidade em que a partícula vai se movimentar. Para cada partícula é calculada e mantida a melhor posição e uma melhor posição global. A próxima posição da partícula é definida pela posição atual somada com a velocidade. Depois que a partícula se movimenta outra velocidade é calculada através da equação (BAI et al., 2021):

$$V_{im}^{k+1} = w * V_{im}^{k} + c_1 * r_1 * (P_{best,im}^{k} - X_{im}^{k}) + c_2 * r_2 * (G_{best,im}^{k} - X_{im}^{k}) \tag{15}$$

onde V_{im}^{k+1} é a próxima velocidade, w é um peso para a velocidade atual, V_{im}^{k} é a velocidade atual, c_1 e c_2 são os coeficientes, r_1 e r_2 são fatores aleatórios gerados a cada iteração, P_{best} é a melhor posição da partícula, X_{im}^{k} é a posição atual e G_{best} é a melhor partícula global. Depois de obter a nova velocidade, é calculada, através da função objetivo, a aptidão de cada uma das partículas do enxame e, em seguida, é atualizada a melhor posição de cada partícula e a melhor posição global, caso seja necessário. As iterações ocorrem até que atinja algum critério de convergência ou uma quantidade máxima de iterações (BAI et al., 2021).

2.5 ENSEMBLES

Em algumas situações ao treinar vários modelos é comum obter a vários com desempenho semelhantes, e em seguida, torna-se necessário aplicar um ou mais critérios para escolher qual usar. Em situações como essas se torna claro que o uso de todos eles ao mesmo tempo pode ser uma solução interessante para o problema (WITTEN; FRANK, 2011).

No dia-a-dia, fora da ciência da computação, temos exemplos de *ensembles*, ou combinações de vários especialistas, acontecendo, como por exemplo a própria democracia, onde existem votações para escolha do resultado de uma eleição ou plebiscito. É comum também ver conselhos ou diretores de grandes empresas que resolvem questões difíceis por meio do voto. Em alguns casos também é montado um comitê na área médica, onde diversos especialistas analisam um caso em conjunto e chegam a um diagnóstico, nesse caso quando o diagnóstico não é unanime pode ser resolvido por convencimento dos outros profissionais ou com exames mais detalhados que gerem provas mais concretas (WITTEN; FRANK, 2011).

Então, voltando para ciência da computação, no uso de *ensembles*, a decisão é tomada por um conjunto de modelos, muitas vezes chamados de especialistas, para que em conjunto cheguem a uma decisão mais confiável, com menores flutuações. As principais desvantagens do uso de *ensembles* é a grande quantidade de treinamentos realizados, o que pode necessitar um maior poder de processamento, e em uma situação onde seja necessário entender o motivo que levou a decisão, será preciso auditar diversos modelos (WITTEN; FRANK, 2011).

As combinações podem ser feitas de várias formas, as mais comuns são *Bagging* (BREIMAN, 1996), *Randomization* (KWOK; CARTER, 1990), *Boosting* (FREUND, 1995) e *Stacking* (WOLPERT, 1992). O *Bagging* (BREIMAN, 1996) é a maneira mais simples, onde vários modelos diferentes, que podem ser tanto modelos de tipos diferentes ou vários de um mesmo tipo com treinamentos diferentes, realizam a previsão e é utilizado um operador para chegar a

decisão final (BREIMAN, 1996). Esse operador pode ser, por exemplo, votação em casos de classificação ou média, mediana ou moda em casos de regressão (KOURENTZES; BARROW; CRONE, 2014). O *Randomization* (KWOK; CARTER, 1990) é muito similar ao *Bagging*, onde sua principal diferença é o componente aleatório, em que muitas vezes pode ser o mesmo modelo treinado da mesma forma, só que com sementes aleatórias diferentes. Essa diferença é inspirada em diversas estratégias já são adotadas por alguns modelos para evitar mínimos locais. O *Boosting* (FREUND, 1995), por sua vez, procura modelos que se complementam, diferente do *Bagging* que cria modelos individuais, o *Boosting* usa os erros anteriores como foco para o treinamento das próximas instâncias. Por fim, o *Stacking* (WOLPERT, 1992) que também é similar ao *Bagging*, mas tem como sua principal característica o uso de um outro modelo de aprendizado de máquina como operador, ou seja, para combinar os resultados dos seus especialistas (WITTEN; FRANK, 2011).

2.6 ALGORITMO GULOSO

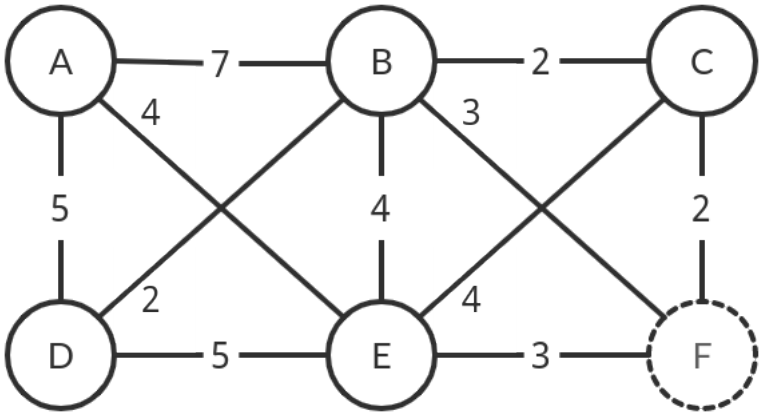
O algoritmo guloso é uma ferramenta utilizada em problemas de otimização, em que a busca por uma solução é feita enquanto minimiza ou maximiza alguma propriedade. De forma geral o algoritmo toma decisões localmente ótimas e espera que essas decisões o levem para uma solução ótima global, porém essa abordagem nem sempre leva para a melhor solução possível (GOODRICH; TAMASSIA; GOLDWASSER, 2013).

Para ilustrar o funcionamento do algoritmo guloso, o grafo da Figura 5 será utilizado como exemplo para encontrar o menor caminho entre o nó A e o nó F. Então, buscando inicialmente a melhor escolha local é ir para o nó E, que possui o menor peso entre as opções, em seguida ir para o nó F, que novamente possui o menor peso, e assim chega em um caminho ótimo. Usando a mesma figura para buscar um caminho entre o nó D e F, encontramos uma situação diferente. Inicialmente o menor peso é para ir para o nó B, em seguida para o nó C e por fim chegando ao nó F, que fica ao todo com o peso de 6. Porém se usasse o caminho D-B-F o peso seria somente 5, que é a escolha ótima para esse exemplo.

Como observado no exemplo a estratégia gulosa nem sempre leva ao menor caminho possível, porém em diversas situações ela pode ser a melhor estratégia como é o caso de problemas NP-completo e NP-difícil, em que se torna a melhor alternativa diante da força bruta que a solução pode demorar anos para ser encontrada (NIAZADEH et al., 2021).

Nesse trabalho o algoritmo guloso terá um papel diferente do PSO, mesmo ambos sendo

Figura 5 – Grafo bidirecional para busca do menor caminho.



Fonte: O autor

técnicas de otimização. Ele é usado em um ponto de decisão onde busca uma combinação de modelos possuindo características semelhantes a um grafo, que se distancia da maneira ideal de funcionamento do PSO.

2.7 MEDIDAS DE DESEMPENHO

Durante o treinamento e depois da criação dos modelos, é importante conhecer sua eficiência e se eles têm capacidade de resolver o problema com desempenho superior aos realizados anteriormente disponíveis na comunidade científica ou, caso não haja referências, ao desempenho de uma decisão aleatória (BALLABIO; GRISONI; TODESCHINI, 2018). Os instrumentos usados para isso são as medidas de desempenho e em todos os trabalho de *machine learning* pelo menos uma medida é usada. As mais comuns em trabalhos de regressão são o *Mean Square Error* (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2, \tag{16}$$

o *Root Mean Square Error* (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2}, \tag{17}$$

o *Mean Absolute Error* (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|X_i - Y_i| \tag{18}$$

e o *Mean Absolute Percentage Error* (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|X_i - Y_i|}{Y_i}, \tag{19}$$

onde o X_i é o valor previsto, o Y_i é o valor real e o n é a quantidade de pontos analisados (CHICCO; WARRENS; JURMAN, 2021). Para essas medidas quanto menor seu valor, implica em uma previsão mais próxima dos valor real (LAHMIRI, 2016). Este trabalho utiliza o MAE, pois é uma medida absoluta, então não importa se o erro é negativo ou positivo ele será considerado da mesma forma, e não possui distorções causadas por raiz quadrada ou divisão por um número que muda ao longo da série, tornando-se assim uma medida mais adequada para o sensor usar na identificação do *concept drift*.

2.8 TESTE ESTATÍSTICO DE FRIEDMAN-NEMENYI

O teste de Friedman-Nemenyi é composto por dois testes, o teste de hipótese não paramétrico de Friedman e o teste post-hoc de Nemenyi. O teste de Friedman também é conhecido como teste de *ranks*, pois o resultado é o ranqueamento dos dados testados (ZHANG et al., 2021; SHELDON; FILLYAW; THOMPSON, 1996). Ele pode ser facilmente calculado segundo as Equações (20) e (21), onde k é o número de conjuntos, N é o tamanho de cada conjunto e r_i é o *rank* médio da conjunto i.

$$\tau_F = \frac{(N-1)\tau_{\chi^2}}{N(k-1) - \tau_{\chi^2}} \tag{20}$$

$$\tau_{\chi^2} = \frac{12N}{k(k+1)}\left(\sum_{i=1}^{k} r_i^2 - \frac{k(k+1)^2}{4}\right) \tag{21}$$

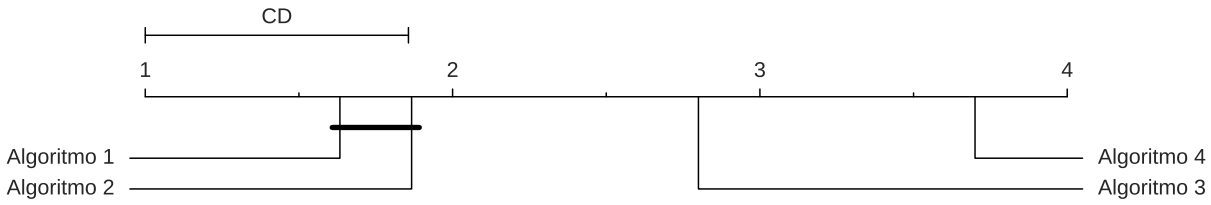
Depois de encontrar os valores dos *ranks*, pode ser feito o teste post-hoc de Nemenyi, que calcula a diferença crítica (CD, do inglês critical difference), que indica que há diferença significativa quando os valores encontrados no teste de Friedman possuem uma distância

superior ao valor encontrado para CD. A CD é calculada de acordo com a Equação (22), onde q_α é o valor crítico de acordo com a distribuição de Tukey (ZHANG et al., 2021).

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{22}$$

O teste de Friedman-Nemenyi pode ter seu resultado final representado de forma gráfica. A Figura 6 mostra um exemplo de resultado de um teste aplicado nos algoritmos 1, 2, 3 e 4, que, por sua vez, foram testados usando as saídas de cada um dos algoritmos. Para montar o gráfico foram utilizadas as saídas para calcular o τ_F, que estão representadas no eixo vertical, e também calculado a CD, que está representada um pouco acima do eixo. Para os algoritmos serem considerados diferentes a distância entre eles deve ser maior que a CD representada. Logo, nessa imagem os algoritmos 1 e 2 não são estatisticamente diferentes, porém eles são considerados diferentes do 3 e do 4, que também são diferentes entre si.

Figura 6 – Gráfico que representa o resultado do testes de Friedman-Nemenyi.



Fonte: O autor

3 TRABALHOS RELACIONADOS

A pesquisa para detecção de *concept drift* é mais comum em problemas de classificação. Porém, na literatura é possível encontrar trabalhos que abordam regressão com previsão de séries temporais na presença de *concept drift*. Desta forma, foi realizada uma pesquisa por trabalhos relacionados visando abranger uma quantidade significativa de trabalhos, e para isso foram utilizados critérios diferentes para aplicações em regressão e classificação. Os critérios utilizados para trabalhos de classificação são ter sido publicado entre os anos de 2019 e 2022, possuir algum método para detecção de *concept drift* e possuir alguma forma de adaptação. Já para os trabalhos de regressão são ter sido publicado entre os anos de 2015 e 2022 e possuir alguma identificação de *concept drift*.

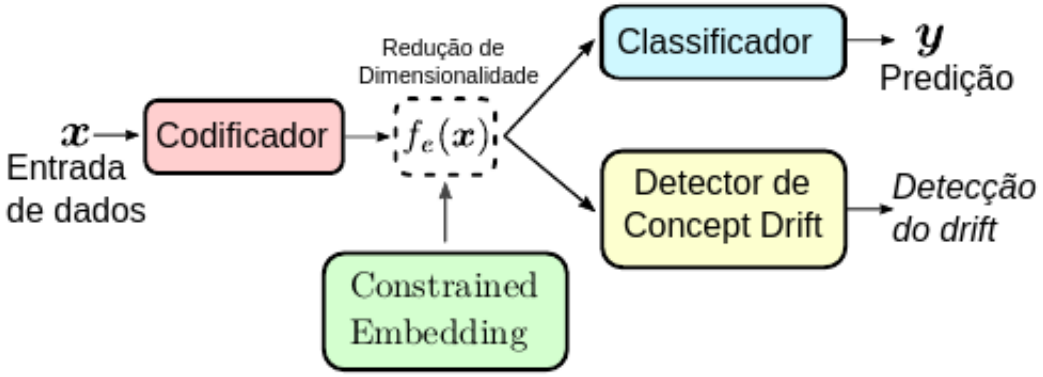
3.1 TRABALHOS DE CLASSIFICAÇÃO

Os trabalhos tratados aqui dividem o problema de *concept drift* em duas etapas: detecção e adaptação. Os autores Eliades e Papadopoulos (2021), por exemplo, usaram como base um trabalho chamado *Inductive Conformal Martingales* (ICM) (VOLKHONSKIY et al., 2017) e realizaram uma mudança no método de detecção de *concept drift*, que, por sua vez, é um teste de hipótese que calcula o *p-value* e o *martingale*, uma medida estatística de aleatoriedade da série, usando o valor atual e os valores passados, caso passe do limiar definido no teste o *concept drift* é identificado. O novo método impõe uma limitação para o valor de martingale e assim permite uma recuperação mais rápida e, consequentemente, uma detecção mais rápida. Após a detecção, outro treinamento é realizado com os dados do novo conceito. Os autores afirmam que ao aplicar essa técnica na detecção de *concept drift* conseguiram uma melhoria de acurácia, chegando a dez pontos percentuais em alguns casos, e redução do tempo de detecção de *concept drift* em relação ao método ICM.

