Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

André Luís Nunes

Um estudo investigativo de algortimos de regressão para *data streams*

São Leopoldo, 2017

André Luís Nunes

UM ESTUDO INVESTIGATIVO DE ALGORITMOS DE REGRESSÃO PARA *DATA STREAMS*

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Dr. João Francisco Valiati

São Leopoldo
2017

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

N972e Nunes, André Luís.
Um estudo investigativo de algoritmos de regressão para *data streams* / André Luís Nunes. – 2017.
92 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, 2017.
"Orientador: Dr. João Francisco Valiati."

1. Mineração de data stream. 2. Regressão. 3. *Concept drift*. I. Título.

CDU 004

Bibliotecário responsável: Flávio Nunes – CRB 10/1298

André Luís Nunes

Um estudo investigativo de algoritmos de regressão para *data streams*

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 28 de março de 2017.

BANCA EXAMINADORA

Dr. Leandro Krug Wives – UFRGS

Dr. Sandro José Rigo – UNISINOS

Prof. Dr.João Francisco Valiati (Orientador)

Visto e permitida a impressão
São Leopoldo,

Prof. Dr. Sandro José Rigo
Coordenador PPG em Computação Aplicada

AGRADECIMENTOS

Para a realização deste trabalho alguns aspectos foram fundamentais. Principalmente o convívio com determinadas pessoas, algumas de forma direta outras de diferente modo. Com isso, presto meus sinceros agradecimentos:

Ao professor Dr. João Francisco Valiati, orientador deste trabalho, pelo conhecimento repassado, paciência, dedicação e orientações, que foram prestadas de forma sem igual, sendo fundamental para o desenvolvimento e crescimento acadêmico e profissional.

A minha família, em especial minha esposa Hananda Farias, com quem pude contar com apoio incondicional, companheirismo e motivação, pela compreensão quando foi preciso me ausentar de inúmeros eventos familiares e sociais neste período.

A minha filha, Mariana Farias Nunes, pela compreensão e principalmente pelo carinho em momentos de dificuldades, por trazer felicidade e descontração em diferentes situações, principalmente pelas preocupações demonstradas quando ao acordar ainda me encontrava estudando sem ter dormido.

Aos meus pais, essenciais pela formação pessoal e incentivo em todos os sentidos, por todo esforço desprendido para proporcionar condições que possibilitaram ambientes favoráveis na busca pelo conhecimento, formação e aprendizado.

Aos demais professores do corpo docente do Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da UNISINOS que contribuíram para aprendizado, tanto através de aulas quanto respondendo questões pontuais sempre que foram solicitados.

A empresa CWI Software pelo suporte e apoio oferecido durante esse período de formação, onde foi oportunizado tempo e espaço físico para pesquisas.

RESUMO

A explosão no volume de dados e a sua velocidade de expansão tornam as tarefas de descoberta do conhecimento e a análise de dados desafiantes, ainda mais quando consideradas bases não-estacionárias. Embora a predição de valores futuros exerça papel fundamental em áreas como: o clima, problemas de roteamentos e economia, entre outros, a classificação ainda parece ser a tarefa mais explorada. Recentemente, alguns algoritmos voltados à regressão de valores foram lançados, como por exemplo: FIMT-DD, AMRules, IBLStreams e SFNRegressor, entretanto seus estudos investigativos exploraram mais aspectos de inovação e análise do erro de predição, do que explorar suas capacidades mediante critérios apontados como fundamentais para *data stream*, como tempo de execução e memória. Dessa forma, o objetivo deste trabalho é apresentar um estudo investigativo sobre estes algoritmos que tratam regressão, considerando ambientes dinâmicos, utilizando bases de dados massivas, além de explorar a capacidade de adaptação dos algoritmos com a presença de *concept drift*. Para isto três bases de dados foram analisadas e estendidas para explorar os principais critérios de avaliação adotados, sendo realizada uma ampla experimentação que produziu uma comparação dos resultados obtidos frente aos algoritmos escolhidos, possibilitando gerar indicativos do comportamento de cada um mediante os diferentes cenários a que foram expostos. Assim, como principais contribuições deste trabalho são destacadas: a avaliação de critérios fundamentais: memória, tempo de execução e poder de generalização, relacionados a regressão para data stream; produção de uma análise crítica dos algoritmos investigados; e a possibilidade de reprodução e extensão dos estudos realizados pela disponibilização das parametrizações empregadas.

Palavras-Chave: Mineração de *data stream*, regressão, *concept drift*.

ABSTRACT

The explosion of data volume and its expansion speed make tasks of finding knowledge and analyzing data challenging, even more so when non-stationary bases are considered. Although the future values prediction plays a fundamental role in areas such as climate, routing problems and economics, among others, classification seems to be still the most exploited task. Recently, some value-regression algorithms have been launched, for example: FIMT-DD, AMRules, IBLStreams and SFNRegressor; however, their investigative studies have explored more aspects of innovation and analysis of error prediction than exploring their capabilities through criteria that are considered fundamental to data stream, such as elapsed time and memory. In this way, the objective of this work is to present an investigative study about these algorithms that treat regression considering dynamic environments, using massive databases, and also explore the algorithm's adaptability capacity with the presence of concept drift. In order to do this, three databases were analyzed and extended to explore the main evaluation criteria adopted. A wide experiment was carried out, which produced a comparison of the results obtained with the chosen algorithms, allowing to generate behavior indication of each one through the different scenarios to which were exposed. Thus, the main contributions of this work are: evaluation of fundamental criteria: memory, execution time and power of generalization, related to regression to data stream; production of a critical analysis of the algorithms investigated; and the possibility of reproducing and extending the studies carried out by making available the parametrizations applyed.

Keywords: data stream mining, regression, concept drift.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Rev.	Revisão
Bib.	Bibliográfica
Trab.	Trabalho(s)

LISTA DE SIGLAS

ADWIN	*ADaptative sliding WINdow*
CART	*Classification and Regression Trees*
DSRRA	*Data Stream Regression Results Analysis*
EMA	*Exponential Moving Average*
EWMA	*Exponential Weight Moving Average*
FIMT-DD	*Fast Incremental Model Tree – Drift Detection*
IBL	*Instance Based Learner*
KDD	*Knowledge Discovery in Databases*
MAE	*Mean Absolute Error*
MOA	*Massive Online Analysis*
MSE	*Mean Squared Error*
PHT	*Page-Hinkley Test*
PIPCA	Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
RMSE	*Root Mean Squared Error*
RRSE	*Root Relative Squared Error*
SFN	*Social Free Network*
VFDT	*Very Fast Decision Tree*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos o termo *big data* tem sido utilizado para descrever o crescimento exponencial de dados estruturados e não estruturados. Fatores como o aumento do volume e detalhes dos dados gerados e capturados pelas organizações, assim como o crescimento das redes sociais e a internet das coisas[1] têm contribuído para este fenômeno nas duas últimas décadas, segundo Lossio-Ventura e Alatrista-Salas (2015). Enquanto Gantz et al. (2012) projetam que de 2005 até 2020 haverá um aumento de trezentas vezes no volume de dados, passando de 130 exabytes para 40.000 exabytes. Porém, a expressão *big data* não está relacionada apenas com o tamanho dos dados, mas sim com três características principais: (a) o volume de dados, (b) a variabilidade dos tipos de dados, como: texto, vídeos, imagens, dados de sensores, etc., e (c) a velocidade que os dados são gerados, capturados e processados (LANEY, 2001).

Este cenário foi favorecido pela evolução das tecnologias de armazenamento de dados, dos dispositivos móveis, como os smartphones, e as melhorias na infraestrutura de comunicações, assim como a redução de custos dos mesmos. Isso tem permitido que diferentes áreas de conhecimento, como: ciência da computação, medicina, educação, entretenimento, saúde, além das organizações privadas e órgãos públicos, produzam cada vez mais dados. Este ambiente tem proporcionado muitos desafios para a área de análise de dados, em especial na Descoberta do Conhecimento em Bases de Dados (*Knowledge Discovery in Databases* - KDD), que tem como objetivo a extração de conhecimento e padrões em conjuntos de dados (FAYYAD et al., 1996).

No KDD a mineração de dados é um processo que permite explorar um conjunto de dados, com o objetivo de estabelecer associações, relações e padrões que são difíceis de visualizar manualmente. Algumas vezes possibilitando a transformação de dados brutos em informações valiosas. Para tanto, envolve a integração de técnicas multidisciplinares como, algoritmos de aprendizado de máquina, reconhecimento de padrões, análises estatísticas, visualização de dados entre outras (HAN e KAMBER, 2006). Como resultado desta etapa é apresentado um modelo, na forma de regras, hipóteses ou árvores de decisão.

Entretanto, os métodos tradicionais de mineração de dados enfrentam três principais limitações computacionais: recursos de memória, tempo de execução e tamanho das amostras utilizadas. Em geral, nas aplicações de aprendizado de máquina para técnicas de mineração de dados tradicionais são utilizadas amostragem pequenas, com menos de 10.000 instâncias (DOMINGOS e HULTEN, 2000). Em um contexto onde é previsto um crescimento enorme, com diferentes tipos de dados sendo gerados, tanto estruturados, como valores de atributos de um banco de dados, quanto não estruturados, como, por exemplo: imagens, vídeos, áudio, textos, etc, surge um outro tipo de ambiente, chamado de fluxo contínuos de dados. Ao contrário das bases tradicionais, as bases com um fluxo contínuos de dados, aqui neste trabalho tratadas como *data stream*, não têm limites de tamanhos, frequentemente, apenas os dados mais recentes são importantes, além de estarem expostas a constantes mudanças (HAN e KAMBER, 2006). Neste contexto, as técnicas tradicionais de mineração não se mostram eficazes e a análise de amostragens, se aplicada incorretamente pode gerar problemas (DOMINGOS e HULTEN, 2001a).

[1] Internet das coisas: do inglês *Internet of Things* ou apenas IoT, é uma rede crescente de dispositivos do cotidiano, desde máquinas industriais até bens de consumo, que podem compartilhar informações.
Fonte: https://mitpress.mit.edu/books/internet-things

Com isso, um tema crescente é a mineração de fluxos contínuos de dados ou *data stream mining*, que trabalha no seu primeiro estágio com análise exploratória de dados estatísticos (GABER et al. 2005), utilizando de teste de hipótese para dar suporte à amostragem utilizada. Dentro de *data stream mining* é preciso lidar com os recursos de uma forma muito mais eficiente, promovendo o foco em três dimensões principais (BIFET, 2015):

- Acurácia (erro de predição);
- Memória ou espaço necessário;
- Tempo necessário para treinar exemplos e então gerar predição.

Os principais desafios de *data stream mining*, conforme Mohammed e Soliman (2010), são: limitação de armazenamento e processamento em tempo real. Na última década, este tema tem despertado interesse de diferentes áreas, e com isso centenas de técnicas têm sido propostas, endereçadas para resolver problemas de análise rápida de *data stream* (GABER, 2012). Dentro dessa área, a predição de valores merece destaque, principalmente para monitoramento, como: detalhes de tráfego e rotas (veículos e aviões), previsões climáticas, diagnósticos de fraudes no sistema financeiro e sensores de redes (GAMA, 2010).

A oportunidade do ganho de informações sobre inúmeras fontes de dados traz consigo as ambições das áreas de negócios, juntamente com os desafios técnicos para propor soluções para resultados rápidos e com maior precisão. Nesse sentido, a expressão *big data* acompanha a necessidade de um *big data analyzed* (SNIJDERS et al., 2012), principalmente com a busca por resultados de análises on-line, onde as mudanças devem ser consideradas e tratadas de forma que possam auxiliar na tomada de decisão.

Apesar de muitas técnicas estarem sendo desenvolvidas, ainda é uma área de estudos recente, e com muito espaço para melhorias e avanços. O uso de métodos de predição, como a regressão, tem surgido com várias propostas para *data stream*. Entretanto Ikonomovska (2015) destaca que o melhor método para regressão de *data stream* ainda não foi publicado.