Diferente do trabalho anterior, Castellani, Schmitt e Hammer (2021) criaram uma nova estrutura de detecção de *concept drift* semi supervisionada e sensível a tarefas. A Figura 7 ilustra o fluxo da informação na estrutura proposta, onde inicialmente o dado é codificado usando o *one-hot encoding*, em paralelo o módulo de *Contrained Embedding* deixa as distâncias das interclasses maiores e as intraclasses menores. Essas informações são combinadas e passam por um processo de redução de dimensionalidade. O resultado desse processamento servirá

como entrada para alimentar o classificador e o detector de *concept drift*, que, por sua vez, fornecem a previsão e se está ou não acontecendo um *concept drift*, respectivamente. Os autores propuseram os detectores o EMAD (CASTELLANI; SCHMITT; HAMMER, 2021) e o ZSD (CASTELLANI; SCHMITT; HAMMER, 2021) e usaram nos seus experimentos, mas ressaltam que esses detectores podem ser substituídos por qualquer detector não supervisionado. Os autores usaram nove diferentes séries nos experimentos com diferentes tipos de *concept drift* e uma medida de desempenho que acumula a acurácia, taxa de falso positivos e o atraso na detecção do *concept drift*. O método com os detectores propostos conseguiram bons resultados em detecções dos *concept drift* reais, além de ter ignorado boa parte dos *concept drift* virtuais, já que, segundo eles, esse tipo de *concept drift* não afeta a precisão final do classificador. Esses resultados foram comparados usando o mesmo método com os detectores *Hellinger Distance Drift Detector Method* (HDDDM) (DITZLER; POLIKAR, 2011) e *Incremental Kolmogorov–Smirnov test* (IKS) (REIS et al., 2016) e aplicado o teste estatístico de Friedmen-Nemenyi.

Figura 7 – Método criado por Castellani, Schmitt e Hammer (2021) para detecção de *concept drift*.



Fonte: Adaptado de Castellani, Schmitt e Hammer (2021)

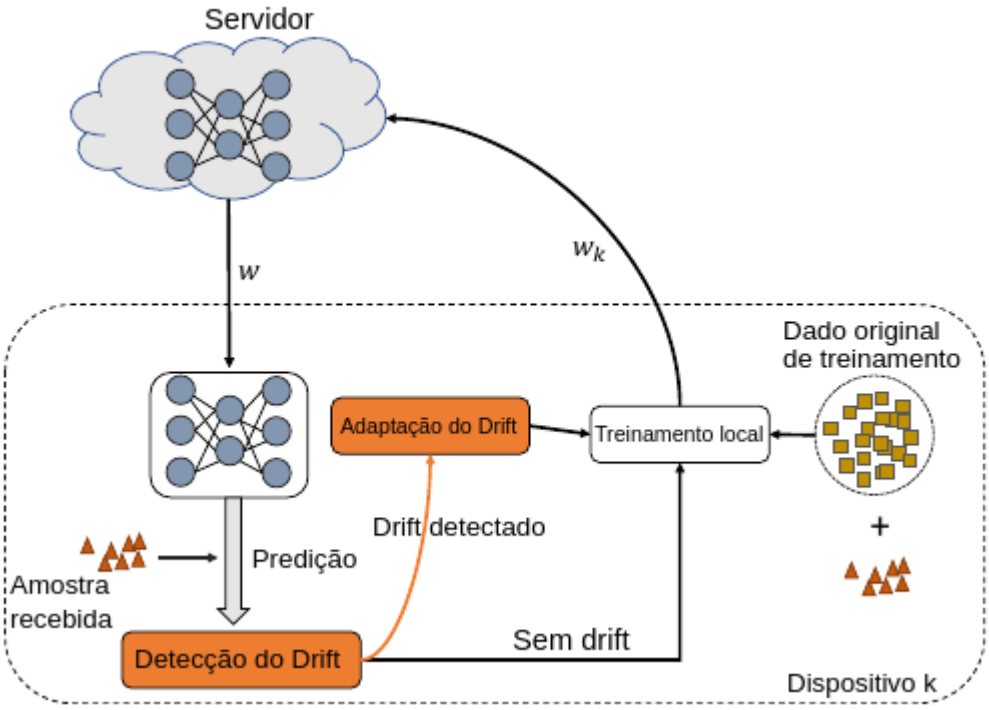
Os trabalhos citados até aqui fazem a previsão, detecção de *concept drift* e, possivelmente, a adaptação, mas nenhuma ação é feita no período em que o *concept drift* é identificado até que um novo modelo, com o novo conceito, esteja pronto para substituir o antigo. Já o método feito por Hidalgo, Santos e Barros (2021) trabalham nesse período, eles usaram um ajuste dinâmico de parâmetros na presença de mudança de conceito. Eles usam o método de estimação de parâmetros para ajuste da diversidade (λ) nos *ensembles Boosting-like Online Learning Ensemble* (BOLE) (BARROS; SANTOS; JÚNIOR, 2016), *Online AdaBoost-based* M1 (OABM1) (SANTOS; BARROS, 2020) e Oza and Russell's Online Bagging (OzaBag) (OZA;

RUSSELL, 2001), criando assim um novo método para cada um deles BOLE-PE, OABM1-PE e OzaBag-PE, consequentemente. Os autores comparam os novos métodos criados com os *ensembles* sem modificação e com um λ fixo de 6. Esse método foi aplicado em problemas de classificação com *concept drift* abruptos e graduais presentes em quatro séries artificiais e nove séries reais. O método proposto obteve resultados melhores que seus pares, porém não teve diferença significativa do método com um λ fixo de 6.

Os autores Alshalali e Josyula (2020), por sua vez, montaram uma configuração experimental para comparar abordagens usando o ELM, que é um modelo simples, e algumas de suas variações, com redes neurais mais robustas como o *Recurrent Neural Network* (RNN), *Long Short-Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) e *Gated Recurrent Units* (GRU) (CHO et al., 2014) em cenários com *concept drift*. Esse trabalho concluiu que em *data stream* com *concept drift* o modelo usando o *Online Recurrent* ELM (OR-ELM) (PARK; KIM, 2017) tem uma acurácia maior, além do tempo de treinamento ser muito menor que as outras abordagens mais complexas. Com isso, os autores buscam mostrar que a reação ao *concept drift*, em diversos casos, é mais importante que a robustez do modelo em si e para eles o uso de um modelo simples permitiu que fossem realizados mais treinamentos e mais análises com um tempo de resposta mais curto.

Concept drift pode ocorrer em diversos contextos, os autores Chen et al. (2021), por exemplo, viram aplicação de detecção e tratamento de *concept drift* em um cenário distribuído. Os autores propuseram um método baseado no *Federated Learning* (FL) (MCMAHAN et al., 2017) onde dispositivos, de forma distribuída, treinam um classificador centralizado sem revelar os seus dados. O FedConD (CHEN et al., 2021), nome dado ao método criado pelos autores, detecta e lida com o *concept drift* de forma distribuída e assíncrona. Em um caso sem *concept drift*, o método recebe um classificador central e usa para realizar as classificações enquanto realiza um treinamento local para combinar com os demais. Localmente, o dispositivo já consegue usar as classificações do modelo central e com essas previsões já consegue também verificar, usando teste estatístico, se houve *concept drift*. O tratamento realizado quando detecta o *concept drift* é aplicar uma penalização na combinação dos métodos, conseguindo assim aumentar ou diminuir a ênfase para o novo classificador recém treinado. A Figura 8 mostra os passos dentro do dispositivo. Esse método apresentou bons resultados em relação aos métodos FedAvg (MCMAHAN et al., 2017), FedProx (LI et al., 2020) e ASO-Fed (CHEN et al., 2020), que são métodos síncronos e assíncronos também baseados no FL. Segundo os autores, esse é o primeiro estudo de *concept drift* em um contexto de FL.

Figura 8 – Método FedConD usado para detectar e lidar com *concept drift* em contexto distribuído.



Fonte: Adaptado de Chen et al. (2021)

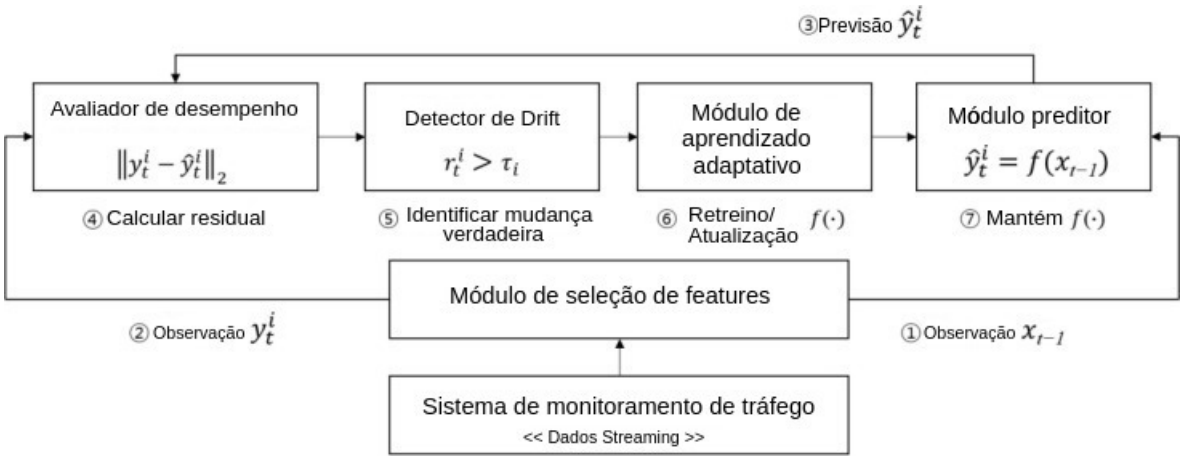
Um outro contexto diferente foi pensado por Lin et al. (2019) que aplicou um *ensemble* de aprendizado offline com detecção de *concept drift* em um problema da indústria. O objetivo era detectar quando um componente se deteriora e assim evitar maiores prejuízos com produtos defeituosos ou de baixa qualidade. Para isso, um *framework* chamado DAMSID foi construído neste trabalho baseando-se em outro método chamado *Diversity for Dealing with Drifts* (DDD) (MINKU; YAO, 2011). Nele existem três estágios: o primeiro é o treinamento do *ensemble* com o AdaBoost.NC (WANG; ABRAHAM, 2015) e o uso do *Synthetic Minority Oversampling TEchnique* (SMOTE) (CHAWLA et al., 2002) para lidar com o desbalanceamento; o segundo tem o objetivo de detectar o *concept drift* e para isso ele usa o LFR (WANG; ABRAHAM, 2015); e o terceiro retreina para adaptar-se ao novo conceito e combina com o *ensemble* previamente treinado. Esse método não fez nenhuma otimização durante a coleta dos dados quando o *concept drift* é detectado. Nos experimentos eles conseguiram, com esse método, detectar todos os *concept drift* e, usando dados com diferentes taxas de desbalanceamento, eles conseguiram uma alta precisão nas classes minoritárias (LIN et al., 2019).

3.2 TRABALHOS DE REGRESSÃO

Trabalhos com tratamento de *concept drift* relacionados a regressão, mesmo sendo mais interessantes para o tema abordado, são menos abundantes, um exemplo, é o trabalho realizado por Adams et al. (2021) tendo como objetivo criar um *framework* para identificar, usando mineração de dados, a causa do *concept drift*. O objetivo do trabalho é um pouco diferente dos abordados até então, mas possui ideias que são interessantes para o tema. Inicialmente eles separaram a entrada em duas perspectivas, a primeira com o processo a ser analisado, que é o fluxo em si. A segunda, por sua vez, depende do domínio do problema e deve ser escolhido recursos com impacto no desempenho do que está sendo avaliado. Os autores dão como exemplos a carga de trabalho nos tempos de serviço, a carga de trabalho nos casos de horas extras ou os dados associados na duração do caso, entre outros. As duas são utilizadas para detectar o *concept drift*, porém a segunda também será usada para descobrir a causa. A detecção do *concept drift* pode ser feita usando técnicas baseadas em custo, testes de hipóteses ou de clusterização. Quando é identificado um desvio na primeira perspectiva, é analisado os desvios anteriores, da segunda perspectiva, e, em seguida, testado se pode ser, potencialmente, a causa do desvio primário. Os exemplos aplicados no trabalho mostram que esse método possui um bom potencial para explicar a causa-efeito dos *concept drift*.

Similar a alguns trabalhos de classificação comentados, os autores Lee e Park (2022) conseguiram visualizar o problema de *concept drift* em um contexto bastante explorado, só que de forma imutável. Eles aplicaram a modelagem de *concept drift* para criar um sistema robusto para carros autônomos. Onde eles contam que existem diversas propostas para prevenir congestionamento de tráfego, porém poucos tratam os problemas relacionados a *concept drift* que são comuns em uma possível aplicação real. O sistema conta com cinco módulos, como podem ser vistos na Figura 9, sendo eles: i) Módulo de monitoramento de tráfego; ii) Módulo de seleção de *features* que cria um subconjunto de características relevantes e independentes para o modelo de previsão; iii) Módulo de previsão que utiliza um modelo linear, um modelo não linear baseado em *kernel* e uma rede neural profunda; iv) Módulo de *concept drift* baseado no desempenho do modelo preditivo; e o v) Módulo de aprendizado adaptativo que atualiza o modelo quando uma mudança é detectada. Durante o experimento os autores conseguiram demonstrar a importância da detecção de *concept drift* para esse problema e conseguiram resultados superiores a outros trabalhos que não lidam com o problema da mesma forma.

Figura 9 – Sistema proposto para modelagem de *concept drift* proposta para um sistema de controle de veículo autônomo.



Fonte: Adaptado de Lee e Park (2022)

Já o trabalho realizado por Liu et al. (2020) usa uma abordagem chamada meta-cognitiva, para problemas de *concept drift*, composta por quatro etapas: primeiro inicializa com um pequeno número de nós escondidos possível; em seguida, quando acontecer o *concept drift*, aumenta a quantidade de neurônios; re-treina os pesos de saída com a amostra atual; por fim, descarta neurônios quando ultrapassa o número máximo permitido. Eles usam uma variação do ELM, o *Kernel Online Sequential ELM* (KOS-ELM) (SCARDAPANE et al., 2014), usando como detectores o DDM (GAMA et al., 2004) e ECDD (ROSS et al., 2012). O experimento realizado foi com séries sintéticas e reais, como S&P500, o principal índice da bolsa de Shanghai e a concentração de ozônio em Toronto. Nos experimentos o *kernel* DDM teve uma maior capacidade de detectar os *concept drifts* e a abordagem metacognitiva descrita teve bons resultados e com baixo tempo de processamento, que é um dos principais objetivos, mas não demonstrou uma diminuição considerável nos erros, em alguns casos até aumentou.

No trabalho realizado por Cavalcante e Oliveira (2015) foi proposta uma combinação de ELM com os detectores ECDD (ROSS et al., 2012) e DDM (GAMA et al., 2004). Criando assim os métodos ELM-ECDD e ELM-DDM, nesses métodos o ELM é treinado e quando um *concept drift* é detectado ele é retreinado com os dados do novo conceito. Os autores também propuseram dois métodos online usando o OS-ELM com os mesmos detectores, criando assim os métodos OS-ELM-ECDD e OS-ELM-DDM, nesses métodos o modelo alterna o estado entre *online* e *offline*, ficando *online* quando os detectores encontram um *concept drift* e *offline* quando se adaptam. Os métodos foram comparados com o ELM e com o OS-ELM, em

termos do tempo de processamento e desempenho, usando o MAPE. Em relação ao tempo de processamento o uso do ELM sem detector é melhor em todos os casos, porém ficando com baixo desempenho na maioria das séries testadas, comparando com os ELM com os detectores. Os modelos online conseguiram melhor tempo de processamento quando usaram os detectores e em três das nove séries testadas o OS-ELM-ECDD conseguiu melhor desempenho e nas outras não tiveram diferença significativa.

O método *Improved Self-Adaptive Particle Swarm Optimization-ELM-Sensor* (IDPSO-ELM-S) proposto por Oliveira et al. (2017) consiste em um treinamento de um enxame de ELM, onde o melhor modelo é utilizado para fazer previsões até que ocorra um *concept drift*. Para detectar o *concept drift* as próprias partículas, modelos treinados pelo enxame, são usadas como sensores. Por fim, depois que o *concept drift* é detectado o método passa a armazenar as informações do novo conceito e quando chega a quantidade n, que pode ser configurada mas nas simulações foram utilizadas como 300, um próximo treinamento é realizado e a partícula que realiza as previsões e os sensores são atualizados. O método proposto conseguiu um desempenho melhor que os trabalhos ELM-ECDD (CAVALCANTE; OLIVEIRA, 2015) e o ELM-DDM (CAVALCANTE; OLIVEIRA, 2015), além disso método de detecção de *concept drift* baseado em sensores conseguiu uma acurácia na detecção melhor que os dois trabalhos já citados e o FEDD (CAVALCANTE; MINKU; OLIVEIRA, 2016) (OLIVEIRA et al., 2017). Esse método tem bom desempenho antes do *concept drift* e depois da adaptação, porém existe uma limitação durante a adaptação, ou seja, no momento em que coleta dos dados para o novo treinamento. Durante essa fase o método faz previsões com base no conceito antigo e o erro aumenta exponencialmente. Então os métodos propostos neste trabalho usam este trabalho como base para construção de um modelo mas robusto.

4 MÉTODO PROPOSTO

Este capítulo trata dos métodos desenvolvidos, apresentando para cada um deles um fluxograma, uma descrição e uma justificativa, o pseudocódigo dos métodos está disponível no Anexo A. São três métodos no total e foram criados de forma incremental. O primeiro método tem como principal característica o uso de *ensemble* e será referido aqui apenas como *Ensemble*, o segundo método que tem como principal característica se adaptar durante o *concept drift* com o melhor *ensemble* será chamado de *BE drift* (abreviação do inglês, *Best Ensemble - drift*), o terceiro método que usa melhor *ensemble* antes, durante e depois do *concept drift* será chamado de *BE always* (abreviação do inglês, *Best Ensemble - always*). Os três são importantes para entender quais mudanças tiveram mais efeito em relação às outras e, além disso, são necessários para responder as perguntas feitas no início do trabalho.

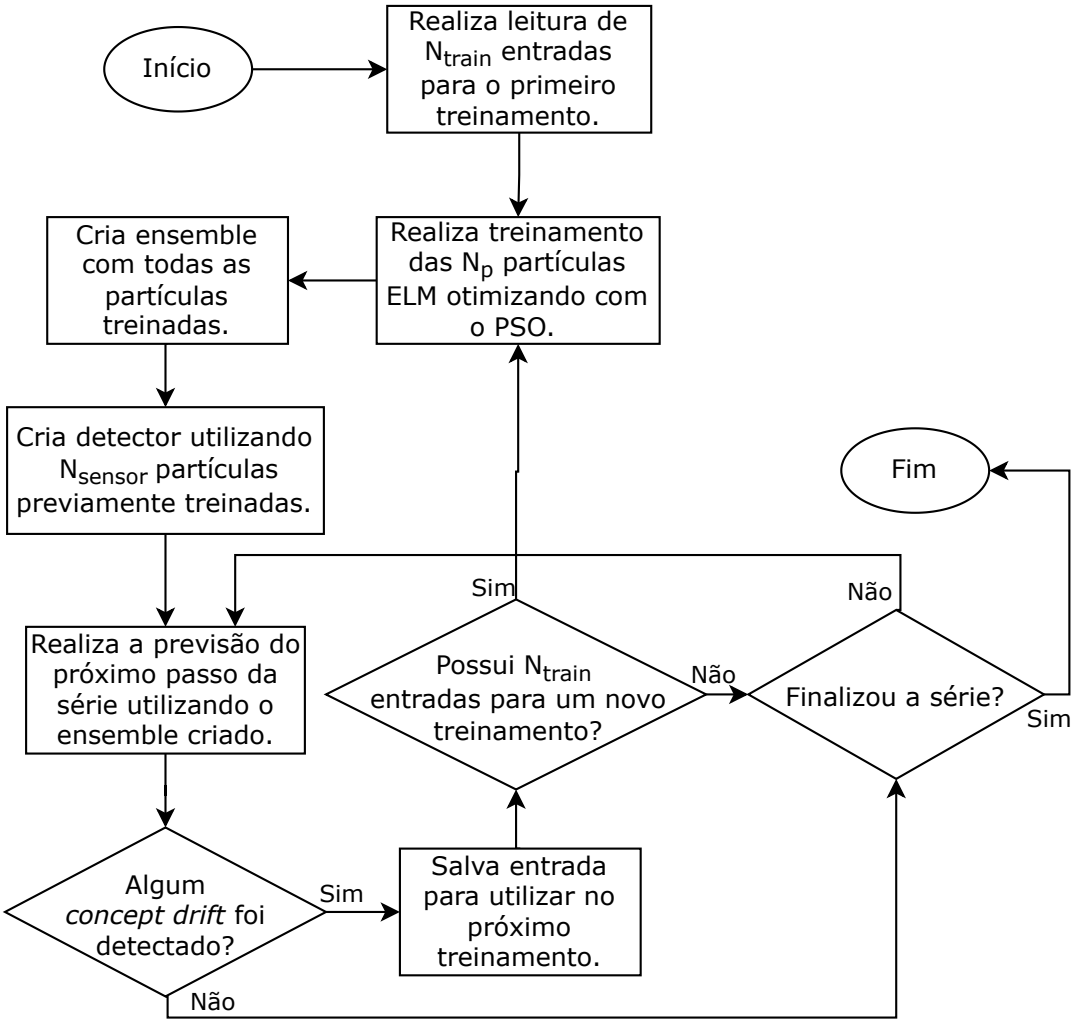
4.1 PRIMEIRO MÉTODO: *ENSEMBLE*

O primeiro método foi criado utilizando o IDPSO-ELM-S como referência, nele é treinado um enxame de partículas ELM, porém é utilizado somente a melhor para realizar as previsões. A principal característica do primeiro método construído é o uso do *ensemble* das partículas treinadas pelo PSO. Com o uso de valores centrais para previsão é esperado que o resultado do final melhore.

Como pode ser visto no fluxograma da Figura 10, o método inicia com a leitura dos dados, durante essa leitura nenhuma previsão é realizada, apenas guarda as informações para o treinamento. A quantidade de dados que ele usa para o treinamento é parametrizada e será especificada na Seção 5.2, que trata da configuração experimental, aqui será referido esse valor como N_{train}. O segundo passo é o treinamento do exame com os dados armazenados no passo anterior. O treinamento é feito usando a otimização do PSO para N_p, outro parâmetro do método, partículas ELM. O terceiro passo é a criação do *ensemble* para todas as partículas, para esse *ensemble* todas as partículas possuem pesos iguais e o operador usado para dar o resultado final é a mediana. Em seguida, no quarto passo, o *ensemble* passa a realizar previsões no conjunto de teste. No quinto passo é testado se aconteceu o *concept drift* usando o sensor, como descrito na Seção 2.2, e a quantidade de partículas usadas como sensores também é um parâmetro do método e será referida como as N_{sensor} melhores partículas. Caso não

seja identificado o *concept drift* e a série não tenha acabado, o método continua fazendo previsões e repetindo o teste normalmente. Mas em algum ponto da série quando o *concept drift* for identificado, o método passa a armazenar informações para um novo treinamento. Nesse ponto nada é modificado e o método volta ao passo de realizar previsões até que tenha dados suficientes (N_{train}) para um novo treinamento, então nesse caso o passo volta para a etapa de treinamento o fluxo é repetido.

Figura 10 – Fluxograma do método *Ensemble*.



Fonte: O autor

4.2 BUSCA GULOSA

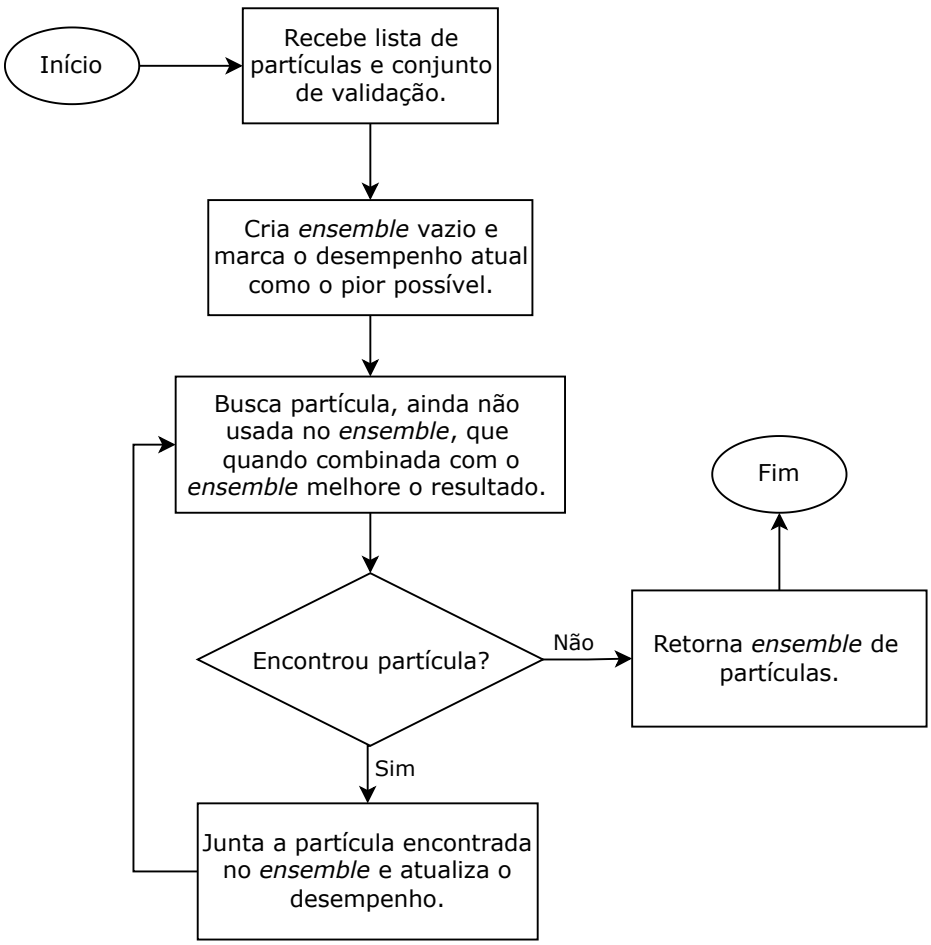
Essa seção não descreve um método em si, ao invés disso será descrito a busca que será utilizada pelos próximos métodos para montar a melhor combinação para o *ensemble*. A busca gulosa é originalmente um algoritmo de otimização para problemas que não é possível verificar todas as alternativas possíveis, como é o caso da busca pela melhor combinação de partículas usada nesse trabalho. A complexidade para a busca completa está na ordem de 2^{N_p}, onde N_p é o número de partículas ELM usadas, e caso sejam usadas, por exemplo, 30 partículas será feita mais de um bilhão de checagens, o que claramente é impossível de ser feito.

A busca pela melhor combinação, ilustrada no fluxograma da Figura 11, inicia com todas as partículas treinadas no enxame e o conjunto de dados de teste, em seguida cria um *ensemble* vazio e armazena o desempenho deste *ensemble* como o pior possível. Os passos anteriores foram feitos como preparação para a lógica da busca gulosa, que usa cada uma das partículas disponíveis combinando com o *ensemble* atual, e a melhor combinação encontrada é comparada com o desempenho atual. Caso este *ensemble* seja melhor, o desempenho é atualizado e a partícula é removida da lista de partículas disponíveis. Quando não possui mais partículas ou as demais combinações não trazem ganho de desempenho, o *ensemble* encontrado é considerado o melhor possível e será definido como modelo final de previsão. Apesar do método iniciar com o *ensemble* vazio o resultado final nunca será vazio, mas é possível que contenha apenas uma ou que contenha todas as partículas disponíveis, basta essa combinação feita com a mediana ter o resultado melhor que as outras testadas.