Ikonomovska, Gama e Džeroski (2011) propõe um algoritmo para regressão on-line com *data stream*, chamado de FIMT-DD, nesse trabalho é apresentado um comparativo entre algoritmos, explorando diferentes abordagens, mas não detalhando uso de recursos como tempo de execução e memória em alguns casos. Já Shaker e Hüllermeier (2012) apresentam um algoritmo, o IBLStreams, que é utilizado tanto para classificação quanto regressão, entretanto apresentam poucos experimentos para a tarefa de regressão. Almeida, Ferreia e Gama (2013) apresentam o algoritmo AMRules, sendo esse comparado com os dois algoritmos anteriores. Seguindo a mesma linha, Barddal, Gomes e Enembreck (2015) propõem o algoritmo SFNRegressor, onde apresentam comparativos com todos os outros algoritmos também, porém deixam de considerar aspectos fundamentais de *data stream*, como tempo de execução e memória.

A dificuldade de encontrar, na literatura, um algoritmo que faça regressão em *data stream mining*, de forma eficiente, que possa ser considerado o estado da arte da técnica, tem feito surgir inúmeras propostas, onde o objetivo parece ser sempre apresentar um novo algoritmo, e não explorar de forma mais exaustiva o que já foi desenvolvido. Dessa forma, Bifet et al. (2015) ressaltam que a avaliação de algoritmos é uma tarefa delicada, e que diversos pesquisadores têm apresentado resultados considerados bons, porém infortunadamente utilizando de práticas inválidas. Com isso apresentam critérios que devem ser considerados na avaliação deste tema, como: a validação da metodologia e o mecanismo de esquecimento para treinamento.

1.1 Objetivos

O objetivo geral deste trabalho é a realização de uma investigação e comparação de algoritmos de regressão para *data stream mining*, publicados recentemente (FIMT-DD, AMRules, IBLStreams e SFNRegressor), considerando conceitos fundamentais como o uso de memória, tempo de execução e erro de predição, respeitando que ambos são igualmente importantes, conforme Domingos (2015) e Bifet (2015). Buscando atender lacunas deixadas pela literatura, demonstrando as principais características, limitações e comportamento de cada algoritmo quando sujeitos a análises, que ainda não foram exploradas.

Para isso, tem-se como objetivos específicos do trabalho o seguinte:

- Realização de experimentos utilizando apenas bases de dados massivas, considerando o mínimo de 500 mil instâncias;
- Realização de experimentação dos algoritmos de regressão sobre bases com apenas um atributo alvo (*single-target*);
- Realização de experimentos com as bases de dados que apresentem mudanças de conceitos (*concept drift*);
- Produção de uma metodologia que possibilite a continuação desse estudo, assim como a possibilidade de reproduzir os experimentos executados;
- Geração de uma análise crítica dos resultados obtidos, apontando vantagens, desvantagens e limitações de cada algoritmo frente aos critérios de memória, tempo e erro.

1.2 Estrutura

Este trabalho apresenta no Capítulo 2 os conceitos fundamentais de *data stream*, abordando caracterítiscas do ambiente, técnicas, métodos de avaliação, *concept drift*, a tarefa de regressão e também a estrutura de um algoritmo para *data stream*. No Capítulo 3 são expostos os trabalhos relacionados, onde são descritos todos os algoritmos estudados neste trabalho, relatando os resultados obtidos em suas respectivas abordagens. Já no Capítulo 4, a abordagem experimental, as ferramentas utilizadas, bases de dados e configuração das experimentações são apresentadas. No Capítulo 5 é dedicado em apresentar os resultados obtidos, além de apontamentos e características relevantes observadas. Por fim, no Capítulo 6 são apresentadas as conclusões deste trabalho investigativo, juntamente com as contribuições e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos para a compreensão da metodologia adotada nesse trabalho, assim como a temática dos trabalhos relacionados. Assim, são descritos inicialmente conceitos relacionados à mineração de dados, com foco na tarefa de regressão. Posteriormente, a seção 2.4 se dedica à contextualização de mineração de *data stream*, *concept drift* e formas de avaliação dos algoritmos de mineração. Por fim, o capítulo consolida os conceitos descritos fornecendo considerações que ressaltam a relação dos conceitos fundamentais com o restante do trabalho.

2.1 Descoberta do Conhecimento em Bases de Dados

Dentro da Descoberta do Conhecimento em Bases de Dados, ou simplesmente KDD, a mineração de dados é uma das etapas. A mineração de dados é a etapa dedicada a resolver problemas por meio da análise de dados já existentes e armazenados (HALL et al., 2011). Enquanto Han e Kamber (2006) destacam que a mineração de dados pode ser vista como o resultado da evolução natural da tecnologia da informação. Economistas, estatísticos e analistas trabalham há muito tempo com o objetivo de encontrar padrões nos dados de forma automática (ou semiautomática) para identificar, validar e utilizar a predição de tendências de eventos futuros com base em informações passadas.

O valor do conhecimento armazenado é dependente da habilidade de extrair relatórios úteis, identificar eventos e tendências interessantes, suporte às decisões e políticas baseadas em análise estatística (FAYYAD et al., 1996). Assim o processo de KDD é não-trivial, definido por várias etapas, interativo e iterativo, utilizado para identificação de padrões e extração do conhecimento de conjuntos de dados. As cinco etapas do KDD, Figura 1, são especificadas como:

a) **Seleção de dados:** nesta etapa ocorre a seleção ou segmentação dos dados apropriados que serão utilizados no processo de descoberta de conhecimento, onde a fonte dos dados pode ser várias bases, arquivos ou outros recursos;

b) **Pré-processamento:** onde ocorrem operações de limpeza, eliminando ruídos, inconsistências de integridade e eventuais tratamentos, como para suprir a ausência de valores;

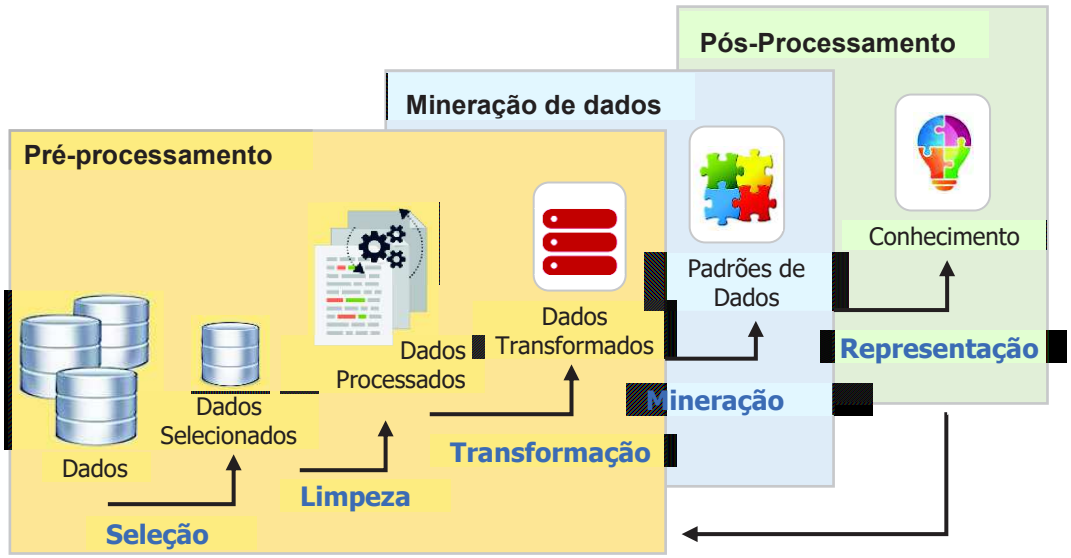
c) **Transformação:** responsável pela padronização dos dados, para disponibilizar os dados em formatos adequados para a próxima etapa;

d) **Mineração de dados:** concentra técnicas e algoritmos específicos com a capacidade de produzir modelos preditivos e classificatórios para a identificação de padrões;

e) **Avaliação:** etapa em que é realizada a análise e interpretação dos resultados, relacionamentos e descoberta de novos fatos.

Esse fluxo é cíclico e a definição deste processo, incluindo configurações específicas, depende muito do domínio de atuação. Sendo assim, é importante observar que a solução adotada para um problema pode não atender outros cenários.

Figura 1: Fases do processo de KDD.



Fonte: Adaptado de Fayyad (1996).

Dentro da etapa de mineração de dados, onde tem-se como objetivo estabelecer associações, relações e padrões, que são difíceis de visualizar manualmente, são utilizadas diferentes técnicas para obter os resultados, com destaque para algoritmos de aprendizado de máquina e análise estatística (HAN e KAMBER, 2006). As principais tarefas de mineração de dados são:

- **Classificação:** predizer itens de uma determinada classe;
- **Regressão:** predizer variáveis contínuas;
- **Regras de Associação:** encontrar associações que ocorrem frequentemente;
- **Agrupamento:** encontrar grupos similares nos dados;
- **Visualização:** mecanismos (gráficos) que auxiliam a descoberta humana;
- **Sumarização:** descrição de um grupo ou classe;
- **Desvio:** encontrar mudanças, alterações.

Mitchell (1999) sintetiza que a mineração de dados trata a questão de como melhor utilizar o histórico de dados para descobrir padrões e fornecer apoio às decisões futuras.

2.2 Aprendizado de Máquina

O aprendizado de máquina (*Machine Learning*) é uma subárea da Inteligência Artificial, que trata do desenvolvimento de técnicas computacionais para aquisição automática de conhecimento ou padrões a partir de exemplos ou observações (MITCHELL, 1997). Sendo inerentemente multidisciplinar, envolvendo probabilidade e estatística, teoria da complexidade computacional, teoria da informação, neurobiologia e outras áreas de estudo.

A definição de aprender, no contexto de aprendizado de máquina, pode ser definida como qualquer programa de computador que melhora seu desempenho (P) em alguma tarefa (T) através da experiência (E). Em geral, para ter um problema de aprendizado bem definido,

é necessário ter três características identificadas: a classe de tarefas, a medida de desempenho a ser melhorado, e a fonte de experiência (JORDAN e MITCHELL, 2015). Um exemplo seria o problema de aprendizado para um jogo de dama, onde tem-se:

- Tarefa $\boldsymbol{T}$: jogar dama;
- Medida de desempenho $\boldsymbol{P}$: percentual de partidas vencidas contra outros oponentes;
- Experiência de treinamento $\boldsymbol{E}$: jogar partidas contra si próprio.

Segundo Mitchell (1997), algoritmos de aprendizado de máquina têm sido utilizados em uma variedade de domínios de aplicações, sendo especialmente úteis em problemas de mineração de dados. Dentro deste contexto, uma questão importante é como utilizar no treinamento a distribuição de exemplos que melhor representa a experiência (E) garantindo o desempenho, ou seja, a amostragem de dados (definida em detalhes seção 2.4).

O tipo de aprendizado mais utilizado, conforme Domingos(2012), Marsland (2015) e Mitchell e Jordan (2015) é o supervisionado, ou seja, se conhece o atributo alvo da predição e ele é utilizado para o processo de aprendizado. Dentro do aprendizado supervisionado, quando as classes as quais se deseja prever são discretas, o problema é resolvido com a tarefa de classificação, quando são contínuas é resolvido com a regressão. É comum encontrar na literatura os termos "regressão" e "predição numérica" como sinônimos (HAN e KAMBER, 2006), entretanto existem algumas técnicas de classificação que também podem ser adaptadas para predição, como Máquinas de Vetores de Suporte (*Support Vector Machines* - SVM), Redes Neurais Artificiais e classificador *k*-NN (*k-Nearest Neighbor*).