4.3 SEGUNDO MÉTODO: *BE DRIFT*

O segundo método difere do método anterior pelo uso do *ensemble* somente quando o *concept drift* é detectado, pois é nesse momento em que o modelo passa a realizar previsões mais distantes das reais e com isso aumenta o MAE geral do modelo. Então, durante o período de coleta de dados para o novo treinamento foram utilizados os modelos existentes, treinados com o conceito anterior, para realizar as previsões, mas ao invés de utilizar um *ensemble* com todos os modelos, foi utilizado um *ensemble* com a melhor combinação dos modelos. Esse *ensemble* também utiliza a mediana como operador de combinação dos resultados e o MAE como medida de desempenho para encontrar a melhor combinação. Essa combinação foi construída utilizando uma busca gulosa para identificar qual a melhor combinação para

Figura 11 – Fluxograma da busca gulosa pelo melhor *ensemble*.



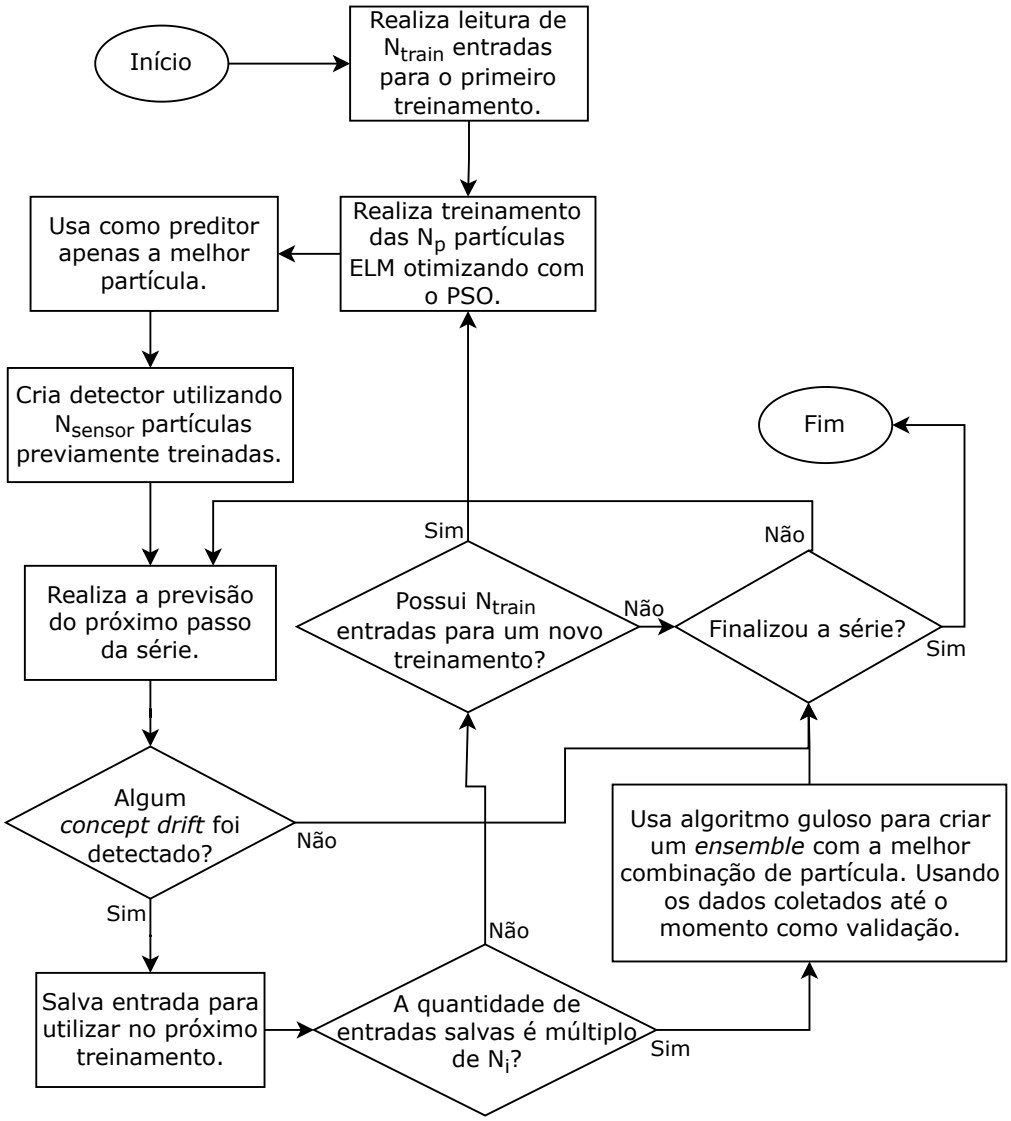
Fonte: O autor

fazer a previsão nesse período, como explicado na Seção 4.2. Para encontrar essa combinação foram utilizados os dados coletados durante esse período e usados como teste, para encontrar justamente qual é a melhor combinação no conceito atual. Essa busca foi feita em intervalos regulares de tamanho N_i, criando um novo *ensemble* a cada etapa e cada vez com mais informações desse novo conceito.

O segundo método, disponível na Figura 12, remove o uso de *ensemble* com todas as partículas feito no primeiro método e passa a usar somente a melhor partícula para previsões. Porém, quando o *concept drift* é detectado, é montado o *ensemble* com a busca gulosa, essa busca está representada no fluxograma como "cria um *ensemble* com a melhor combinação de particulas". É possível que a melhor combinação seja de somente uma partícula e com isso o uso de *ensemble* não é necessário. Para os casos onde foi feito uma combinação de mais de uma partícula a mediana foi o operador utilizado como método de combinação dos resultados,

pois experimentalmente ela obteve melhores resultados do que os demais operadores.

Figura 12 – Fluxograma do método *BE drift*.



Fonte: O autor

Este método difere do primeiro método na construção inicial do *ensemble* e a partir do ponto de coleta dos dados, depois que o *concept drift* é identificado. Nesse ponto, a cada N_i novos dados acumulados para o treinamento, é criado um *ensemble*, com as partículas treinadas no conceito anterior, que respondem melhor ao novo conceito, usando a busca gulosa. Depois da criação desse *ensemble*, o fluxo segue normalmente e ele será utilizado para as próximas previsões.

4.4 TERCEIRO MÉTODO: *BE ALWAYS*

O terceiro método usa o algoritmo busca gulosa para encontrar o melhor *ensemble* de partículas desde o início do método. O objetivo é entender se aplicando a mesma otimização usada no segundo método em todas as fases (antes do *concept drift*, durante a coleta de dados e depois da adaptação) traz uma melhoria em relação ao segundo método e entender a importância da fase em que o método está se adaptando no desempenho geral do método. Então, essas modificações podem ser vistas na Figura 13, onde os pontos do fluxograma em que é feita a escolha da melhor combinação está se referindo ao uso do algoritmo guloso que foi realizado como explicado na Seção 4.2.

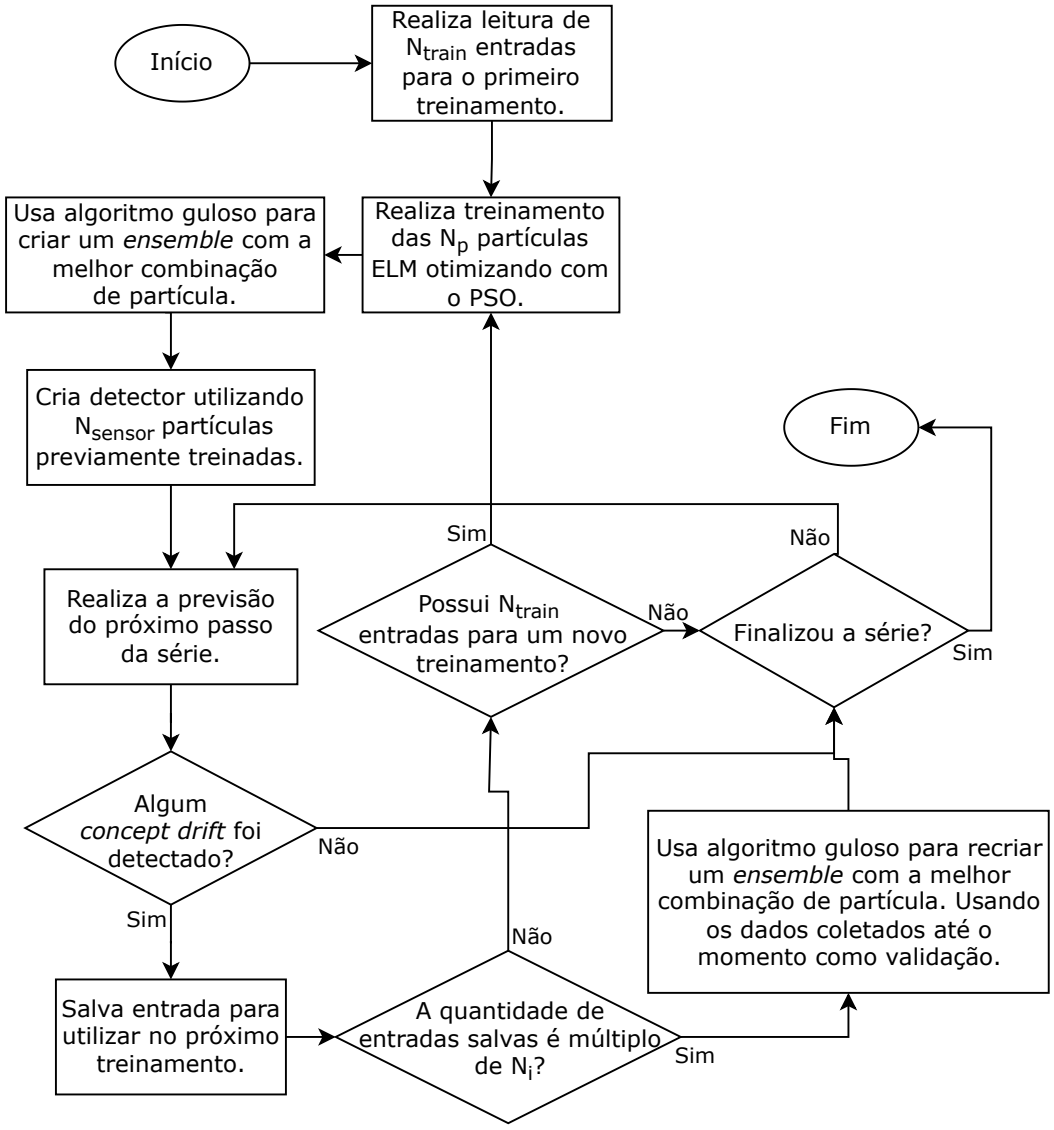
Esse método possui uma semelhança maior com o método *BE drift*, onde a diferença principal está depois que o enxame é treinado, que ao invés de utilizar somente a melhor partícula, ele utiliza também a busca gulosa para escolha da melhor combinação de partículas. Com isso a mesma otimização realizada durante a adaptação do *concept drift* também é feita com enxame recém treinado do novo conceito, constituindo assim o método final.

4.5 COMPARAÇÃO ENTRE OS MÉTODOS

Os três métodos possuem lógicas semelhantes, porém com mudanças específicas no que está sendo experimentado. Para facilitar o entendimento, a Tabela 1 traz um resumo das principais diferenças entre eles. As colunas mostram qual o método, como é feita a previsão antes de detectar o *concept drift* e o que é feito no período durante a adaptação, ou seja, entre o momento em que é detectado até o momento da adaptação. As linhas mostram para cada um dos métodos, o que eles fazem nessas situações.

Para o método com *Ensemble* é montado uma combinação de todas as partículas após o treinamento e mantido até o próximo treinamento. Para o *BE Drift* não é montado ensemble, porém quando o *concept drift* é detectado é montado um ensemble com a melhor combinação possível das partículas treinadas no conceito anterior e essa combinação é atualizada periodicamente até o próximo treinamento. Já para o *BE Always*, esse ensemble com as melhores partículas é montado desde o treinamento e passa a ser atualizado periodicamente quando o *concept drift* é detectado.

Figura 13 – Fluxograma do método *BE always*.



Fonte: O autor

Tabela 1 – Resumo das diferenças entre os métodos propostos.

Método	Antes da detecção do *concept drift*	Durante a adaptação do *concept drift*
Ensemble	*Ensemble* com todas as partículas	Não muda
BE Drift	Melhor partícula	Atualização periódica com a melhor combinação de partículas
BE Always	Melhor combinação de partículas	Atualização periódica com a melhor combinação de partículas

Fonte: O autor

5 EXPERIMENTOS

Este capítulo descreve todos os experimentos realizados. Inicialmente será explicitado todo o protocolo experimental, seguido pelos dados utilizados e necessários para execução do experimento, os resultados que foram obtidos e finaliza com a discussão dos resultados. Os experimentos realizados têm o objetivo de responder às perguntas de pesquisa feitas no Capítulo 1, usando os métodos descritos no Capítulo 4 e comparando com os métodos usado como referência, descritos no Capítulo 3.

5.1 PROTOCOLO EXPERIMENTAL

O experimento utiliza sete séries temporais com características distintas e descritas na Seção 5.3, com o objetivo de avaliar o desempenho dos métodos propostos em relação aos métodos usado como referência. Para isso, para cada experimento, foram executadas trinta vezes para cada uma das séries temporais, onde foi calculado o MAE para avaliar o desempenho preditivo. Por fim, para conferir estatisticamente os resultados obtidos foi empregado o teste de Friedman-Nemenyi.

5.2 PARÂMETROS DO MÉTODO

De acordo com o que foi explicado no Capítulo 4, o método proposto possui alguns parâmetros para configuração do experimento. Esses parâmetros servem para configurar algum comportamento do método e a mudança de algum valor pode interferir positivamente ou negativamente nos resultados. Todos os parâmetros disponíveis para esse método estão listados na Tabela 2 juntamente com os valores utilizados nesse experimento.

Pensando em diferentes cenários é possível imaginar a importância da parametrização do método, como por exemplo, se a aplicação for feita em uma série muito pequena ou muito grande, o tamanho da amostra de treinamento pode ser diminuído ou aumentado. Se o local de execução do método tiver menos ou mais poder de processamento, o método pode ser configurado em relação ao número de partículas, ao intervalo de reconstrução do *ensemble* ou ao número de neurônios para atender essa necessidade.

Os parâmetros usados foram escolhidos empiricamente, em alguns casos aproveitando

Tabela 2 – Parâmetros do método usados nas execuções.

Símbolo	Descrição	Valor utilizado
N_{train}	Tamanho da amostra necessária para treinamento	300
N_p	Número máximo de partículas	30
N_{sensor}	Número de sensores	30
N_i	Intervalo de reconstrução do *ensemble*	10
$N_{neurons}$	Número de neurônios nos modelos ELM	10
N_{lags}	Número de lags	5
N_{it}	Número de iteração de treinamento para o PSO	20
P_{train}	Proporção de treinamento	80%
φ	Função de ativação	$\tanh$

Fonte: O autor

valores sugeridos na literatura, como a proporção de treinamento (RÁCZ; BAJUSZ; HÉBERGER, 2021). O número de iterações de treinamento do PSO foi mantida intencionalmente baixa para manter as partículas mais diferentes entre si, e dessa forma fazer sentido o uso em um *ensemble* e aumentar as chances na busca por uma combinação que seja capaz de melhorar a previsão durante a fase de adaptação do modelo.

5.3 DADOS UTILIZADAS

No total, durante o experimento foram usadas sete séries, sendo que quatro delas foram criadas sinteticamente e três foram retiradas do mercado financeiro americano. As séries sintéticas possuem um total de 20 mil pontos cada, tendo elas 10 diferentes conceitos com 2 mil pontos cada conceito.

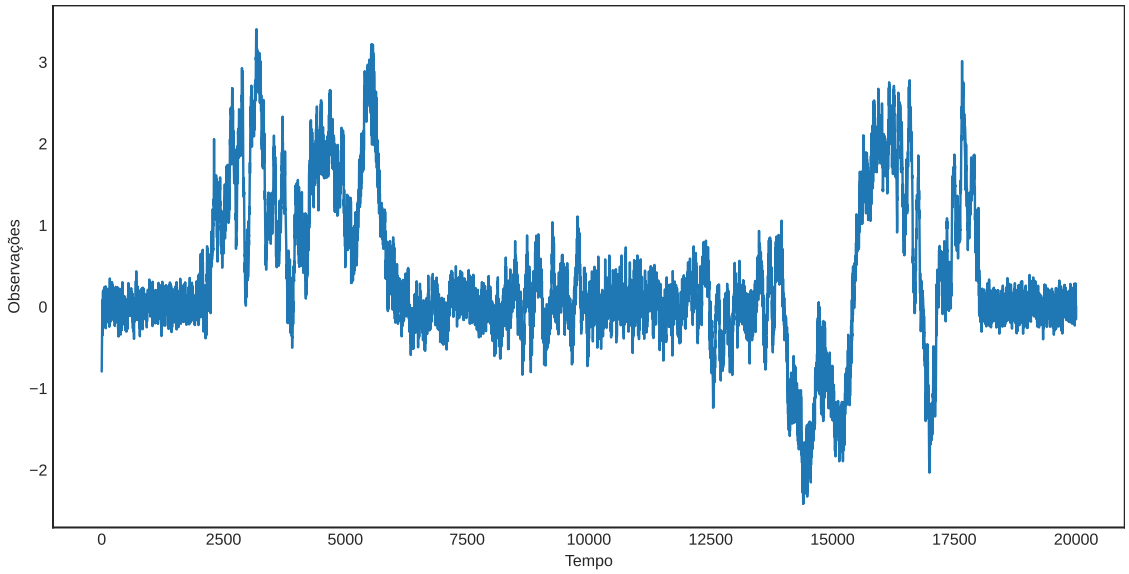
A série linear, ilustrada na Figura 14, foi criada usando o processo autoregressivo (AR), definido por:

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + ... + a_p x_{t-p} + w_t \tag{23}$$

em que $a_1, a_2, ..., a_p$ são parâmetros do modelo, definidos na Tabela 3, w_t é o ruido branco Gaussiano, t é o tempo que vai de 0 a 20 mil e p é a quantidade de parâmetros definido empiricamente como seis.

A série não linear, ilustrada na Figura 15, foi gerada por dois modelos de suavização definidos por Cavalcante, Minku e Oliveira (2016), que seguem as Equações 24 e 25. Os

Figura 14 – Série linear criada sinteticamente.



Fonte: O autor

parâmetros desse modelo estão disponíveis na Tabela 3.

$$x_t = [a_1 x_{t-1} + a_2 x_{t-2} + a_3 x_{t-3} + a_4 x_{t-4}] * [1 - exp(-10 x_{t-1})]^{-1} + w_t \tag{24}$$

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + [a_3 x_{t-1} + a_4 x_{t-2}] * [1 - exp(-10 x_{t-1})]^{-1} + w_t \tag{25}$$

A série sezonal, por sua vez, ilustrada na Figura 16, foi criada a partir de um modelo linear com períodos diferentes na série. Esse modelo é definido por:

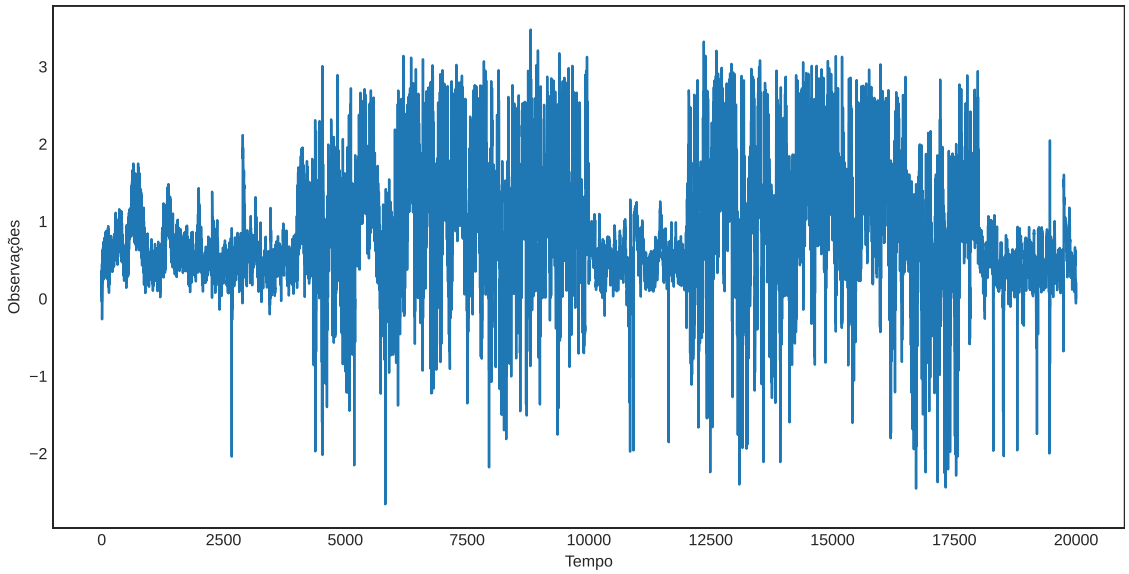
$$x_t = m_t + \beta_{1+mod(t-1,s)} + w_t \tag{26}$$

onde m_t é a tendência, $\beta_{1+mod(t-1,s)}$ é o fator sazonal e w_t é o ruido gaussiano. Os parâmetros desse modelo também estão disponíveis na Tabela 3.

A série híbrida, ilustrada na Figura 17, é uma mistura das séries linear, não linear e sazonal. Para cada uma das diferentes partes da séries utilizadas os parâmetros estão disponíveis na Tabela 3.

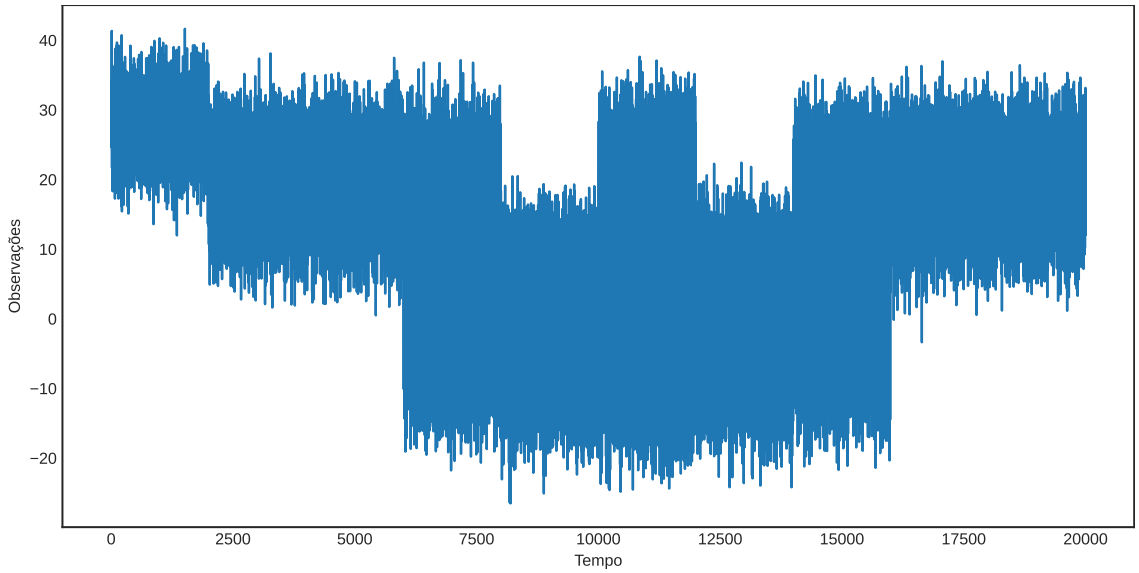
A primeira série real utilizada foi o índice Standard and Poor's 500 (S&P500), que é o índice ponderado das 500 maiores empresas americanas de capital aberto. Esse índice é considerado o melhor indicador de ações em grande escala e um indicador geral da economia dos Estados Unidos. Até o ano de 2021 o índice possui uma tendência positiva ao longo do tempo, porém

Figura 15 – Série não linear criada sinteticamente.



Fonte: O autor

Figura 16 – Série sazonal criada sinteticamente.



Fonte: O autor

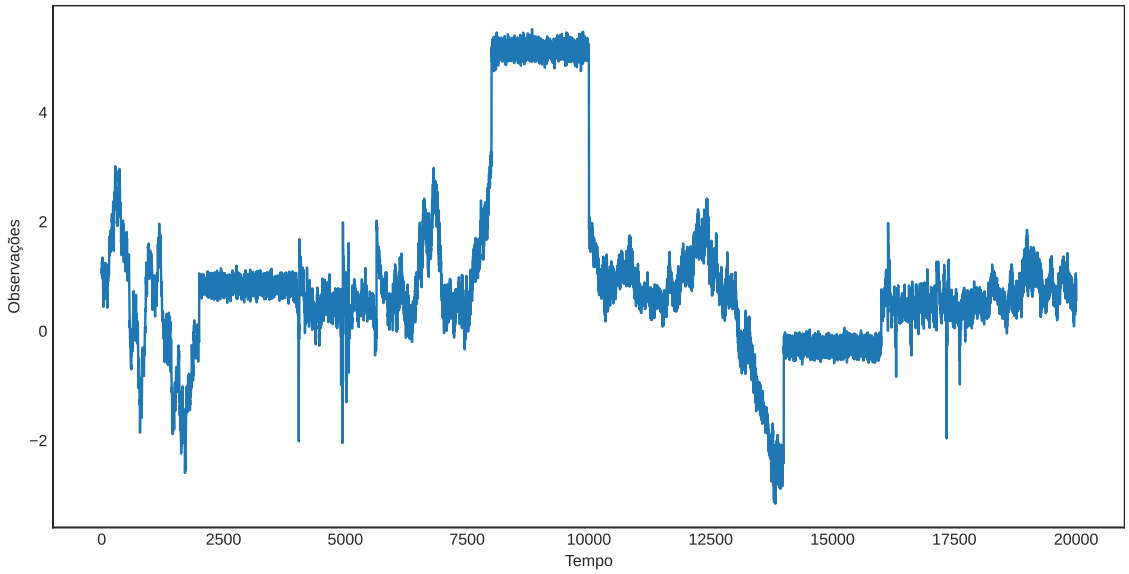
Tabela 3 – Parâmetros usados para criação das séries sintéticas.

		α	p
Linear	1	{0,42; 0,28; 0,005}	3
	2	{0,003; -0,005; 1,0}	3
	3	{0,018; 0,95; 0,032}	3
	4	{0,11; 0,32; -0,028; 0,038; 0,48}	5
	5	{0,032; 0,41; -0,24; -0,22; 1,0}	5
	6	{0,14; -0,29; 0,0025; 1,0}	4
		α	Modelo
Não linear	1	{0,55; 0,024; 0,41; 0,009}	NL 1
	2	{0,059; 0,086; 0,62; 0,21}	NL 2
	3	{0,47; 0,57; 0,14; -0,19}	NL 1
	4	{0,55; 0,024; 0,41; 0,009}	NL 1
	5	{0,059; 0,086; 0,62; 0,21}	NL 2
	6	{0,55; 1,0; 0,0028; -0,58}	NL 2
		β	s
Sazonal	1	{34; 32; 30; 28; 26; 24; 22; 24; 26; 28; 30; 32}	12
	2	{34; 26; 18; 10; 18; 26; 10}	7
	3	{34; 26; 18; 10; 18; 26}	6
	4	{34; 26; 18; 10; 2; -6; -14; -6; 2; 10; 18; 26}	12
	5	{34; 10; -14; 10}	4
	6	{38; 28; 18; 8; 0; -8; -18; -8; 0; 8; 18; 28}	12
		α, β	Modelo
Híbrida	1	{0,003; -0,005; 1,0}	AR
	2	{Últimas 3 observações}	Sazonal
	3	{0,059; 0,086; 0,62; 0,21}	NL 2
	4	{0,018; 0,95; 0,032}	AR
	5	{Últimas 3 observações}	Sazonal
	6	{0,55; 0,024; 0,41; 0,009}	NL 1

Fonte: Adaptado de Oliveira et al. (2017)

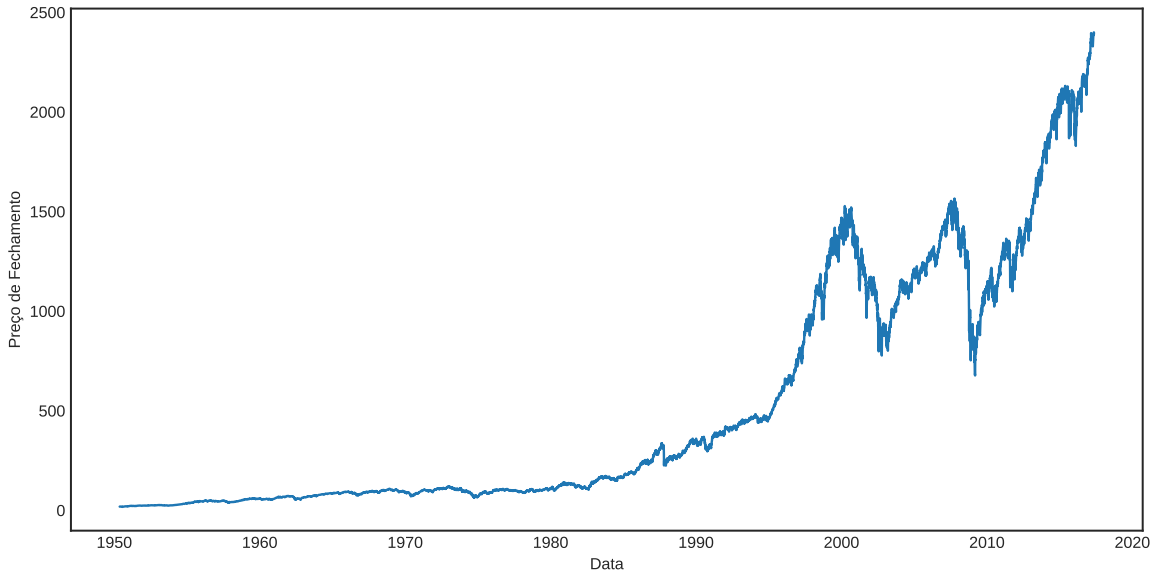
ele já sofreu muitas baixas em momentos de recessão econômica e em momentos mais otimistas ele costuma ter uma boa performance pois as pessoas investem mais em ações (BALCILAR; OZDEMIR; OZDEMIR, 2021). A série temporal foi obtida utilizando o Yahoo Finance (2021), com observações relacionadas ao valor de fechamento do índice entre os dias 15 de Maio de 1950 e 12 de Maio de 2017, como pode ser visto na Figura 18.