Dessa maneira, um formato muito comum de representar os dados para um sistema de aprendizado, na mineração de dados, é com um conjunto no formato de atributo-valor, conforme demonstrado no Quadro 1. Sendo um conjunto composto por n exemplos e d atributos (características), nessa tabela, a linha i refere-se ao i-ésimo exemplo (i=1,2,3..., n) e a entrada x_{ij} refere-se ao valor do j-ésimo (j=1,2,..., d) atributo X_j do exemplo i. A última coluna, representada por Y, corresponde ao atributo-meta (classe).

Quadro 1: Conjunto de dados no formato comumente utilizado para o aprendizado.

Exemplo	$\boldsymbol{X_1}$	$\boldsymbol{X_2}$	…	$\boldsymbol{X_d}$	$\boldsymbol{Y}$
1	x_{11}	x_{12}	…	x_{1d}	y_1
2	x_{21}	x_{22}	…	x_{2d}	y_2
3	x_{31}	x_{32}	…	x_{3d}	y_3
⋮	⋮	⋮	⋱	⋮	⋮
$\boldsymbol{n}$	x_{n1}	x_{n2}		x_{nd}	y_n

Fonte: Adaptado de Mitchell (1997).

A predição geralmente ocorre via mapeamento $f(x)$, que produz uma saída y para cada entrada x (ou então uma distribuição de probabilidade sobre y dado x). A Tabela 1 apresenta exemplos de problemas com suas respectivas funções, onde pode ser visto que x representa os atributos ou características de um determinado conjunto de dados.

Tabela 1: Exemplos de problemas e o mapeamento $f(x)$ com o resultado esperado.

Exemplo	x	$f(x)$
Avaliação de risco de crédito	Características do cliente	Aprovado ou Não Aprovado
Diagnóstico de doença	Características do paciente	A doença do paciente
Reconhecimento facial	Bitmaps das faces de pessoas	Nome da pessoa

Fonte: Adaptado de Domingos (2015).

O Aprendizado de Conceito (*Concept Learning*) é a aquisição da definição de uma categoria genérica a partir de exemplos positivos ou negativos dessa categoria (MITCHELL, 1997). Essa tarefa é realizada a partir de um conjunto de dados, buscando pela hipótese que melhor categorize esse conjunto, com o objetivo de "aprender" um conceito usando um conjunto de treinamento. Geralmente, um conjunto de exemplos é dividido em dois subconjuntos disjuntos: o conjunto de treinamento que é utilizado para o aprendizado do conceito e o conjunto de teste usado para medir o grau de efetividade do conceito aprendido (MITCHELL, 1997).

Dentro desta abordagem, Diettriech e Kong (1995) destacam que ao escolher uma linguagem de representação das hipóteses, também conhecida como *bias* de representação ou *bias* de linguagens, a técnica pode preferir determinadas hipóteses em detrimento de outras, chamada de *bias* de preferência. Dentro desse contexto tem-se também o *bias* de busca, responsável por definir como as hipóteses serão procuradas num espaço. Como exemplos, tem-se os algoritmos de árvore de decisão ID3 e C4.5 que preferem árvores mais curtas em relação às mais longas. Mitchell (1980) afirma que todos os indutores possuem alguma forma de *bias*, com isso o aprendizado sem *bias* é impossível.

Apesar de aprendizado de máquina ter seus conceitos definidos há tempo na literatura, existem muitos algoritmos disponíveis e vários são publicadas anualmente, definir qual deles utilizar para uma determinada aplicação pode ser um desafio. Para isso, Domingos (2012) define que a chave para não se perder nesse universo é avaliar que o aprendizado de máquina consiste da combinação de três componentes:

- **Representação**: o classificador deve ser representado em alguma linguagem formal que possa ser controlada com computação. Por outro lado, a escolha de um representante para o aprendizado é equivalente a escolha de um conjunto de classificadores que possibilite aprender. Isso é chamado de espaço da hipótese do aprendizado. Se um classificador não está dentro do espaço da hipótese, não pode ser aprendido;
- **Avaliação**: uma função de avaliação (também conhecida como função objetivo) é necessária para distinguir bons e maus classificadores. A função de avaliação usada internamente pelo algoritmo pode ser diferente de uma utilizada externamente para otimizar a classificação;
- **Desempenho**: é preciso um método para encontrar entre os classificadores o que tenha melhor pontuação, denominado *score*. A escolha da técnica de avaliação do desempenho é a chave para eficiência de um classificador.

Entre as diversas abordagens existentes para estimar o erro de um classificador, Japkowicz e Shah (2011) apresentam um conjunto de métodos disponíveis. Sendo eles divididos em dois tipos: sem reamostragem (*No Re-sampling*) e com reamostragem (*Re-sampling*). Dentro do grupo que não utiliza a reamostragem, o método mais utilizado é o *holdout*. Enquanto para os métodos com reamostragem o destaque é a utilização do *cross-*

validation, que possui três variações: *Stratified k-fold Cross-Validation*, *Non-Stratified k-fold Cross-Validation* e *Leave-One-Out*. Na seção 2.4 são explicados os métodos utilizados dentro do contexto de mineração para *data stream.*

2.3 Regressão

A regressão é uma tarefa que permite gerar modelos preditivos para valores contínuos. Tendo como objetivo encontrar a relação entre um conjunto de atributos de entrada (variáveis preditoras) e um atributo-meta contínuo. Com isso, sendo $x = \{x_1, x_2, \dots, x_d\}$ os atributos de entrada e y o atributo-meta, o objetivo é encontrar um mapeamento, também chamado de função objetivo, definido pela equação (1) (APTÉ e WEISS, 1997).

$$y = f(x_1, x_2, \dots, x_d) \tag{1}$$

Conforme Dean (2014), o termo regressão foi definido por Francis Galton (1822-1911) para descrever o processo biológico de valores extremos que se deslocam em direção a uma média da população. Um exemplo comum disso são filhos de pais altos (acima da média de altura) "regredindo" para a média da população, da mesma forma o inverso. Antes de Galton, a técnica era referida como "mínimos quadrados" (*least squares*).

A regressão pode ser utilizada para modelar o relacionamento entre uma ou mais variáveis independentes (preditora) e uma variável dependente (resposta) de valor contínuo. Existem tipos diferentes de regressão, sendo linear e não-linear os principais (HAN e KAMBER, 2006). Como o foco deste trabalho é a regressão para uma única variável, não são abordadas técnicas para regressão de multiplas variáveis, ou múltiplos níveis.

A regressão linear compreende responder uma variável y a partir de uma única variável preditora x (HAN e KAMBER, 2006). Representa a forma mais simples de regressão, e pode ser resolvida com uma função linear, qual tal a equação (2):

$$y = b + dx, \tag{2}$$

onde b e d são os coeficientes de regressão, sendo b a interceptação da reta no eixo vertical, e d o declive (coeficiente angular) da reta, enquanto x é a variável independente. Para determinar os coeficientes é aplicado o método dos mínimos quadrados. Sendo d então definida conforme a equação (3):

$$d = \frac{\sum_{i=1}(x_i - \bar{x}) - (y - \bar{y}))}{\sum_{i=1}(x_i - \bar{x})^2}, \tag{3}$$

onde $\bar{x}$ é a média de todos os valores de x, da mesma forma que $\bar{y}$ é a média de todos os y, a partir da definição do d, é possível definir b, conforme a sequinte equação:

$$b = \bar{y} - d\,\bar{x} \tag{4}$$

Entretanto, a simples regressão linear tem suas limitações, e modelos de dados mais complexos exigem outras técnicas. A regressão não-linear, que segundo Apté e Weiss (1997) tem emergido com diversos trabalhos e tem sido explorada de muitas maneiras, muitas vezes apresenta um problema de grandeza polinomial, que exige uma transformação para que o modelo possa ser convertido em uma regressão linear (HAN e KAMBER, 2006).

Gelman e Hill (2007) destacam ainda que outros métodos para regressão são utilizados, tais como redes neurais artificiais e árvores de regressão. O uso de árvores de decisão em regressão já foi proposto por Breiman (1984), com o algortimo CART (*Classification and Regression Trees*). Uysal e Güvenir (2013) citam também que algoritmos IBL (*Instance-Based Learning*) são facilmente utilizados para regressão, assim como a regressão baseada em regras (WEISS e INDURKHYA, 1993).

2.3.1 Métricas de Avaliação

Na regressão a predição resulta em um valor numérico contínuo, envolvendo assim um grau de incerteza quanto a exatidão do valor predito (HAN e KAMBER, 2006). Com isso, ao invés de focar no valor exato, o objetivo é minimizar a distância entre o valor predito (y_i') e o valor real (y_i). Na literatura existem diferentes nomenclaturas para as duas principais medidas de avaliação: MSE (*Mean Squared Error*) e MAE (*Mean Absolute Error*).

A MSE é chamada desta forma por Armstrong e Collopy (1992), Han e Kamber (2006) e Bennett et al. (2013), e é denominada de *Variance* por Weiss e Indurkhya (1995), assim como Uysal e Güvenir (2013). Neste trabalho é adotada a primeira forma (MSE), que é definida:

$$MSE = \frac{1}{n}\sum_{i-1}^{n}(y_i - y_i')^2, \tag{5}$$

onde n é o número de amostras, i a posição da amostra, y_i corresponde ao valor real e y_i' é o valor predito. Uma derivação desta métrica é o RMSE (*Root Mean Squared Error*), eq. (6), que é comumente utilizado nos modelos preditivos (ARMSTRONG e COLLOPY, 1992), sendo útil na medida em que o erro avaliado é da mesma magnitude que a quantidade a ser predita.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i-1}^{n}(y_i - y_i')^2} \tag{6}$$

Enquanto a MAE é chamada desta forma por Abramowitz e Stegun (1972), Han e Kamber (2006) e Bennett et al. (2013), outros trabalhos chamam de MAD (*Mean Absolute Deviation*). Neste trabalho é adotada a primeira forma, sendo definida por:

$$MAE = \frac{1}{n}\sum_{i-1}^{n}|y_i - y_i'| \tag{7}$$

Algumas vezes são utilizadas métricas para avaliar erros relativos, sendo RRSE (*Root Relative Squared Error*) a mais utilizada, conforme a equação (8):

$$RRSE = \sqrt{\frac{1}{n}\sum_{i-1}^{n}\frac{(y_i - y_i')^2}{(y_i - \bar{y})^2}}, \tag{8}$$

onde $\bar{y}$ é a média de todos os y_i utilizados. Segundo Han e Kamber (2006), a MSE exagera a presença de *outliers* (discrepâncias), enquanto a MAE não.

2.4 Mineração de *Data Stream*

O processo de mineração de *data stream* difere do processo de mineração de dados tradicional devido ao ambiente e características dos dados serem diferentes (HAN e KAMBER, 2006). Os dados não têm limites de tamanho, a chegada é contínua, sendo que frequentemente apenas os dados mais recentes são considerados importantes, além de estarem expostos a constantes mudanças de conceito (tema abordado na subseção 2.4.1). Na Tabela 2 são listadas as principais diferenças entre mineração de dados tradicionais e *data stream*, com destaque para os recursos de tempo de processamento e memória, onde dentro do processo de mineração de dados tradicional não há um limite à execução das tarefas, utiliza-se o quanto o ambiente tem disponível.

Tabela 2: Diferenças entre mineração de dados tradicionais e *stream data*.

	Tradicional	***Data Stream***
Número de passos	Múltiplos	Único
Tempo de processamento	Ilimitado	Restrito
Memória usada	Ilimitado	Restrito
Tipo de resultado	Acurácia	Aproximação
Distribuído	Não	Sim

Fonte: Adaptado de Gama e Rodrigues (2007).

Dados relativos a *data stream* estão inseridos em diversos setores, como: sistemas com geração em tempo de real, sistemas de redes de comunicação, tráfego de internet, transações on-line no sistema financeiro, redes de varejo, sistemas inteligentes de energia, processos industriais, experimentos científicos e de engenharia, sensores remotos de localização e outros sistemas dinâmicos (DOMINGOS e HULTEN, 2001b).