Figura 17 – Série híbrida criada sinteticamente.



Fonte: O autor

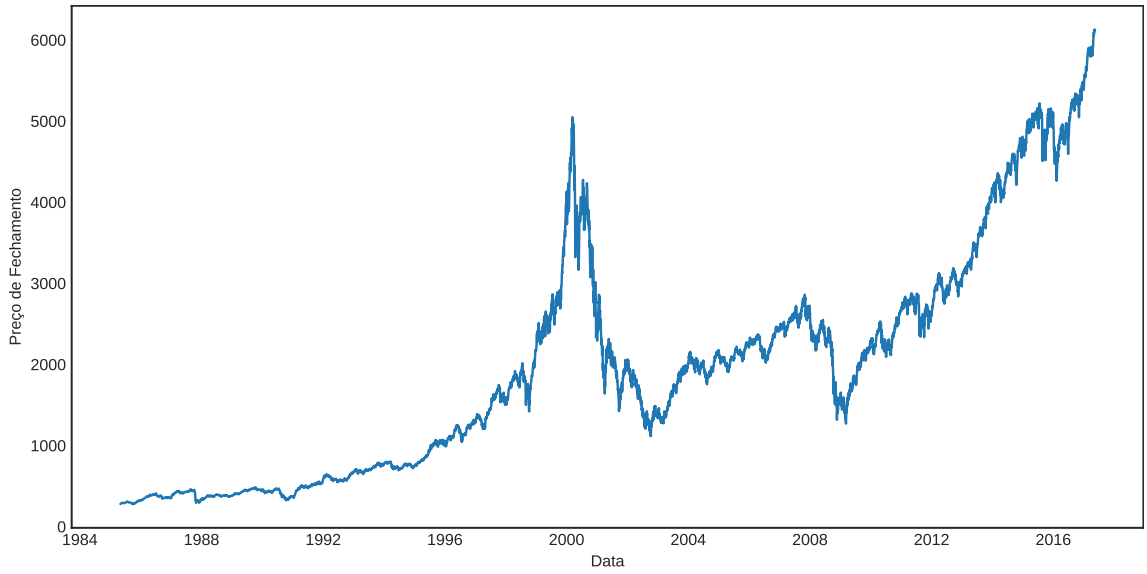
Figura 18 – Série S&P 500 de 15 de Maio de 1950 a 12 de Maio de 2017.



Fonte: O autor

Outra série real utilizada foi o índice Nasdaq, que é outro índice do mercado americano focado em empresas de alta tecnologia, telecomunicações e biotecnologia. Por sua composição, esse índice tem como características o alto crescimento e a alta sensibilidade a crises (ALIYEV; AJAYI; GASIM, 2020). A série temporal foi obtida utilizando o Yahoo Finance (2021), com observações relacionadas ao valor de fechamento do índice entre os dias 2 de Maio de 1985 e 12 de Maio de 2017, como pode ser visto na Figura 19.

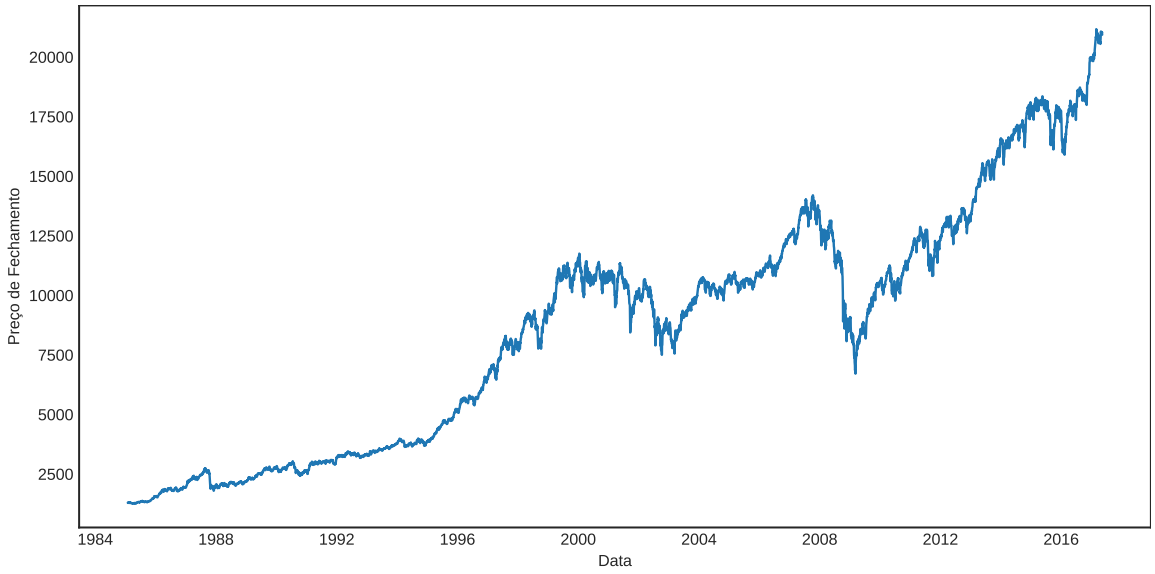
Figura 19 – Série Nasdaq de 2 de Maio de 1985 a 12 de Maio de 2017.



Fonte: O autor

A última série real utilizada foi o índice Dow Jones Industrial Average (DJIA), que é composto por ações de 30 das maiores empresas dos Estados Unidos ponderada por preço. Apesar de ter industrial no nome, esse índice evoluiu para incluir empresas de tecnologia e de serviços financeiros. As empresas que o compõem representam cerca de um quarto do valor de todo o mercado de ações americano (JABLANOVIC39, 2020). A série temporal foi obtida utilizando o Yahoo Finance (2021), com observações relacionadas ao valor de fechamento do índice entre os dias 29 de Janeiro de 1985 e 12 de Maio de 2017, como pode ser visto na Figura 20.

Figura 20 – Série Dow Jones de 29 de Janeiro de 1985 a 12 de Maio de 2017.



Fonte: O autor

5.4 RESULTADOS

Essa seção apresenta os resultados obtidos pelos métodos propostos: *Ensemble*; *BE drift*; e *BE always*. Além disso, será apresentado os resultados dos métodos disponíveis na literatura que são usados para comparação: o IDPSO ELM S, disponível em Oliveira et al. (2017); o ELM ECDD, disponível em Cavalcante e Oliveira (2015); o ELM DDM, disponível em Cavalcante e Oliveira (2015); e o ELM FEDD, que usa o mesmos método proposto por Cavalcante e Oliveira (2015) porém com o detector FEDD que está disponível em Cavalcante, Minku e Oliveira (2016).

5.4.1 Série linear sintética

A Tabela 4 apresenta as estatísticas descritivas para as execuções do experimento para todos os métodos, incluindo o método os métodos usados para comparação de resultados. Para a série linear observamos que o método que substitui o uso da melhor partícula para um *ensemble* teve um desempenho pior que o método IDPSO ELM S, o que pode indicar que somente essa mudança não é suficiente para melhorar o método. Mas os outros dois métodos usando o melhor *ensemble* conseguiram um desempenho superior, ficando em primeiro ou segundo lugar

nas estatísticas descritivas. Para essa série o *BE always* teve o melhor desempenho nas duas estatísticas descritivas, chamando a atenção o desvio padrão que ficou aproximadamente um terço do *BE drift*.

Tabela 4 – Estatísticas descritivas dos resultados do experimento para a série linear sintética.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0323^{[4]}$	$0.0409^{[4]}$
BE drift	$0.0157^{[2]}$	$0.0104^{[2]}$
BE always	$0.0154^{[1]}$	$0.0036^{[1]}$
IDPSO ELM S	$0.0186^{[3]}$	$0.0332^{[3]}$
ELM FEDD	$0.0479^{[6]}$	$0.0502^{[5]}$
ELM ECDD	$0.0464^{[5]}$	$0.0704^{[6]}$
ELM DDM	$0.0739^{[7]}$	$0.2008^{[7]}$

Fonte: O autor

5.4.2 Série não linear sintética

Para a série não linear sintética os resultados, apresentados na Figura 5, se diferenciam dos encontrados na série linear sintética, pois os métodos usados para comparação ficaram todos com as piores classificações nas estatísticas descritivas usadas. Os métodos com os melhores *ensemble* ficaram alternando o primeiro e segundo, chamando atenção mais uma vez para o desvio padrão dos métodos *BE drift* e *BE always* que ficaram menos da metade dos demais.

5.4.3 Série sazonal sintética

Na série sazonal sintética os resultados, mostrados na Figura 6, seguiram a mesma tendência da série não linear, pois os métodos usados para comparação ficaram com a pior classificação em todas as estatísticas descritivas. Os métodos com o melhor *ensemble* alternam entre o primeiro e o segundo lugar, demostrando estabilidade nos resultados.

Tabela 5 – Estatísticas descritivas dos resultados do experimento para a série não linear sintética.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0215^{[3]}$	$0.0023^{[3]}$
BE drift	$0.0180^{[2]}$	$0.0009^{[1]}$
BE always	$0.0178^{[1]}$	$0.0010^{[2]}$
IDPSO ELM S	$0.0222^{[4]}$	$0.0027^{[4]}$
ELM FEDD	$0.0375^{[6]}$	$0.0167^{[7]}$
ELM ECDD	$0.0379^{[7]}$	$0.0033^{[5]}$
ELM DDM	$0.0240^{[5]}$	$0.0044^{[6]}$

Fonte: O autor

Tabela 6 – Estatísticas descritivas dos resultados do experimento para a série sazonal sintética.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0443^{[3]}$	$0.0004^{[3]}$
BE drift	$0.0435^{[1]}$	$0.0004^{[2]}$
BE always	$0.0438^{[2]}$	$0.0003^{[1]}$
IDPSO ELM S	$0.0459^{[4]}$	$0.0005^{[4]}$
ELM FEDD	$0.0477^{[6]}$	$0.0009^{[6]}$
ELM ECDD	$0.0841^{[7]}$	$0.0045^{[7]}$
ELM DDM	$0.0474^{[5]}$	$0.0007^{[5]}$

Fonte: O autor

5.4.4 Série híbrida sintética

Nos resultados para a série híbrida sintética, mostrados na Tabela 7, o método ELM ECDD teve o menor desvio padrão, porém bem próximo dos métodos usando o melhor *ensemble*. O *BE drift* teve a melhor mediana, seguido pelo *BE always*. E o método *Ensemble* teve um dos piores resultados. Mesmo não ficando com o menor desvio padrão, os métodos com o melhor *ensemble*, ficaram com cerca de um terço do desvio padrão dos outros métodos.

Tabela 7 – Estatísticas descritivas dos resultados do experimento para a série híbrida sintética.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0704^{[5]}$	$0.0035^{[5]}$
BE drift	$0.0655^{[1]}$	$0.0011^{[2]}$
BE always	$0.0655^{[2]}$	$0.0012^{[3]}$
IDPSO ELM S	$0.0664^{[3]}$	$0.0034^{[4]}$
ELM FEDD	$0.1402^{[7]}$	$0.0184^{[7]}$
ELM ECDD	$0.0694^{[4]}$	$0.0010^{[1]}$
ELM DDM	$0.0731^{[6]}$	$0.0063^{[6]}$

Fonte: O autor

5.4.5 Série S&P500

O S&P 500 entra como primeira série financeira analisada e juntamente consigo traz uma maior imprevisibilidade, característica desse tipo de série. Mas, como pode ser observado nos resultados, disponíveis na Tabela 8, para essa série a ordem dos resultados não sofreu grandes modificações. Com o *BE always* ficando com as melhores estatísticas descritivas, com o *BE drift* muito próximo e o *Ensemble* tendo piores resultados, ficando entre os outros métodos usados para comparação. Nesse caso também vale a pena ressaltar que o desvio padrão dos métodos que usam o melhor *ensemble* ficaram com cerca da metade do valor do método IDPSO ELM S.

Tabela 8 – Estatísticas descritivas dos resultados do experimento para a série S&P 500 do mercado financeiro americano.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0123^{[6]}$	$0.0044^{[6]}$
BE drift	$0.0062^{[2]}$	$0.0017^{[3]}$
BE always	$0.0061^{[1]}$	$0.0013^{[1]}$
IDPSO ELM S	$0.0081^{[4]}$	$0.0029^{[5]}$
ELM FEDD	$0.0560^{[7]}$	$0.0469^{[7]}$
ELM ECDD	$0.0113^{[5]}$	$0.0015^{[2]}$
ELM DDM	$0.0078^{[3]}$	$0.0028^{[4]}$

Fonte: O autor

5.4.6 Série Nasdaq

Devido a características de maior volatilidade do índice Nasdaq, ela traz consigo uma maior detecção de *concept drift* e imprevisibilidade. Os resultados, apresentados na Tabela 9, mostram resultados um pouco diferentes dos acompanhados até então, onde os métodos com o melhor *ensemble* não tiveram os melhores desvio padrão, porém ainda assim mantiveram as melhores medianas. De forma geral o *BE always* ficou melhor que os demais, mas ao observar o desvio padrão podemos supor que esse foi um *outlier*.

Tabela 9 – Estatísticas descritivas dos resultados do experimento para a série Nasdaq do mercado financeiro americano.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0072^{[5]}$	$0.0032^{[5]}$
BE drift	$0.0045^{[2]}$	$0.0025^{[4]}$
BE always	$0.0044^{[1]}$	$0.0022^{[2]}$
IDPSO ELM S	$0.0061^{[3]}$	$0.0024^{[3]}$
ELM FEDD	$0.0080^{[6]}$	$0.0043^{[7]}$
ELM ECDD	$0.0082^{[7]}$	$0.0014^{[1]}$
ELM DDM	$0.0062^{[4]}$	$0.0034^{[6]}$

Fonte: O autor

5.4.7 Série Dow Jones

Como a composição do índice Dow Jones são empresas mais sólidas, é esperado que ele tenha o desempenho mais estável até que o índice S&P 500. Com essas características os resultados apresentados, na Tabela 10, mostram que o método *BE drift* teve desempenho melhor que os demais. Isso pode estar associado a possuir menos *concept drift* e o desempenho da melhor partícula ser, nesse caso, melhor que o do melhor *ensemble*. Ao observar o comportamento semelhante entre o IDPSO ELM S e o *Ensemble* essa possibilidade ganha força.

Tabela 10 – Estatísticas descritivas dos resultados do experimento para a série Dow Jones do mercado financeiro americano.

	MAE	
	Mediana	D. Padrão
Ensemble	$0.0055^{[6]}$	$0.0019^{[4]}$
BE drift	$0.0031^{[1]}$	$0.0011^{[2]}$
BE always	$0.0033^{[2]}$	$0.0008^{[1]}$
IDPSO ELM S	$0.0045^{[4]}$	$0.0026^{[5]}$
ELM FEDD	$0.0077^{[7]}$	$0.0074^{[7]}$
ELM ECDD	$0.0055^{[5]}$	$0.0016^{[3]}$
ELM DDM	$0.0035^{[3]}$	$0.0026^{[6]}$

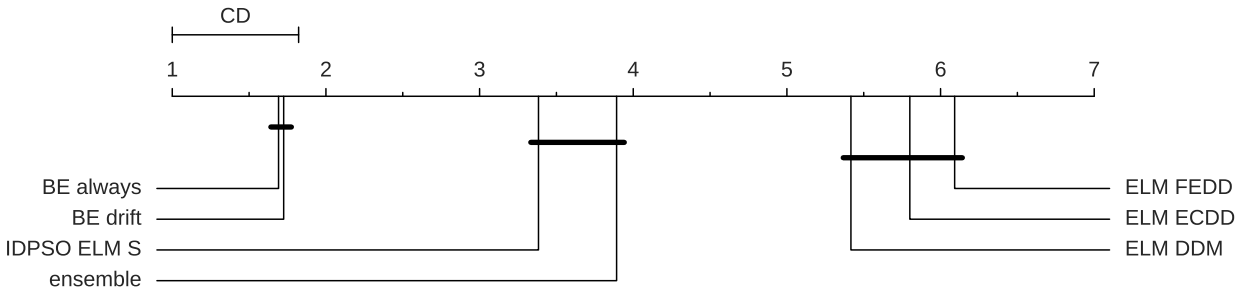
Fonte: O autor

5.4.8 Testes estatísticos

Com posse de todos os resultados foi aplicado os testes estatísticos de Friedman-Nemenyi para verificar se as diferenças encontradas possuem relevância estatística, com um nível de significância 5% para o teste de Nemenyi. O primeiro teste foi feito com os resultados de todas as séries sintéticas juntas, disponíveis na Figura 21. Segundo o ranking de Friedman, o *BE always* teve um desempenho melhor, seguido pelo *BE drift*, o IDPSO ELM S e por último o *Ensemble*. Porém, não houve diferença significativa entre o *BE always* e o *BE drift* e, apesar de um pouco mais distante, não houve diferença significativa entre o IDPSO ELM S e o *Ensemble*. Os outros métodos da literatura tiveram os piores desempenhos, mas sem diferença estatística entre eles. Para ter diferença significativa estatisticamente pelo teste de Nemenyi, a distância entre os métodos precisa ser maior que CD, e como mostrado na Figura 21 houve diferença entre os métodos que usaram o melhor *ensemble* e os métodos usados como referência, e pelo ranking eles tiveram melhores desempenhos.

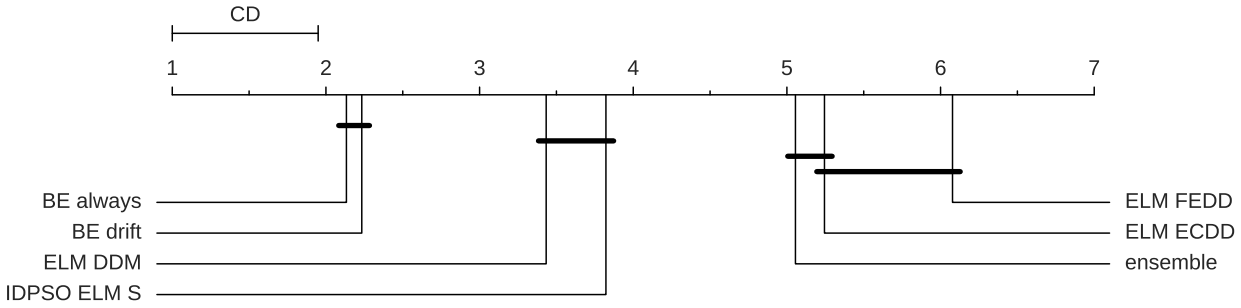
A Figura 22 contém o resultado dos testes de Friedman-Nemenyi para as séries financeiras, com o mesmo nível de significância 5% para o teste de Nemenyi. Para esse caso o ranking de Friedman teve o valor um pouco diferentes, onde o *BE drift* ficou em primeiro, o *BE always* em segundo, seguido pelo ELM DDM e o IDPSO ELM S. O método *Ensemble* ficou em quinto, sem diferença estatística para os demais. Segundo o teste de Nemenyi, não houve diferença significativa entre os métodos que usaram o melhor *ensemble*.

Figura 21 – Testes de Friedman-Nemenyi para os métodos em todas as séries sintéticas.



Fonte: O autor

Figura 22 – Testes de Friedman-Nemenyi para os métodos em todas as séries reais financeiras.



Fonte: O autor

5.5 DISCUSSÕES

A seção de discussão é apresentada de acordo com as questões de pesquisa, então para cada uma das perguntas feitas terá uma resposta baseada nos resultados mostrados na seção anterior.

Questão 1: Combinar os melhores modelos do PSO oferece vantagens ao uso apenas do melhor? Sabendo que as partículas de um PSO tendem a convergir para uma mesma solução, combinar modelos de um PSO vai realmente reduzir a quantidade de variações nas previsões e com isso diminuir o erro?

O uso de uma combinação das melhores partículas demonstrou uma melhora significativa em relação ao uso de apenas uma. Porém, o uso de um *ensemble* com todas as partículas, sem critério para seleção de quais usar, demonstrou não trazer melhoras no resultado final. Para que

as partículas do PSO ficassem com maior diferença a quantidade de ciclos de treinamento foi reduzida, com isso não ocorreu o problema de usar um *ensemble* com partículas semelhantes ao extremo.

Questão 2: Tendo em vista que ao acontecer o *concept drift* teoricamente os modelos ficam obsoletos, é possível usar modelos treinados em conceitos passados para acelerar na adaptação do novo conceito?

Em ambos os métodos, que usaram a estratégias de reaproveitamento das partículas depois que o *concept drift* é identificado, conseguiram melhores resultados e de forma significativa. Mas entre eles os resultados não tiveram grandes modificações, levando a pensar que a melhora do desempenho veio da estratégia de reaproveitamento. Considerando também o método que usa o *ensemble* com todas as partículas, é possível dizer que apenas o uso de um *ensemble* quando o *concept drift* é identificado não é suficiente e sim escolher quais são as melhores para o novo conceito.

6 CONCLUSÃO

Este trabalho apresenta três métodos para previsão de séries temporais com *concept drift*. Os métodos usam o modelo de aprendizado de máquina ELM para realizar previsões, além de fazer detecção e tratamento de *concept drift*. Para otimização do treinamento é usado o PSO, e as partículas treinadas são usadas para previsão e sensor de *concept drift*. O primeiro método usa um *ensemble* com todas as partículas, o segundo usa uma combinação das melhores partículas quando o *concept drift* é detectado até a adaptação ou geração de um novo modelo e o terceiro usa uma combinação das partículas antes e atualiza depois da detecção, enquanto os dados são coletados para o novo treinamento.

Durante o experimento foi utilizado métodos disponíveis na literatura para comparação dos resultados e os métodos propostos que se adaptam usando uma combinação das melhores partículas, após a detecção de mudança de conceito, conseguiram resultados melhores do que os seus pares, mostrando que o período de adaptação do método no novo conceito é importante no resultado final da previsão e que o treinamento realizado anteriormente pode ajudar a diminuir, significativamente, esse erro.

Os experimentos foram realizados em sete séries, sendo elas quatro sintéticas e três reais, de índices do mercado financeiro, onde os métodos propostos que realizam adaptações após a detecção do *concept drift* tiveram resultados estatisticamente melhores em ambos os casos. Contudo não tiveram resultados estatisticamente diferentes entre si, o que pode levar a deduzir que o fator determinante para a melhoria no método foi a combinação feita durante a adaptação do *concept drift*.

6.1 TRABALHOS FUTUROS

Como trabalho futuro esse mesmo método pode ser feito usando outro modelo de aprendizado de máquina no lugar do ELM, mesmo o ELM sendo um modelo com boa capacidade de generalização, alguns outros como LSTM ou GRU podem ser utilizados para substituir, principalmente, os modelos que realizam as previsões. Porém como sensor um modelo mais simples pode ser mais eficaz. Além disso, outros métodos de detecção de *concept drift* podem ser testados ou até uma combinação de métodos. É esperado que caso o método detecte mais *concept drift* o desempenho não mude muito, porém caso passe a detectar menos o método

pode ser afetado negativamente.

Uma outra variação no experimento que pode ser feita futuramente é usar esse método para realizar previsões em uma frequência maior. Todos os experimentos com as séries financeiras foram feitos com valores diários e quanto maior intervalo, maior a probabilidade de outras informações, além do histórico contido na série, poder influenciar o preço futuro. Com isso, o uso de intervalos de quinze minutos, trinta minutos ou uma hora pode ajudar o método a realizar melhores previsões.

REFERÊNCIAS

ADAMS, J. N.; ZELST, S. J. van; QUACK, L.; HAUSMANN, K.; AALST, W. M. van der; ROSE, T. A framework for explainable concept drift detection in process mining. *arXiv preprint arXiv:2105.13155*, 2021.

ALIYEV, F.; AJAYI, R.; GASIM, N. Modelling asymmetric market volatility with univariate garch models: Evidence from nasdaq-100. *The Journal of Economic Asymmetries*, Elsevier, v. 22, p. e00167, 2020.

ALSHALALI, T.; JOSYULA, D. Evaluating extreme learning machine models in the presence of concept drift in streaming data. In: *Proceedings of the 2020 the 4th International Conference on Information System and Data Mining*. [S.l.: s.n.], 2020. p. 107–112.

BAI, B.; GUO, Z.; ZHOU, C.; ZHANG, W.; ZHANG, J. Application of adaptive reliability importance sampling-based extended domain pso on single mode failure in reliability engineering. *Information Sciences*, Elsevier, v. 546, p. 42–59, 2021.

BALCILAR, M.; OZDEMIR, Z. A.; OZDEMIR, H. Dynamic return and volatility spillovers among s&p 500, crude oil, and gold. *International Journal of Finance & Economics*, Wiley Online Library, v. 26, n. 1, p. 153–170, 2021.

BALLABIO, D.; GRISONI, F.; TODESCHINI, R. Multivariate comparison of classification performance measures. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 174, p. 33–44, 2018.

BARROS, R. S. M. de; SANTOS, S. G. T. de C.; JÚNIOR, P. M. G. A boosting-like online learning ensemble. In: IEEE. *2016 international joint conference on neural networks (IJCNN)*. [S.l.], 2016. p. 1871–1878.

BOX, G. E.; JENKINS, G. M.; REINSEL, G. C. *Time series analysis: forecasting and control*. [S.l.]: Prentice Hall PTR, 1994.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

CASTELLANI, A.; SCHMITT, S.; HAMMER, B. Task-sensitive concept drift detector with metric learning. *arXiv e-prints*, p. arXiv–2108, 2021.

CAVALCANTE, R. C.; MINKU, L. L.; OLIVEIRA, A. L. Fedd: Feature extraction for explicit concept drift detection in time series. In: IEEE. *2016 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2016. p. 740–747.

CAVALCANTE, R. C.; OLIVEIRA, A. L. An approach to handle concept drift in financial time series based on extreme learning machines and explicit drift detection. In: IEEE. *2015 international joint conference on neural networks (IJCNN)*. [S.l.], 2015. p. 1–8.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, v. 16, p. 321–357, 2002.

CHEN, Y.; CHAI, Z.; CHENG, Y.; RANGWALA, H. Asynchronous federated learning for sensor data with concept drift. *arXiv preprint arXiv:2109.00151*, 2021.

CHEN, Y.; NING, Y.; SLAWSKI, M.; RANGWALA, H. Asynchronous online federated learning for edge devices with non-iid data. In: IEEE. *2020 IEEE International Conference on Big Data (Big Data)*. [S.l.], 2020. p. 15–24.

CHICCO, D.; WARRENS, M. J.; JURMAN, G. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. *PeerJ Computer Science*, PeerJ Inc., v. 7, p. e623, 2021.

CHO, K.; MERRIËNBOER, B. V.; BAHDANAU, D.; BENGIO, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

COWPERTWAIT, P. S.; ANDREW, V. *Introductory time series with R Springer Science & Business Media*. [S.l.]: LLC, 2009.

DITZLER, G.; POLIKAR, R. Hellinger distance based drift detection for nonstationary environments. In: IEEE. *2011 IEEE symposium on computational intelligence in dynamic and uncertain environments (CIDUE)*. [S.l.], 2011. p. 41–48.