Domingos e Hulten (2001b) definem como propriedades desejadas para sistemas de aprendizado em mineração de *data stream* os seguintes pontos:

- Requerer um pequeno espaço de tempo para os exemplos de dados;
- Utilizar um limite de memória principal fixa, independentemente do número de exemplos;
- Construir um modelo de decisão a partir de uma única leitura de dados de treino;
- Permitir criar um modelo em qualquer instante de tempo;
- Deve ser independente da ordem de chegada dos dados;
- Deve ser capaz de tratar *concept drift*[2], para modelos de dados estacionários deve ser capaz de produzir modelos que são próximos aos obtidos a partir de aprendizado em *batch*[3].

Para garantir essas propriedades desejadas, Gama, Sebastião e Rodrigues (2009) listam as três dimensões que influenciam diretamente no processo de aprendizado de *data stream*:

[2] *Concept drift*: termo técnico em inglês para definir mudança de um conceito já aprendido.

[3] *Batch*: um tipo de treinamento que utiliza a abordagem de ciclos, onde os ajustes de pesos ocorrem após o processamento de todos os exemplos (CLEARWATER et al., 1989).

- **Espaço**: a memória disponível é fixa;
- **Tempo**: processar os dados assim que chegam;
- **Poder de generalização**: o quão efetivo é o modelo para capturar os dados com taxas aceitáveis de acertos.

É importante destacar que o espaço e o tempo influenciam diretamente no poder de generalização. Como uma das principais características desse tipo de dado é o tamanho, não há viabilidade de submeter todos os dados a um processo de mineração de dados, novas abordagens são adotadas nesse contexto e o uso de amostragem randômica é uma delas (HAN e KAMBER, 2006). Porém, como obter uma amostragem imparcial não conhecendo o tamanho total dos dados? A resposta para esta questão é a criação de uma "reserva de amostragem" (*reservoir sampling*) (VITTER, 1985), que tem como objetivo oferecer uma representação da amostragem dos dados observados.

O uso de amostragens randômicas é um método comumente utilizado nas técnicas de captura de dados para *data stream* (BABCOCK et al., 2002). A abordagem mais simples sugere a captura de amostras em intervalos de tempos periódicos, que oferece uma opção de "abranger" os dados, mas pode envolver uma enorme perda de informações em bases com altas taxas de alterações. Vitter (1985) propôs uma abordagem de análise dos dados em uma única passagem, removendo exemplos de uma amostragem baseada em uma determinada probabilidade. Domingos e Hulten (2000) apresentaram outra abordagem utilizando o limitador de Hoeffding (HOEFFDING, 1963) para teste estatístico, explicado na seção 2.5.1. Para a atualização de dados de treino, um mecanismo de esquecimento é utilizado com a inserção de novos valores e atualização ou descarte de valores antigos (AGGARWAL, 2006). Para a efetivação deste mecanismo, abordagens de janelamento como: janelas deslizantes, fixas e janelas adaptativas são adotadas (BIFET et al., 2015).

Outra importante característica de *data stream* é o *concept drift* (mudança de conceito). O modelo de aprendizado deve prever que, durante a evolução dos dados, alterações ocorrerão e será preciso ajustar o modelo adequando-se aos dados mais recentes. Nas próximas subseções são detalhados os conceitos das principais características de modelo de aprendizagem para *data stream*.

2.4.1 *Concept Drift*

Em muitas aplicações, algoritmos de aprendizado que atuam em ambientes dinâmicos de *data stream* ficam sujeitos a mudanças na distribuição dos dados de entrada, este termo é definido dentro de aprendizado de máquina e mineração de dados como *concept drift* (GAMA e PEDERSEN, 2007). Como consequência disso, os resultados passam a ser menos precisos. A definição de *concept drift* foi apresentada pela primeira vez por Basseville e Nikiforov (1993), onde demonstraram a importância de identificar esta situação, em que um determinado conceito aprendido pode ao longo do tempo sofrer alterações na evolução dos dados.

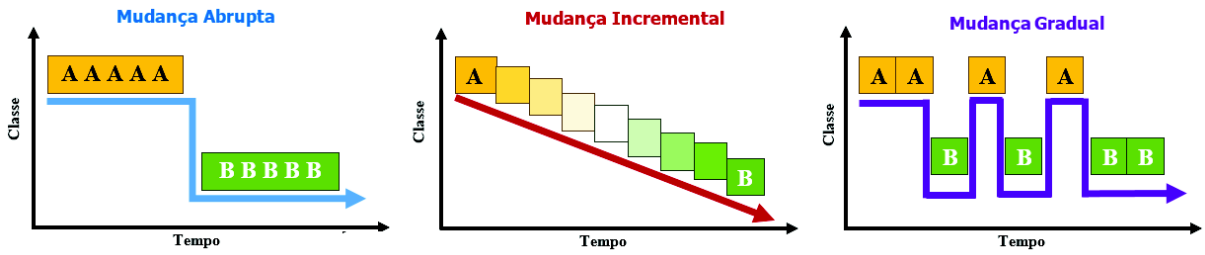
As mudanças podem ser reais ou virtuais. Quando reais, as alterações ocorrem na distribuição condicional da variável de saída a ser predita, $p(y|x)$ (GAO et al., 2007). Quando virtuais ocorrem na distribuição dos dados, $p(x)$. *Concept drift* representa uma mudança gradual no conceito, apesar de gradual ser apenas um dos tipos de *concept drift*, essas mudanças ocorrem frequentemente no mundo real. Conforme Bifet et al. (2009), quando um conceito é aprendido, tornando-se estático, o mesmo não pode ser utilizado para classificar instâncias futuras de forma indefinida. Essas mudanças dependem da área de pesquisa, podem ser

evoluções temporais, variações temporáriais ou não estacionárias. Tsymbal (2004) cita um típico exemplo, a previsão climática, onde muitas vezes tem-se mudanças radicais como as estações. Outro exemplo é referente aos padrões de preferências de compras de clientes que muitas vezes podem alterar com o tempo, dependendo do dia da semana, avaliação de alternativas, inflação e outros fatores.

Segundo Gama (2010), as causas de *concept drift* também são importantes, pois podem ter origem de duas formas: mudança no contexto do aprendizado ou mudanças nas variáveis observadas. Existem três tipos de *concept drift*, conforme podem ser vistos abaixo, seguidos da Figura 2, onde é exemplificado a ocorrência de mudanças de cada tipo.

- **Abrupta**: identificada logo após a ocorrência, onde um determinado conceito é instantaneamente alterado, sendo substituído por outro;
- **Incremental**: as mudanças ocorrem de forma mais lenta;
- **Gradativa**: ocorrem alternâncias entre os conceitos.

Figura 2: Exemplos de mudanças de conceito.



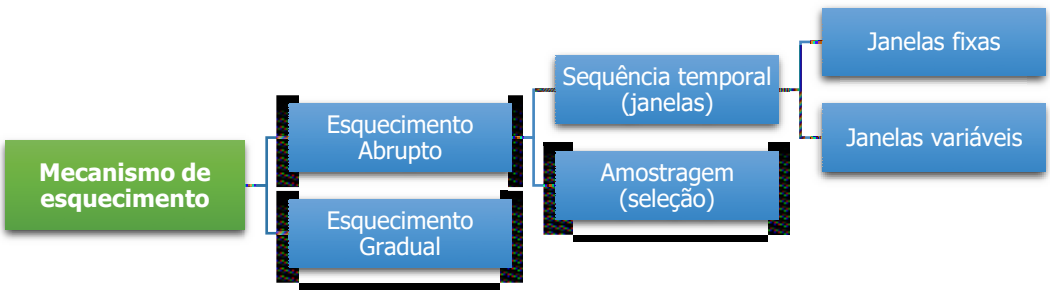
Fonte: Elaborado pelo autor.

Dentro de um cenário de *data stream* deve-se prever que as mudanças não ocorrerão em um determinado tipo de forma exclusiva, podendo ocorrer variações entre os tipos. De qualquer forma, a importância na identificação deles é necessária para projetar estratégias de adaptação (GAMA, 2010).

2.4.2 Mecanismos de Esquecimento

O aprendizado sobre *concept drift* não requer apenas a atualização do modelo com novos dados, mas também é necessário esquecer os dados antigos (GAMA et al., 2014). Os mecanismos de esquecimento estão relacionados aos métodos de detecção de mudanças. Dentro deste contexto, a Figura 3 apresenta a taxonomia dos mecanismos de esquecimento para *data stream*.

Figura 3: Taxonomia dos mecanismos de esquecimento.



Fonte: Adaptado de Gama et al. (2014).

Os dois principais mecanismos de esquecimento, abrupto e gradual, são descritos da seguinte forma:

a) Esquecimento Abrupto: as observações são definidas por janelas de informações para aprendizado. O uso de janelas deslizantes (*sliding window*) para análise de dados é apresentado com dois tipos, baseado em sequência e baseado em tempo, conforme Babcock et al. (2002):

Janelas baseadas em sequência: onde é especificado o número de exemplos que será utilizado. Dois modelos diferentes são utilizados:

(a) janelas deslizantes com tamanho fixo *w*, que apresentam uma abordagem simples e semelhante à estrutura de dados do tipo fila, enquanto um elemento *j* é observado e inserido na janela, outro elemento $j - w$ (onde w é o tamanho da janela) é esquecido.

(b) janelas de pontos de referência, são de tamanho variável (w).

Janelas baseadas em tempo: onde são atualizadas em um intervalo de tempo.

A ideia básica deste mecanismo é permitir que sejam utilizadas apenas as amostragens mais recentes, mais precisamente a cada *t* intervalo de tempo. Estes elementos expiram a cada $t + w$, onde *w* é o tamanho da janela. Esta abordagem se mostra útil em cenários onde apenas os dados mais recentes devem ser considerados (HAN e KAMBER, 2006), sendo assim um dos mecanismos mais usados para a indução do aprendizado de *data stream*, onde apenas os dados dentro destas janelas deslizantes são utilizados.

Uma alternativa em relação às janelas deslizantes é o mecanismo de amostragem. Sendo *Reservoir Sampling* (VITTER, 1985) um dos algoritmos mais utilizados para esta tarefa, é o que se caracteriza em realizar uma amostragem aleatória uniforme de m elementos sem conhecer o tamanho do conjunto de dados.

b) Esquecimento Gradual: é uma abordagem para esquecimento da memória cheia, o que significa que os exemplos não são completamente descartados da memória. Os exemplos em memória são associados a pesos, onde estão relacionados com a idade dos exemplos. Estes pesos são definidos pela relação inversa com a idade, conforme aumenta a idade, diminui a importância. Suponha que no momento *i*, a estatística suficiente (HOGG e CRAIG, 1995) armazenada é S_{i-1}, e observado o exemplo x_i, assumindo a função de agrupamento $A(x, S)$, a nova estatística suficiente é calculada como $x_i = A(x_i, \alpha S_{i-1})$, onde $\alpha \in (0,1)$ é o fator de enfraquecimento (*fading factor*). Desta forma, a informação mais antiga torna-se menos importante (GAMA et al., 2014).

Outras abordagens para configuração deste mecanismo são: as técnicas de decaimento linear, encontradas em Koychev (2002), e técnicas de decaimento exponencial, presentes em Klinkenberg (2004). Esta última técnica de pesos atua de acordo com a idade, utilizando uma função exponencial de envelhecimento $\omega_\Lambda(x) = exp(-\Lambda j)$, onde o exemplo x aparece j ocorrências atrás. O parâmetro Λ controla a rapidez que os pesos diminuem. Os valores de Λ com baixo peso são associados a exemplos antigos, como eles são considerados menos informativos para a predição atual. Se $\Lambda = 0$, todos os exemplos têm o mesmo peso.

2.4.3 Detecção de *Concept Drift*

A detecção de *concept drift* não é uma tarefa trivial, conforme mencionado na seção anterior pode ser real ou virtual, e ainda deve prever a ocorrência de ruídos que diferem efetivamente de uma mudança real de conceito (AGGARWAL, 2006). A maior dificuldade de detectar um *concept drift* está em distinguir uma verdadeira mudança de conceito de um ruído (TSYMBAL, 2004).

Dentro dos requisitos para modelos preditivos em ambientes não estacionários se faz necessário ter um mecanismo para detectar e adaptar a evolução dos dados ao longo do tempo, e de outra forma não degradar o desempenho da acurácia do modelo (GAMA et al., 2014). Quatro requisitos são importantes neste contexto:

(1) Detectar *concept drift* e adaptar-se se necessário o mais cedo possível;

(2) Distinguir entre ruídos e *concept drift* e ser adaptivo às mudanças, sendo o mais robusto possível contra ruídos;

(3) Operar com baixo número de exemplos com chegadas constantes, respeitando os limites de memória e tempo;

(4) Projetar um mecanismo de monitoramento para detectar *concept drift.*

Para identificação dessas alterações é comum o uso de técnicas que tenham monitoramento de indicadores. O passo seguinte à detecção do *concept drift* é a adaptação do modelo, este processo está fortemente relacionado ao mecanismo de esquecimento. Algoritmos para aprendizado em *data stream* que trabalham com dados não estacionários realizam o aprendizado on-line, utilizando de estratégias de adaptações às mudanças, porém a principal limitação dessa abordagem é que a reação é lenta (GAMA et al., 2014).

Uma das técnicas mais utilizadas para detectar *concept drift* é o *Page-Hinkley Test* (PHT) (PAGE, 1954), que é uma variante do CUSUM (*Cumulative Sum*), sendo esta uma técnica de análise sequencial que utiliza os princípios de *Sequential Probability Ratio Test* (SPRT). Para realizar o teste monitora-se a diferença entre duas variáveis m_T e M_T, onde a variável m_T é definida como a diferença acumulativa entre os valores observados e a média até o momento T, eq (9):

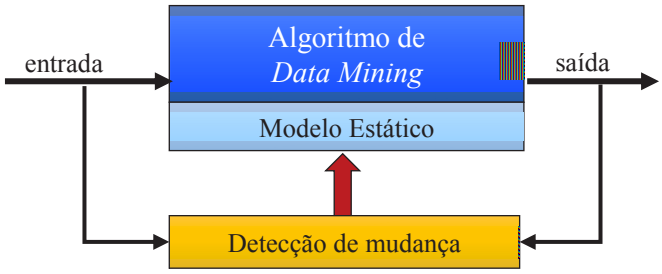
$$m_T = \sum_{t=1}^{T} (x_t - \bar{x}_T - \varphi)\,, \tag{9}$$

onde, $\bar{x}_T = \frac{1}{t}\sum_{t=1}^{T} x_t$ e φ especifica a magnitude de tolerância da mudança (GAMA et al., 2014). A segunda variável (M_T) é definida como o valor mínimo observado de m_T, conforme equação (10):

$$M_T = \min(m_t, t = 1 \ldots T). \tag{10}$$

A partir disso, se a diferença entre M_T e m_T é maior que um limite (λ), que é definido pelo usuário, o *drift* é sinalizado (SEBASTIÃO e GAMA, 2009). O λ é a quantidade de falsos alarmes. A Figura 4 exemplifica como atua esse tipo de controle no modelo de um aprendizado com *data stream*.

Figura 4: Algoritmo de *data stream* com técnica de detecção de *concept drift*.



Fonte: Adaptado de Bifet et al. (2010).

Uma abordagem de "janela adaptativa deslizante" chamada de ADWIN (*Adptative sliding WINdow*), adequada para detectar *drift* foi proposta por Bifet e Gavaldà (2007). Este método trabalha com janelas deslizantes, definidas por w, utilizando os dados mais recentes e compara a distribuição em duas sub-janelas de w. Sempre que duas grandes sub-janelas, w_0 e w_1, apresentam médias altas e com grande diferença, a sub-janela mais antiga é eliminada e a alteração na distribuição de exemplos é assimilada. O limitador de corte da janela é calculado conforme a equação (11), sendo m apresentado na equação (12).

$$\epsilon_{cut} = \sqrt{\frac{1}{2m} . \ln \frac{4|w|}{\varphi}}, \tag{11}$$

onde w denota o tamanho da sub-janela, m é a média harmônica de $|w_0|$ e $|w_1|$, conforme a equação (12), e $\varphi \in (0,1)$ é o parâmetro que define a tolerância a mudanças, Gama et al. (2014) recomendam o valor de 0,2 para essa configuração.

$$m = \frac{2}{\frac{1}{|w_0|} + \frac{1}{|w_1|}} \tag{12}$$

Destaca-se ainda que Gama et al. (2014) apresentam outros métodos para detecção de *drift*, porém conforme Bifet et al. (2015) o PHT e ADWIN apresentaram melhores resultados em trabalhos recentes. Sendo esses os métodos utilizados nos algoritmos estudados neste trabalho.

2.4.4 Métodos para Avaliação

No processo de aprendizado em lotes, os métodos de validação cruzada, aqui tratada como *cross-validation*, e variantes são os métodos mais utilizados para validação em cenários com tamanho finito de treino. Em ambientes com dados estacionários é a forma mais apropriada de validação (DOMINGOS e GABER, 2007). Já no contexto de *data stream* onde os dados são potencialmente infinitos, gerados por uma distribuição que pode ser alterada ao longo do tempo, são necessárias outras estratégias. A partir disso, duas alternativas viáveis são: *holdout* e *prequential*.

Na avaliação *holdout* são utilizadas amostragens separadas de dados para teste, onde os dados não foram utilizados no treinamento. Tipicamente utilizando dois terços para treinos e um terço destinado para testes (HAN e KAMBER, 2006).

A utilidade desta abordagem está na facilidade de comparar o treinamento oferecendo dados de testes. Nessa configuração, sempre são utilizadas amostragens de exemplos semelhantes a treinamentos em lotes. A desvantagem está na necessidade de dedicar memória adicional para os exemplos de testes (IKONOMOSVKA, 2012), a perda estimada com esta abordagem é a possibilidade de imparcialidade (GAMA, SEBASTIÃO e RODRIGUES, 2009).

Já o método de avaliação *prequential* apresenta a metodologia de teste-e-treino. Neste método, primeiro é realizado o teste com uma determinada amostragem, e em seguida o mesmo conjunto é utilizado para realizar o treinamento. O erro estatístico geralmente é o principal gerenciador deste processo. Nele o erro é calculado baseado na soma acumulada da função de perda $L(y_i, \hat{y}_i)$, que é a diferença entre o valor real e o valor esperado (GAMA, 2010), conforme a equação (13).

$$S_i = \sum_{1}^{n} L(y_i, \hat{y}_i) \tag{13}$$

É importante salientar que no *prequential* não é preciso conhecer o valor verdadeiro de y_i (predição), para todos os pontos no *stream*. Esse tipo de avaliação oferece uma curva de aprendizado que monitora a evolução do aprendizado como um processo (GAMA et al., 2014).

2.5 Algoritmos para *Data Stream*

Os algoritmos para mineração de *data stream* geralmente são derivações de métodos já existentes na literatura, porém com adaptações para atender as características de volume de dados e tempo de processamento. Domingos e Hulten (2000) foram os primeiros a propor uma árvore de decisão adaptada à *data stream*, chamada de Árvore de Hoeffding, posteriomente Ikonomovska, Gama e Džeroski (2011) apresentam FIMT-DD (*Fast Incremental Model Tree with Drift Detection*) também com árvores de decisão. Enquanto o IBLStreams (SHAKER e HÜLLERMEIER, 2012) segue o princípio de IBL (*Instance-Based Learning*), já o AMRules se caracteriza por trabalhar com regras. A Árvore de Hoeffding apresenta aspectos que foram utilizados nos demais trabalhos, e é considerado um algoritmo base em *data stream*.

2.5.1 Árvore de Hoeffding

As árvores de decisão e regressão são conhecidas por oferecerem soluções eficientes em muitos problemas complexos de decisão não-linear, empregando a estratégia de divisão e conquista (GABER et al., 2005). Conforme Domingos e Hulten (2000), os algoritmos clássicos de árvores de decisão C4.5, ID3 e CART assumem que é necessário entrar com todos os exemplos de dados para o treinamento, se mostrando desta forma inviável para *data stream*. A partir disso eles apresentaram uma árvore com base no limite estatístico de Hoeffding (HOEFFDING, 1963), a implementação é chamada de *Very Fast Decision Tree* (VFDT), e foi a primeira para trabalhar *data stream mining*, o pseudo-código está disponível no Algoritmo 1.

Uma árvore de decisão é representada por uma estrutura *top-down*, onde os particionamentos são definidos a partir de um nó raiz posicionado no topo da árvore. Diferentemente das árvores de decisão tradicionais, o algoritmo de Hoeffding induz a árvore de decisão a partir de *data stream* de forma incremental, sem a necessidade de armazenar todos os

exemplos. Através de um limitador de estado, denominado Hoeffding, decide quantos exemplos são necessários até atingir um determinado nível de confiança.

Esse limitador atua da seguinte forma, dado r como um número real randômico, com um intervalo R, e n sendo o número de observações independentes realizadas, define-se $\bar{r}$ como a média calculada a partir das n observações. O limitador de estado, com 1 – δ sendo a probabilidade de escolher corretamente o melhor atributo, deve ser igual a média verdadeira da variável sendo esta, pelo menos, $\bar{r} - \varepsilon$, onde ε é calculado conforme a eq. (14).

$$\varepsilon = \sqrt{\frac{R^2 \ln(\frac{1}{\delta})}{2n}} \tag{14}$$

Algoritmo 1. Pseudo-código do algoritmo de Hoeffding.

Entrada

- S: sequência de exemplos de *data stream*;
- X: a definição de atributos discretos;
- $G(\cdot)$: função de avaliação da divisão do nó;
- δ: probabilidade desejada para a escolha do atributo errado em qualquer nó.

Saída

- H_T: árvore de decisão de Hoeffding.

1: $H_T \leftarrow$ uma árvore com uma única folha l_1 (a raiz);
2: $X_1 \leftarrow \chi \cup \{X_0\}$;
3: $G_1(X_0) \leftarrow$ G obtido por predição da classe mais frequente em S;
4: **Para cada** classe $y_k \in Y$ **faça**
5: **Para cada** valor x_{ij} de cada atributo $X_i \in \boldsymbol{X}$ **faça**
6: $n_{ijk}(l_1) \leftarrow 0$;
7: **Para cada** exemplo $(\bar{x}, \bar{y}) \in S$ **faça**
8: Ordena $(\bar{x}, y)$ dentro da folha l utilizando H_T;
9: Atualiza estatísticas suficientes na folha l;
10: **Para cada** valor atributo $x_{ij} \in \bar{x}$ tal que $X_i \in \boldsymbol{X}_l$ **faça**
11: $n_{ijk}(l_1) \leftarrow n_{ijk}(l_1) + 1$;
12: rótulo l com a classe majoritária entre os exemplos vistos até agora em l;
13: **se** os exemplos vistos até agora em l não são todos da mesma classe **e** $mod(n, n_{min}) = 0$ **então**
14: Computa $G_1(X_i)$ para cada $X_i \in \boldsymbol{X}_l - \{X_0\}$ utilizando o contador $n_{ijk}(l)$;
15: $X_a \leftarrow$ o atributo com mais alto G_l;
16: $X_b \leftarrow$ o atributo com o segundo mais alto G_l;
17: computa Hoeffding limitador ε utilizando a equação (14);
18: **se** $X_a \neq X_0$ **e (** $G_l(X_a) - G_l(X_b) > \varepsilon$ **ou** $\varepsilon < \tau$ **) então**
19: substitua l por um nó interno que divida em X_a;
20: **Para cada** ramo da divisão **faça**
21: adicione uma nova folha l_m com estatísticas suficientes atualizadas;
22: $\boldsymbol{X}_m \leftarrow \boldsymbol{X} - \{X_A\}$;
23: $G_m(X_0) \leftarrow G$ obtida por predição da classe mais frequente em l_m;
24: **Para cada** classe $y_k \in Y$ e cada valor x_{ij} de cada atributo $X_i \in \boldsymbol{X}_m - \{X_0\}$ **faça**
25: $n_{ijk}(l_1) \leftarrow 0$;
26: Retorna H_T

Fonte: Adaptado de Domingos e Hulten (2000).