ELIADES, C.; PAPADOPOULOS, H. Using inductive conformal martingales for addressing concept drift in data stream classification. In: PMLR. *Conformal and Probabilistic Prediction and Applications*. [S.l.], 2021. p. 171–190.

FREUND, Y. Boosting a weak learning algorithm by majority. *Information and computation*, Elsevier, v. 121, n. 2, p. 256–285, 1995.

GAMA, J.; MEDAS, P.; CASTILLO, G.; RODRIGUES, P. Learning with drift detection. In: SPRINGER. *Brazilian symposium on artificial intelligence*. [S.l.], 2004. p. 286–295.

GOODRICH, M. T.; TAMASSIA, R.; GOLDWASSER, M. H. *Data structures and algorithms in Python*. [S.l.]: John Wiley & Sons Ltd, 2013.

HE, Y.; WANG, X.; HE, H.; ZHAI, J.; WANG, B. Moving average based index for judging the peak of the covid-19 epidemic. *International Journal of Environmental Research and Public Health*, Multidisciplinary Digital Publishing Institute, v. 17, n. 15, p. 5288, 2020.

HIDALGO, J. I.; SANTOS, S. G.; BARROS, R. S. Dynamically adjusting diversity in ensembles for the classification of data streams with concept drift. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, ACM New York, NY, v. 16, n. 2, p. 1–20, 2021.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

IWASHITA, A. S.; PAPA, J. P. An overview on concept drift learning. *Ieee Access*, IEEE, v. 7, p. 1532–1547, 2018.

JABLANOVIC39, V. The dow jones industrial average (djia) stock market index and the chaotic growth model. *Recent Advances in Information Technology, Tourism, Economics, Management and Agriculture*, p. 113, 2020.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: IEEE. *Proceedings of ICNN'95-international conference on neural networks*. [S.l.], 1995. v. 4, p. 1942–1948.

KIRCHGÄSSNER, G.; WOLTERS, J.; HASSLER, U. *Introduction to modern time series analysis*. [S.l.]: Springer Science & Business Media, 2012.

KOURENTZES, N.; BARROW, D. K.; CRONE, S. F. Neural network ensemble operators for time series forecasting. *Expert Systems with Applications*, Elsevier, v. 41, n. 9, p. 4235–4244, 2014.

KWOK, S. W.; CARTER, C. Multiple decision trees. In: *Machine intelligence and pattern recognition*. [S.l.]: Elsevier, 1990. v. 9, p. 327–335.

LAHMIRI, S. A variational mode decompoisition approach for analysis and forecasting of economic and financial time series. *Expert Systems with Applications*, Elsevier, v. 55, p. 268–273, 2016.

LEE, S.; PARK, S. H. Concept drift modeling for robust autonomous vehicle control systems in time-varying traffic environments. *Expert Systems with Applications*, Elsevier, v. 190, p. 116206, 2022.

LI, T.; SAHU, A. K.; ZAHEER, M.; SANJABI, M.; TALWALKAR, A.; SMITH, V. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, v. 2, p. 429–450, 2020.

LIM, B.; ZOHREN, S. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, The Royal Society Publishing, v. 379, n. 2194, p. 20200209, 2021.

LIN, C.-C.; DENG, D.-J.; KUO, C.-H.; CHEN, L. Concept drift detection and adaption in big imbalance industrial iot data using an ensemble learning method of offline classifiers. *IEEE Access*, IEEE, v. 7, p. 56198–56207, 2019.

LIU, Z.; LOO, C. K.; PASUPA, K.; SEERA, M. Meta-cognitive recurrent kernel online sequential extreme learning machine with kernel adaptive filter for concept drift handling. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 88, p. 103327, 2020.

MCMAHAN, B.; MOORE, E.; RAMAGE, D.; HAMPSON, S.; ARCAS, B. A. y. Communication-efficient learning of deep networks from decentralized data. In: PMLR. *Artificial intelligence and statistics*. [S.l.], 2017. p. 1273–1282.

MINKU, L. L.; YAO, X. Ddd: A new ensemble approach for dealing with concept drift. *IEEE transactions on knowledge and data engineering*, IEEE, v. 24, n. 4, p. 619–633, 2011.

MORETTIN, P. A.; TOLOI, C. M. Time series analysis. *Blucher. Portuguese*, 2006.

NIAZADEH, R.; GOLREZAEI, N.; WANG, J. R.; SUSAN, F.; BADANIDIYURU, A. Online learning via offline greedy algorithms: Applications in market design and optimization. In: *Proceedings of the 22nd ACM Conference on Economics and Computation*. [S.l.: s.n.], 2021. p. 737–738.

OLIVEIRA, G. H.; CAVALCANTE, R. C.; CABRAL, G. G.; MINKU, L. L.; OLIVEIRA, A. L. Time series forecasting in the presence of concept drift: A pso-based approach. In: IEEE. *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. [S.l.], 2017. p. 239–246.

OZA, N. C.; RUSSELL, S. Experimental comparisons of online and batch versions of bagging and boosting. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.: s.n.], 2001. p. 359–364.

PAN, Z.; MENG, Z.; CHEN, Z.; GAO, W.; SHI, Y. A two-stage method based on extreme learning machine for predicting the remaining useful life of rolling-element bearings. *Mechanical Systems and Signal Processing*, Elsevier, v. 144, p. 106899, 2020.

PARK, J.-M.; KIM, J.-H. Online recurrent extreme learning machine and its application to time-series prediction. In: IEEE. *2017 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2017. p. 1983–1990.

PRADHAN, A.; BISOY, S. K.; DAS, A. A survey on pso based meta-heuristic scheduling mechanism in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, 2021.

RÁCZ, A.; BAJUSZ, D.; HÉBERGER, K. Effect of dataset size and train/test split ratios in qsar/qspr multiclass classification. *Molecules*, Multidisciplinary Digital Publishing Institute, v. 26, n. 4, p. 1111, 2021.

REIS, D. M. dos; FLACH, P.; MATWIN, S.; BATISTA, G. Fast unsupervised online drift detection using incremental kolmogorov-smirnov test. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2016. p. 1545–1554.

ROSS, G. J.; ADAMS, N. M.; TASOULIS, D. K.; HAND, D. J. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, Elsevier, v. 33, n. 2, p. 191–198, 2012.

SANTOS, S. G. T. d. C.; BARROS, R. S. M. de. Online adaboost-based methods for multiclass problems. *Artificial Intelligence Review*, Springer, v. 53, n. 2, p. 1293–1322, 2020.

SATO, D. M. V.; FREITAS, S. C. D.; BARDDAL, J. P.; SCALABRIN, E. E. A survey on concept drift in process mining. *ACM Computing Surveys (CSUR)*, ACM New York, NY, v. 54, n. 9, p. 1–38, 2021.

SCARDAPANE, S.; COMMINIELLO, D.; SCARPINITI, M.; UNCINI, A. Online sequential extreme learning machine with kernels. *IEEE transactions on neural networks and learning systems*, IEEE, v. 26, n. 9, p. 2214–2220, 2014.

SHELDON, M. R.; FILLYAW, M. J.; THOMPSON, W. D. The use and interpretation of the friedman test in the analysis of ordinal-scale data in repeated measures designs. *Physiotherapy Research International*, Wiley Online Library, v. 1, n. 4, p. 221–228, 1996.

TIAN, F. How covid-19 impacts the us economy and predictions about the future. In: EDP SCIENCES. *E3S Web of Conferences*. [S.l.], 2020. v. 218, p. 04007.

VOLKHONSKIY, D.; BURNAEV, E.; NOURETDINOV, I.; GAMMERMAN, A.; VOVK, V. Inductive conformal martingales for change-point detection. In: PMLR. *Conformal and Probabilistic Prediction and Applications*. [S.l.], 2017. p. 132–153.

WANG, H.; ABRAHAM, Z. Concept drift detection for streaming data. In: IEEE. *2015 international joint conference on neural networks (IJCNN)*. [S.l.], 2015. p. 1–9.

WITTEN, I. H.; FRANK, E. Data mining: practical machine learning tools and techniques. *Acm Sigmod Record*, ACM New York, NY, USA, v. 3, 2011.

WOLPERT, D. H. Stacked generalization. *Neural networks*, Elsevier, v. 5, n. 2, p. 241–259, 1992.

Yahoo Finance. *Stock Market Live, Quotes, Business e Finance News*. 2021. Disponível em: <https://finance.yahoo.com/>. Acesso em: 10 out. 2021.

ZHANG, Z.; LIU, F.; ZHOU, Z.; HE, Y.; FANG, H. Roughness measurement of leaf surface based on shape from focus. *Plant Methods*, Springer, v. 17, n. 1, p. 1–11, 2021.

ZHOU, Y.; ZHOU, N.; GONG, L.; JIANG, M. Prediction of photovoltaic power output based on similar day analysis, genetic algorithm and extreme learning machine. *Energy*, Elsevier, v. 204, p. 117894, 2020.

ANEXO A – PSEUDOCÓDIGOS

Algoritmo 1: Pseudocódigo do algoritmo guloso.

```
Entradas:
   Clist: conjunto com todas as particulas
   Dtest: dados de teste

Cmelhor = {} # cria o conjunto de particulas vazio
MAE_atual = +∞ # coloca temporariamente o pior desempenho possivel

faca:
    Cbusca = Clist - Cmelhor # seleciona as particulas que nao estao nas melhores
    MAE_mudanca = MAE_atual
    nova_particula = ∅
    para cada particula em Cbusca:
        MAE_teste = testa_ensemble(Cmelhor ∪ {particula}, Dtest)
        se M_teste < MAE_mudanca:
            nova_particula = particula
            MAE_mudanca = MAE_teste

    se existir nova_particula:
        Cmelhor = Cmelhor ∪ {nova_particula}
        MAE_atual = MAE_mudanca
ate nao existir nova_particula

retorne Pmelhor
```

Algoritmo 2: Pseudocódigo do método *Ensemble*

```
Entradas:
    N_train: Tamanho da amostra necessaria para treinamento
    N_p: Numero de particulas
    N_sensor: Numero de sensores
    N_neurons: Numero de neuronios
    N_it: Numero de iteracao de treinamento para o PSO
    P_train: Proporcao de treinamento

Dtreino = ler e armazena primeiras N_train entradas
PSO = realiza_treinamento(Dtreino, P_train, N_it, N_neurons, N_p)
Cparticulas = construa_ensemble(PSO) # usa todas as particulas
Dserie = Dtreino
Dprevista = {}
aconteceu_mudanca = falso

para toda a nova_observacao faca:
    previsao = realiza_previsao(Cparticulas, Dserie)

    Dserie = Dserie ∪ {nova_observacao}
    Dprevista = Dprevista ∪ {previsao}

    se nao aconteceu_mudanca:
        aconteceu_mudanca = detector_concept_drift(Dprevista, Dserie, PSO, N_sensor)
        Dtreino = {}

    se aconteceu_mudanca:
        Dtreino = Dtreino ∪ {nova_observacao} # armazena para o novo treinamento

        se tamanho(Dtreino) == N_train:
            PSO = realiza_treinamento(Dtreino, P_train, N_it, N_neurons, N_p)
            Cparticulas = construa_ensemble(PSO) # usa todas as particulas
```

Algoritmo 3: Pseudocódigo do método *BE-drift*.

```
Entradas:
    N_train: Tamanho da amostra necessaria para treinamento
    N_p: Numero de particulas
    N_sensor: Numero de sensores
    N_i: Intervalo de reconstrucao do ensemble
    N_neurons: Numero de neuronios
    N_lags: Numero de lags
    N_it: Numero de iteracao de treinamento para o PSO
    P_train: Proporcao de treinamento

Dtreino = ler e armazena primeiras N_min entradas
PSO = realiza_treinamento(Dtreino, P_train, N_it, N_neurons, N_p)
Cparticulas = melhor_particula(PSO) # somente a melhor particula
Dserie = Dtreino
Dprevista = {}
aconteceu_mudanca = falso

para toda a nova_observacao faca:
    previsao = realiza_previsao(Cparticulas, Dserie)

    Dserie = Dserie ∪ {nova_observacao}
    Dprevista = Dprevista ∪ {previsao}

    se nao aconteceu_mudanca:
        aconteceu_mudanca = detector_concept_drift(Dprevista, Dserie, PSO, N_sensor)
        Dtreino = {}

    se aconteceu_mudanca:
        Dtreino = Dtreino ∪ {nova_observacao} # armazena para o novo treinamento

        se tamanho(Dtreino) for um multiplo de N_i:
            Cparticulas = melhor_combinacao(PSO, Dtreino) # algoritmo guloso

        se tamanho(Dtreino) == N_min:
            PSO = realiza_treinamento(Dtreino, P_train, N_it, N_neurons, N_p)
            Cparticulas = melhor_combinacao(PSO, Dtreino, P_train) # algoritmo
    guloso
```

Algoritmo 4: Pseudocódigo do método *BE-always*.

```
Entradas:
    N_train: Tamanho da amostra necessaria para treinamento
    N_p: Numero de particulas
    N_sensor: Numero de sensores
    N_i: Intervalo de reconstrucao do ensemble
    N_neurons: Numero de neuronios
    N_lags: Numero de lags
    N_it: Numero de iteracao de treinamento para o PSO
    P_train: Proporcao de treinamento

Dtreino = ler e armazena primeiras N_min entradas
PSO = realiza_treinamento(Dtreino, P_train, N_it, N_neurons, N_p)
Cparticulas = melhor_combinacao(PSO, Dtreino, P_train) # algoritmo guloso
Dserie = Dtreino
Dprevista = {}
aconteceu_mudanca = falso

para toda a nova_observacao faca:
    previsao = realiza_previsao(Cparticulas, Dserie)

    Dserie = Dserie ∪ {nova_observacao}
    Dprevista = Dprevista ∪ {previsao}

    se nao aconteceu_mudanca:
        aconteceu_mudanca = detector_concept_drift(Dprevista, Dserie, PSO, N_sensor)
        Dtreino = {}

    se aconteceu_mudanca:
        Dtreino = Dtreino ∪ {nova_observacao} # armazena para o novo treinamento

        se tamanho(Dtreino) for um multiplo de N_i:
            Cparticulas = melhor_combinacao(PSO, Dtreino) # algoritmo guloso

        se tamanho(Dtreino) == N_min:
            PSO = realiza_treinamento(Dtreino, P_train, N_it, N_neurons, N_p)
            Cparticulas = melhor_combinacao(PSO, Dtreino, P_train) # algoritmo
    guloso
```