O que torna atrativo Hoeffding é a sua capacidade para obter os mesmos resultados, independentemente da distribuição de probablidade r. Entretanto, o número de observações necessárias para atingir determinados valores de δ e ε são diferentes entre as distribuições de probabilidade. Para isso, uma medida heurística, denominada $G(X_i)$, é utilizada na escolha de um atributo, a entropia pode ser calculada com *information gain* ou *Gini index*. Onde o objetivo é garantir que, com uma alta probabilidade, o atributo escolhido utilizando n exemplos é o mesmo que seria escolhido utilizando uma quantidade desconhecida de exemplos.

Em resumo, é possível exemplificar o conceito da seguinte forma: a partir de dois atributos, X_a e X_b, obtém-se o ganho de informação de cada um através de uma função de avaliação, $G(X_a)$ e $G(X_b)$. Define-se $\Delta G = G(X_a) - G(X_b)$, e aplica-se a validação, se $\Delta G > \varepsilon$ então o atributo X_a é selecionado. Caso contrário será necessário acumular mais exemplos até que a condição seja atendida.

Destacam-se cinco aspectos importantes deste algoritmo:

(1) Taxa de confiança: a definição de quando deve ocorrer a divisão é relacionada à parametrização do algoritmo, na linha 18 é realizada a verificação que indica se um atributo será utilizado. Essa decisão é fortemente influenciada pela definição do parâmetro δ, dentro do VFDT o valor inicial é definido em 0,000001;

(2) Estatística Suficiente: as estatísticas de cada folha devem ser suficientes para calcular o ganho de informação. Esta característica torna a classificação simples e fácil, através de atributos discretos. Um exemplo dessas informações pode ser visto na Tabela 3;

Tabela 3: Tabela com estatísticas suficientes de três atributos com duas classes.

Atributo	A_1		A_2			A_3				
Valor	A	B	C	D	E	F	G	H	I	**Total**
Classe 1	12	28	5	10	25	13	9	3	15	**40**
Classe 2	34	26	21	8	31	11	21	19	9	**60**

Fonte: Adaptado de Domingos e Hulten (2000).

Neste exemplo (Tabela 3), com cem amostras (soma total das classes), são armazenadas informações de três atributos (A_1, A_2 e A_3) e contabilizado quantas ocorrências cada classe teve em cada valor do atributo;

(3) *gracePeriod* (período de carência): o algoritmo trabalha com um parâmetro (n_{min}) indicando um número mínimo de exemplos que devem ser acumulados antes de calcular o ganho de cada folha;

(4) Pré-poda: o atributo X_0, visto em diversas linhas do pseudo-código, é um "atributo nulo" que representa escolher não transformar a folha em nó;

(5) *tie-breaking* (desempate): o algoritmo utiliza um critério para desempate para acelerar a escolha entre dois atributos quando esses apresentam valores de ganhos muito próximos. Para isso, utiliza o parâmetro τ, utilizado como alternativa para decidir na escolha de um atributo, conforme a linha 18.

Diferentes abordagens para adicionar mecanismos de esquecimento para Hoeffding são utilizados, como a EWMA (*Exponential Weight Moving Average*) ou ADWIN (BIFET, 2009). Conforme Domingos e Hulten (2000), na comparação com árvores de decisão, *Naïve Bayes* e

k-NN, ele tem se mostrado mais eficiente em uso de recursos e classificação para grandes bases de dados.

2.6 Considerações

Neste capítulo foram abordados conceitos necessários para compreensão da temática do trabalho. Inicialmente foi apresentada a área de KDD para contextualizar onde o processo de mineração de dados está inserido, e com isso foram explicados alguns conceitos de aprendizado de máquina.

Foram apresentadas as características de aprendizado de máquina, através da utilização de conjuntos de dados para treinamento, assim como a introdução do termo 'aprendizado de conceito' dentro deste contexto, permitindo a assimilação do *data stream* e do *concept drift*.

O tipo de aprendizado tratado neste trabalho é o supervisionado, onde são exibidos dois métodos: classificação e regressão. Alguns conceitos são comuns entre eles, porém existem diferenças significativas. Na seção 2.3 foram apresentadas as características de regressão, assim como as métricas de avaliação comumente utilizadas, conforme a literatura. Esses aspectos são fundamentais para compreensão dos trabalhos relacionados, assim como para a metodologia utilizada no estudo investigativo proposto.

Na seção 2.4 foram explicados conceitos de *data stream* que diferem do contexto de mineração de dados tradicional. Como trata de aprendizado sobre bases não-estacionárias, e com limitação de recursos computacionais, como tempo de execução e memória utilizada, como premissas para o *data stream*. Ainda foram explicados os mecanismos de esquecimento, que tratam da atualização do modelo. Da mesma forma, foram expostas as técnicas utilizadas na detecção de *concept drift*, visto que esta característica é utilizada na metodologia do trabalho, e também é relatada em alguns trabalhos relacionados, sendo uma característica muito importante.

Por fim, foi apresentado o algoritmo base denominado Árvore de Hoeffding, sendo este o primeiro a trabalhar com os fundamentos de *data stream*, e tratar da classificação através de árvores de decisão. A partir de evoluções e adaptações deste algoritmo outros foram propostos e são apresentados no Capítulo 3.

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos que apresentam a implementação dos algoritmos de regressão para *data stream* estudados. Ikonomovska, Gama e Džeroski (2011) propõem um algoritmo para regressão on-line, baseado em árvore de decisão, chamado FIMT-DD que trata de *concept drift*. Shaker e Hüllermeier (2012) apresentam o IBLStreams, baseado em *Instance-Based Learning*, que é utilizado tanto para regressão quanto classificação. Já Almeida, Ferreira e Gama (2013) apresentam o AMRules, baseado em regras. Enquanto Barddal, Gomes e Enembreck (2015) propõem um algoritmo para regressão baseado no conceito de redes sociais, chamado de SFNRegressor. Na última seção é apresentada uma discussão sobre os trabalhos de uma forma geral.

3.1 Algoritmo FIMT-DD

O algoritmo FIMT-DD apresentado por Ikonomovska, Gama e Džeroski (2011), se propõe em tratar de forma eficaz de modelos lineares com árvores de decisão para regressão, através de um método de seleção de atributos numéricos dentro de um aprendizado incremental. Além de tratar a detecção de *concept drift* e oferecer mecanismos para adaptação dentro do algoritmo.

Sendo um algoritmo de árvore de decisão e seguindo os princípios de mineração de dados sobre *data stream*, o algoritmo segue a premissa de que não é possível realizar múltiplas leituras sobre uma base de dados não-estacionária. Com isso, coleta as informações relevantes de acordo com os dados e de forma incremental decide sobre qual o critério de divisão da árvore. É considerando também que deve trabalhar com dados de diferentes distribuições para o treinamento.

O algoritmo é baseado na árvore de Hoeffding, onde na divisão de atributos numéricos é utilizado um método com base em busca binária, sendo denominado E-BST (*Extended Binary Search Tree*) que representa uma adaptação do método proposto por Gama, Rocha e Meda (2003). Esse método atua desabilitando divisores considerados ruins, com o objetivo de diminuir a memória exigida. Da mesma forma, controla o crescimento da árvore de acordo com os resultados demonstrados em Ikonomovska, Gama e Džeroski (2011). Para a detecção de *concept drift* o algoritmo apresenta três estratégias: a) aumentar a divisão de novas sub-árvores; b) poda da árvore apenas na região onde foi identificado o *concept drift* e; c) construção de uma sub-árvore alternativa para a região onde foi detectado o *concept drift*.

Nas experimentações realizadas em Ikonomovska, Gama e Džeroski (2011) ocorreram duas abordagens, onde se realizou um comparativo com nove bases sem características de *data stream*, com objetivo de comparar o desempenho em relação aos algoritmos para aprendizado em *batch*, como o CART (BRIEMAN, 1986) e M5Rules (QUINLAN, 1992), e também foram utilizadas características de *data stream*, sendo uma base sintética e outra base real. Com destaque para a base *Airlines*[4], com 116 milhões de linhas, que possui informações sobre voos dos EUA entre 1987 e 2008, onde o objetivo é prever o atraso na chegada dos voos.

[4] *Airlines*: http://stat-computing.org/dataexpo/2009/

Na primeira parte das experimentações foram avaliadas características como acurácia, através do RE[5], RRSE, e também outros indicadores como: CC[6] (coeficiente de correlação), tempo de execução e crescimento da árvore. Foram realizados experimentos com o método de avaliação *cross-validation* (10-*fold*) e também *holdout* (1000k/300k). Onde se obteve variações nos resultados, destacando-se no tempo de execução, onde o FIMT-DD executou em menos tempo para todos os experimentos. Já na acurácia o FIMT-DD obteve melhor desempenho com *holdout*, apresentando variações nos resultados e se mostrando competitivo com os demais algoritmos. Já na segunda parte dos experimentos, com bases *data stream*, foram abordadas estratégias para detecção de *concept drift*, porém não foram avaliados aspectos como tempo de execução e memória utilizada. Assim como não apresentou comparativo com outros algoritmos.

Em Ikonomovska, Gama e Džeroski (2015) são apresentadas diferentes formas de submeter o treinamento com árvores de decisão para regressão com *data stream*, onde são realizadas extensões dos experimentos do primeiro trabalho. Neste trabalho são realizadas comparações com outros algoritmos em diferentes abordagens para indução do treinamento. Utilizando de outras bases com dados reais, como a *City Traffic*[7] (59 mil registros), além da *Airlines* e também a base *Infibiotcs PSP*[8] (257 mil registros).

Nos experimentos realizados são abordadas as métricas de avaliação MSE e *bias-variance*, não apresentando o RMSE. Apesar de realizar comparativos com o crescimento da árvore, memória e tempo de execução, não são apresentados os tempos de execuções para bases com características de *data stream*, apenas a memória utilizada. Sendo que nos experimentos com a base *Airlines* foi utilizada uma amostragem da base (5 milhões de registros), visto que o objetivo era comparar entre as abordagens o comportamento em relação ao MSE e detecção de *concept drift*. A detecção de *concept drift* ocorreu apenas em uma das bases, a *Airlines*, onde é apresentado um gráfico indicando o ponto que ocorre.

3.2 Algoritmo IBLStreams

O algoritmo IBLStreams de Shaker e Hüllermeier (2012) possibilita o trabalho com classificação e regressão. Tem como característica o aprendizado *lazy* permitindo adaptação no tamanho das amostras avaliadas, onde na medida que chegam novas instâncias descarta as primeiras. Diferente de métodos de aprendizado de máquina que induzem um modelo geral a partir dos dados e utilizam esse modelo para obter mais raciocínios, algoritmos IBL simplesmente armazenam os dados, geralmente em memória (SRIMANI e PATIL, 2014). Eles adiam o processamento dos dados até que a predição seja solicitada, propriedade que o qualifica como um método de aprendizado *lazy* (AHA, KIBLER e ALBERT, 1991).

As predições são derivadas da combinação de informações fornecidas pelos dados armazenados. Neste método a função objetivo é aproximada localmente, onde cada combinação pode ser realizada por meio do princípio de estimação do *k*-NN (DASARATHY, 1990), também conhecido como *k*-vizinhos mais próximos.

[5] RE: abreviação de *Relative Error*, uma variação de RRSE, onde não é realizada a raiz quadrada.
[6] CC: Conhecido na área da estatística, responsável por medir a correlação (positiva ou negativa) entre duas (ou mais) variáveis de escala métrica. Fonte: http://mathworld.wolfram.com/CorrelationCoefficient.html
[7] *City Traffic*: http://tunedit.org/challenge/IEEE-ICDM-2010
[8] *Infobiotics PSP*: http://ico2s.org/software/infobiotics.html

Durante o treinamento, o IBLStreams se propõe em otimizar a composição e o tamanho da base de forma autônoma. A cada chegada de um novo exemplo (x_0, y_0), esse é adicionado à base, e a partir disso é verificado se outros exemplos podem ser removidos, ou se identificado como redundantes, são então considerados discrepantes. Para esse fim é definido um conjunto C de exemplos com a vizinhança de x_0 como candidatos. Essa vizinhança é dada pelo k_{cand} vizinhos mais próximos de x_0, determinado de acordo com a distância medida. Os exemplos mais recentes são excluídos da remoção, devido à dificuldade de distinguir potenciais ruídos de possíveis *concept drift*.

Para as experimentações, com regressão, foi realizado um comparativo com o FLEXFIS (LUGHOFER, 2008), que constrói e gerencia um tipo específico de modelo baseado em regras *fuzzy*, chamado de Takagi-Sugeno (TAKAGI e SUGENO, 1985). Como fonte de dados foram utilizadas duas opções: base sintética e uma base real. Na opção sintética foi utilizado o gerador *hyperplane* disponível no MOA, sendo gerado tanto com o *concept drift* quanto sem. Enquanto para base real foi utilizada uma compressão de concreto[9]. Para avaliação comparativa foi verificado o RMSE entre os algoritmos. Os resultados apresentados indicaram oscilações no RMSE, em alguns momentos se mostrou melhor, em outros não. O trabalho apresenta um foco maior sobre a classificação, onde são realizados experimentos adicionais em diferentes perspectivas.

3.3 Algoritmo AMRules

Almeida, Ferreira e Gama (2013) apresentam um algoritmo chamado AMRules (*Adaptative Model Rules*), que é um modelo adaptativo baseado em regras. No qual se propõe tratar de *concept drift* através do teste de Page-Hinkley. Com a proposta de ser o primeiro algoritmo de apenas um passo para aprendizado de regras de regressão para bases de *data stream*. É um algoritmo que promete gerenciar um modelo de decisões em tempo real, conforme a chegada de novos exemplos, respeitando os recursos computacionais disponíveis.

O modelo se caracteriza também por atender regras ordenadas, ou não ordenadas. Para predições com regras ordenadas utiliza a primeira regra que cobrir o exemplo, enquanto para regras não ordenadas atua sobre uma média ponderada de todas as regras que cobrem o exemplo.

Para as experimentações realizadas foram consideradas as duas principais abordagens para *data stream,* métodos *holdout* e *prequential*, sendo que este segundo utilizado apenas em bases com *concept drift*. Foram utilizadas diversas bases de dados com características de bases estacionárias (*UCI Machine Learning Repository* [10]), uma base real (*Airlines*) com características de *data stream* e algumas bases sintéticas. Foram realizadas avaliações comparativas com algoritmos de *data stream*, como FIMT-DD e IBLStreams, além de outros algoritmos considerados estado-da-arte em regressão, como o M5Rules (QUINLAN, 1992) indicado para aprendizado em *batch*.

Os resultados demonstraram que o algoritmo M5Rules foi superior ao AMRules na comparação com bases de dados estacionárias, na avaliação do RMSE e MAE, enquanto na

[9] Compressão de concreto: https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength
[10] *UCI Machine Learning Repository*: https://archive.ics.uci.edu/ml/datasets.html

avaliação do tempo de execução o AMRules foi superior em todas as experimentações. O método de avaliação utilizado foi o *cross-validation* (10-*folds*). Já nas experimentações com a base *Airlines*, comparando com os algoritmos FIMT-DD e IBLStreams, apresentou melhores resultados no RMSE e MAE. É apresentada uma tabela indicando o tempo de aprendizado de cada algoritmo, onde o AMRules executa em todos os experimentos com apenas 1 segundo, comparando com FIMT-DD, que executou em 10 segundos e IBLStreams em 208 segundos. Porém não apresenta explicações sobre o resultado, também não é realizada a avaliação do quanto de memória cada experimento exigiu.

3.4 Algoritmo SFNRegressor

No algoritmo SFNRegressor, Barddal e Enembreck (2013) propõem um algoritmo para regressão baseado no conceito de redes sociais, atendendo à tarefa de detecção de *concept drift*. Com base no modelo de uma rede social livre de escalas (ALBERT e BARABÁSI, 2002), onde um modelo evolutivo da rede mantém um conjunto de indutores de tamanho variável.

A proposta do algoritmo é trabalhar com uma rede, composta por vértices denominados *hubs*. Eles aparecem quando conexões entre os nós são criadas utilizando uma lei de formação, denominada como conexão preferencial. Nesse processo é aplicada uma função exponencial, onde $p(k)$ é a probabilidade de um nó qualquer da rede ser conectado a outros k nós. A definição dessa probabilidade é calculada conforme a seguinte equação:

$$p(k) = \frac{1}{\sum_j erro_j} . \left(\sum_i |erro_i - erro_k| \right) \quad (15)$$

Sendo que $erro$ pode ser parametrizado, podendo ser MAE ou MSE. Com isso é calculada a ponderação de votos dos indutores da rede, onde são utilizadas métricas de centralização, permitindo o uso de diversas formas para calcular a métrica: *degree*, *betweenness*, *closeness*, *eignvector* ou *pagerank*, discutidas por Newman (2010).

Diferencia-se dos demais trabalhos relacionados no sentido que permite a utilização de qualquer algoritmo de aprendizado único, ou seja, possibilita que qualquer algoritmo de regressão seja utilizado como base.

Nas experimentações são realizadas comparações com os algoritmos AddExpert, FIMT-DD, AMRules e IBLStreams, onde é avaliada apenas a métrica de erro RMSE. E apresenta como resultado para algumas bases sintéticas resultados não conclusivos, demonstrando um empate entre os algoritmos. Utiliza também uma base desenvolvida para os experimentos, com característica de *data stream*, chamada Stock Market [11]. O algoritmo utilizando do SFNRegressor apresentou ganhos na comparação com a EMA (*Exponential Moving Average*), apesar de não deixar claro quais os recursos computacionais foram utilizados para esse experimento em especial, e não comparar com outros algoritmos. Como método de avaliação foi utilizado o *prequential*.

[11] Stock Market: foi implementado um extrator de dados da bolsa de valores, que se conecta no Yahoo! e obtém dados de uma ação desejada em um determinado período, de tempo e de acordo com a periodicidade desejada (BARDDAL e ENEMBRECK, 2013).

3.5 Discussão

Todos os trabalhos relatados nas seções anteriores tratam de regressão para *data stream*, todos abordam detecção de *concept drift*, mas nenhum deles realiza estudos mais detalhados a partir de bases com características de *data stream*, com diversos *concept drift*, no máximo duas ocorrências são verificadas nos resultados apresentados. Tão pouco são detalhados os recursos computacionais utilizados, tais como memória e tempo de execução.

Na avaliação dos experimentos e resultados obtidos, é perceptível avaliar que o algoritmo FIMT-DD apesar de apresentar uma série de parâmetros para novos estudos e experimentos, foi avaliado, na sua maioria, com bases de dados não caracterizadas como *data stream*, e sim apenas bases com características estacionárias. Importante destacar dois experimentos, aplicados sob as bases *City Traffic* e *Airlines*, porém os resultados apresentados não demonstram eficiência do FIMT-DD frente a outros algoritmos, as comparações realizadas indicaram que uma abordagem baseada em árvores de decisão, denominada ORTO-A obteve melhor desempenho. Ikonomovska, Gama e Džeroski (2011) apresentaram o algoritmo fazendo comparativos com algoritmos como o CART e M5Rules, que não tratam de *data stream*.

Posteriormente em Ikonomovska, Gama e Džeroski (2015), onde o objetivo é avaliar variações na indução de conjuntos sobre árvores de decisão com dados evolutivos (*data stream*), são estendidas experimentações com FIMT-DD, apresentando comparativos apenas com algoritmos que tratam de *data stream*, porém na maioria dos experimentos as bases utilizadas não se caracterizam como *data stream*. Outro aspecto que se destaca no trabalho é o fato de ser utilizada apenas uma amostragem da base *Airlines*, com a justificativa de que é muito difícil realizar a avaliação sobre um grande volume de dados (116 milhões de linhas), gerando assim um paradoxo, visto que o objetivo é trabalhar com grandes bases de dados e ainda manter um comportamento adequado (BIFET et al., 2010). Deixando assim a dúvida se realmente o algoritmo atende a uma base *data stream* de forma competitiva, além de não abordar a métrica de erro RMSE, que segundo Japkowicz e Shah (2011) é largamente utilizada na avaliação de modelos preditivos.

Na avaliação dos experimentos do IBLStreams não foram encontradas bases reais com características de *data stream*, as experimentações com dados sintéticos também não deixam claro qual a quantidade de registros, assim como em que momentos ocorreram os *concept drift*. A comparação, no caso de regressão, com apenas um algoritmo, o FLEXFIS, deixa a desejar, visto que existem diversos outros algoritmos que se propõem a trabalhar com regressão, como CART e M5Rules, principalmente para bases estacionárias com as características das bases utilizadas, da mesma forma que não são citadas quais as razões para este ter sido escolhido para a comparação.

O IBLStreams é um algoritmo que trabalha com aprendizado do tipo *lazy*, conforme Mitchell (1997), tipicamente caracterizado por carregar todos os dados para a memória para executar o processamento uma única vez. Baseado nesse conceito, seria fundamental apresentar o comportamento e números referentes ao uso de memória, assim como o tempo de execução.

O trabalho sobre o algoritmo AMRules foi o que apresentou experimentações com maior variedade de algoritmos, considerando FIMT-DD e IBLStreams, porém utilizou apenas uma base com característica de *data stream*. A comparação com bases de dados pequenas (menos de 50.000 linhas) não permite afirmar que, em situações com grande volume de dados, o algoritmo também apresentará bons resultados (BIFET et al., 2011). Avaliando a comparação

com o algoritmo M5Rules, que não trata de *data stream*, este ainda assim apresentou melhores resultados considerando o RMSE. O fato de que apenas uma experimentação pode ser considerada para avaliação de um algoritmo de *data stream*, que é com a base *Airlines*. Foram apresentados comparativos com o tempo de execução, porém sem detalhes, onde apresenta apenas um gráfico com o tempo de cada algoritmo (FIMT-DD, IBLStreams e AMRules), mostrando que o AMRules demorou 1 segundo, dado intrigante visto que a base *Airlines* tem 116 milhões de registros e 5 GB de dados. Não apresentou também informações sobre o quanto utilizou de memória.

Já o algoritmo SFNRegressor que apresenta uma abordagem baseada em redes sociais para *data stream*, e oferece a possibilidade de utilizar um algoritmo de aprendizado junto a sua configuração, apresentou resultados não conclusivos. Por trabalhar com uma rede baseada em grafos, onde também precisa controlar o crescimento, não explica o quanto de memória é utilizada nas suas comparações.

Apesar de todos os algoritmos apresentarem as características necessárias para o que é exigido de um método para trabalhar com regressão em *data stream* e ainda atender ao *concept drift*, os resultados apresentados não deixam claro as três características básicas de mineração em *data stream*, conforme Domingos (2015). Também não são padronizadas as métricas de avaliação do erro da predição, alguns utilizam RMSE, outros MAE. A maioria dos trabalhos experimentou apenas uma base considerada grande, utilizando bases típicas de aprendizado em *batch*. Bifet et al. (2015) indicam que o método *prequential* é recomendado para bases com *concept drift* e não-estacionárias, ainda assim alguns trabalhos utilizaram *cross-validation*, o que não representa um problema, mas difere de um ambiente real de aprendizado de regressão sobre *data stream*.

De uma forma geral, esses trabalhos também não apresentaram diversas ocorrências de *concept drift* para avaliar se após um determinado número de adaptações o algoritmo se mantém estável. É importante pontuar, que muitos experimentos utilizaram de bases com poucos registros, embora tenham utilizado bases com características de *data stream*, porém com cenários limitados.

A partir da revisão desses trabalhos, verificou-se que critérios considerados por Domingos (2015) e Gaber (2012) como chaves para mineração de *data stream*, como tempo de execução e memória utilizada, são pouco explorados. Assim como, não há um consenso sobre qual métrica de erro deva ser utilizada para avaliação de regressão. Dessa forma, este trabalho visa apresentar um estudo investigativo de algoritmos de regressão para *data stream,* considerando critérios recomendados na literatura, conforme Bifet et al. (2015), que trazem como premissas básicas a medição de memória e tempo de execução. Além disso, estudar as métricas de avaliação de erro indicados para métodos de regressão, conforme Japkowicz e Shah (2011), utilizando de bases com características de *data stream* que permitam também experimentar o comportamento de cada algoritmo na detecção de *concept drift*, conforme Gama (2010).

4 MATERIAIS E MÉTODOS

Neste capítulo é descrita a metodologia empregada para o estudo investigativo dos algoritmos de regressão para *data stream,* seguindo um fluxo de processos desde a entrada das bases de dados passando pela configuração dos algoritmos e geração dos resultados. Onde são abordados conceitos importantes tratados na seção 2.4, buscando uma análise crítica dos resultados através de diversas experimentações.

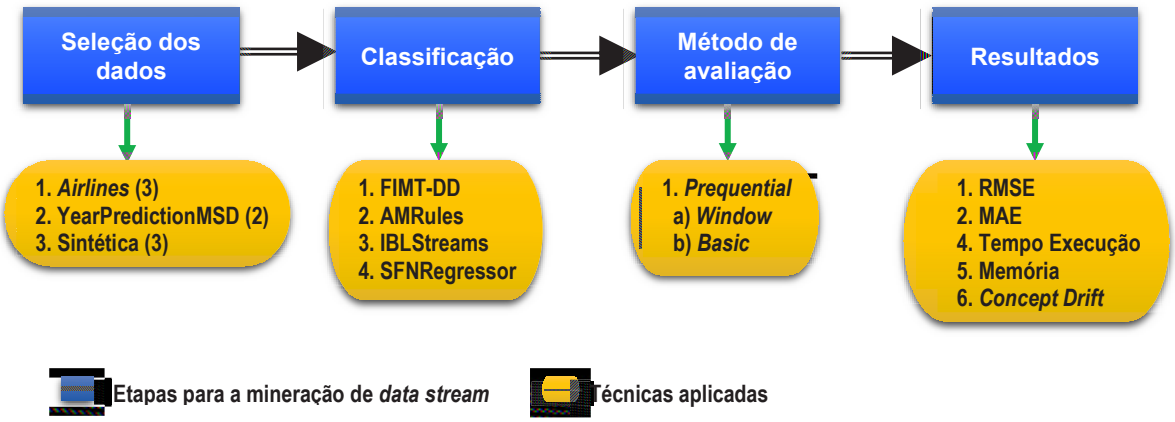
A abordagem adotada é focada em aspectos de avaliação, considerando que tempo e memória têm igual importância ao erro (métrica de avaliação). Para isso na seção 4.1 é descrito o fluxo do processo, apresentando também o ferramental utilizado. As bases de dados utilizadas são apresentadas na seção 4.2, detalhando cada uma e o objetivo buscado sobre cada. Por fim, na seção 4.3, um resumo das configurações utilizadas são descritas, fornecendo informações complementares do ambiente utilizado.

4.1 Metodologia Aplicada

A metodologia aplicada para o estudo investigativo de algoritmos tem como base as premissas estabelecidas para *data stream*, conforme Bifet et al. (2015), Domingos (2015) e Gaber (2012). Onde os recursos como tempo de execução e memória utilizada são limitados. Como a técnica estudada é a regressão, são avaliadas também as métricas de MAE e RMSE, apresentadas na seção 2.3.

O modelo proposto segue um fluxo, conforme ilustrado na Figura 5, onde são utilizadas três bases de dados originais que geram oito bases finais, sendo estas detalhadas na seção 4.2. Tais bases são submetidas ao processo de aprendizado (classificação) executando a tarefa de regressão com os quatro algoritmos estudados, com a aplicação de dois métodos de avaliações que geram como resultados: as métricas de erros para regressão, os recursos computacionais (tempo e memória) e também as ocorrências de *concept drift*.

Figura 5: Bases de dados, etapas e critérios utilizados na avaliação.



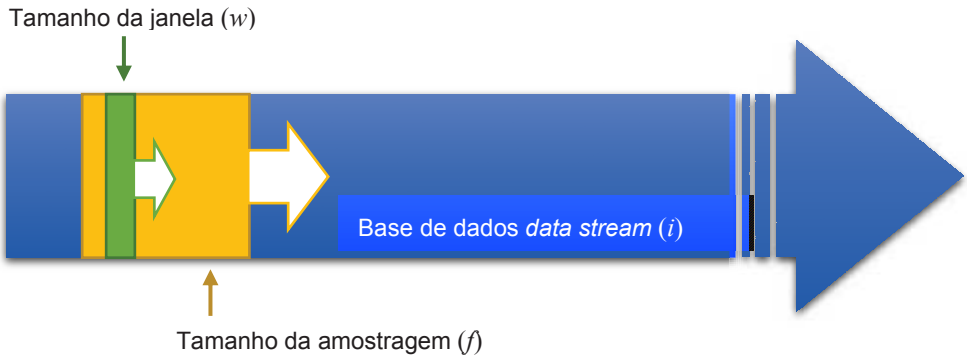
Fonte: Elaborador pelo autor.

Para trabalhar com atributos numéricos, a abordagem mais comum em aprendizado em *batch* é a discretização (DOMINGOS e HULTEN, 2001b), definida na fase de pré-processamento, tipicamente particionando em intervalos de valores. Isso requer uma passagem inicial dos dados antes do aprendizado, bem como operações de ordenações. Porém, pré-

processamento não é uma opção para *data stream* e a ordenação pode ser muito custosa. O intervalo de valores para atributos numéricos também é desconhecido e pode variar em caso de mudança na amostragem (GAMA, SEBASTIÃO e RODRIGUES, 2009). Portanto, os dados utilizados neste trabalho não passam por esta etapa, sendo enviados diretamente para a classificação.

Como método de avaliação é adotada a recomendação de Gama, Sebastião e Rodrigues (2013) e Bifet et al. (2015), onde bases com características não estacionárias e possibilidade de *concept drift* sejam avaliadas com *prequential*. Dentro desse método são adotadas duas opções de configurações: *Basic* e *Window*. Essas variações definem como ocorrerá a evolução do aprendizado sobre os dados submetidos. A Figura 6 ilustra esse processo, onde é preciso especificar algumas configurações, como: o total de instâncias da base de dados que será utilizada (i), o tamanho da amostragem (f) – número de instâncias – que será capturado e mantido em memória para o aprendizado, e por último o tamanho da janela (w), que representa o número utilizado para o aprendizado e atualização do modelo, dentro de uma amostragem são realizadas $\frac{f}{w}$ atualizações, onde a janela desliza sobre o tamanho total da amostragem, e na última atualização é projetado o resultado do teste do modelo gerado. Dentro da opção *window* são empregadas variações de 1.000 à 10.000, enquanto na opção *basic* é considerado o tamanho fixo de uma instância.

Figura 6: Configuração do método de avaliação com *Window*.



Fonte: Elaborador pelo autor.

Após a janela percorrer todo o tamanho da amostragem é gerada uma saída. Foram utilizadas amostragens de 10 mil e 100 mil instâncias nos experimentos. Com isso, após cada execução são coletadas informações como: tempo de execução, memória utilizada, além das medidas de erro. Para avaliação da capacidade de detectar *concept drift* foram utilizadas bases de dados específicas, considerando quatro ocorrências e dados recorrentes.

Dentro dos aspectos investigados também são avaliadas as características específicas de cada algoritmo. No caso do algoritmo FIMT-DD, é analisada a evolução do tamanho da árvore. Já no algoritmo AMRules é analisado o número de regras criadas. Enquanto no IBLStreams é considerado o uso de memória, e no SFNRegressor é avaliado o tamanho da rede.

Para o algoritmo SFNRegressor são empregadas duas abordagens. Como o algoritmo permite utilizar outros algoritmos para gerar o modelo foram selecionados: FIMT-DD e AMRules. Desse modo, os quatro algoritmos selecionados possibilitam cinco diferentes abordagens. Para o SFNRegressor também foram exploradas duas opções para detecção de *drift*: PHT e ADWIN, assim como para o algoritmo AMRules. Enquanto para os experimentos

com o algoritmo FIMT-DD é disponibilizada apenas a opção do PHT, e para IBLStreams é não possível configurar nenhuma técnica.

Como método para coletar os dados dos experimentos são consideradas dez rodadas em cada configuração, base e algoritmo, onde são consideradas as médias de cada configuração, avaliando também o desvio padrão para permitir eliminar possíveis distorções. Como ferramenta principal para execução dos experimentos foi utilizado o *framework* MOA (*Massive Online Analysis*), os dados resultantes dos experimentos foram salvos em uma base de dados estruturada de forma que possibilita a leitura e comparação dos dados de forma comparativa.

4.1.1 *Massive Online Analysis*

O MOA (BIEFT et al., 2010) é um *framework* para aprendizado sobre *data streams*. Apresenta características muito semelhantes ao WEKA[12] desenvolvido para mineração de dados (HALL et al., 2009). Ambos foram implementados em Java, sob o licenciamento GPL[13].

A principal finalidade do MOA foi tratar problemas de *data stream*, disponibilizando algoritmos para classificação, regressão, clusterização, *MultiTarget*, *Outiliers* e *concept drift*. Ele também oferece recursos para geração de dados sintéticos, sendo alguns conhecidos na literatura (BIFET et al., 2009) como: *Random Tree Generator, SEA Concepts Generator, STAGGER Concepts Generator, Rotating Hyperplane*, entre outros. Além de permitir o uso de bases de dados reais, trabalhando com arquivos ARFF[14] e arquivos CSV (*Comma Separated Values*).

Apesar de ter como foco as tarefas de classificação e clusterização, apresenta também recursos para regressão, onde são disponibilizados alguns algoritmos, entre eles o FIMT-DD e o AMRules. A execução da ferramenta pode ser dada tanto através de uma interface gráfica quanto por linha de comando.

Mesmo contendo uma coleção com muitos algoritmos, o MOA como tem seu código-fonte disponibilizado e documentado, permite alterações ou o desenvolvimento de novas implementações. O *framework* está disponibilizado no GitHub[15], onde podem ser feitas cópias ou solicitações de contribuições.

Para uso dos experimentos deste trabalho foram incorporados dois algoritmos, SFNRegressor[16] e IBLStreams[17] que não estão incluídos na versão oficial do *framework*, ambos são disponibilizados por seus autores e referenciados na documentação[18] on-line do MOA como extensões. Para utilização desses algoritmos foi necessário importar as bibliotecas (.jar), como bibliotecas externas.

Na interface visual da ferramenta é possível executar diversos trabalhos, na Figura 7 é apresentada a tela responsável pela tarefa de regressão. Em destaque na região A da figura pode ser visto o resumo das atividades em execução no momento, na região B é visualizado o

---

[12] WEKA: *Waikato Environment for Knowledge Analysis.*

[13] GPL: do inglês *General Public License*, é uma designação para software livre.

[14] ARFF: *Attribute-Relation File Format*, é um formato de arquivo padrão no WEKA e MOA, que contém uma lista de instâncias e incluindo a definição dos atributos no início.
Fonte: http://www.cs.waikato.ac.nz/ml/weka/arff.html

[15] Disponível em: <https://github.com/Waikato/moa>

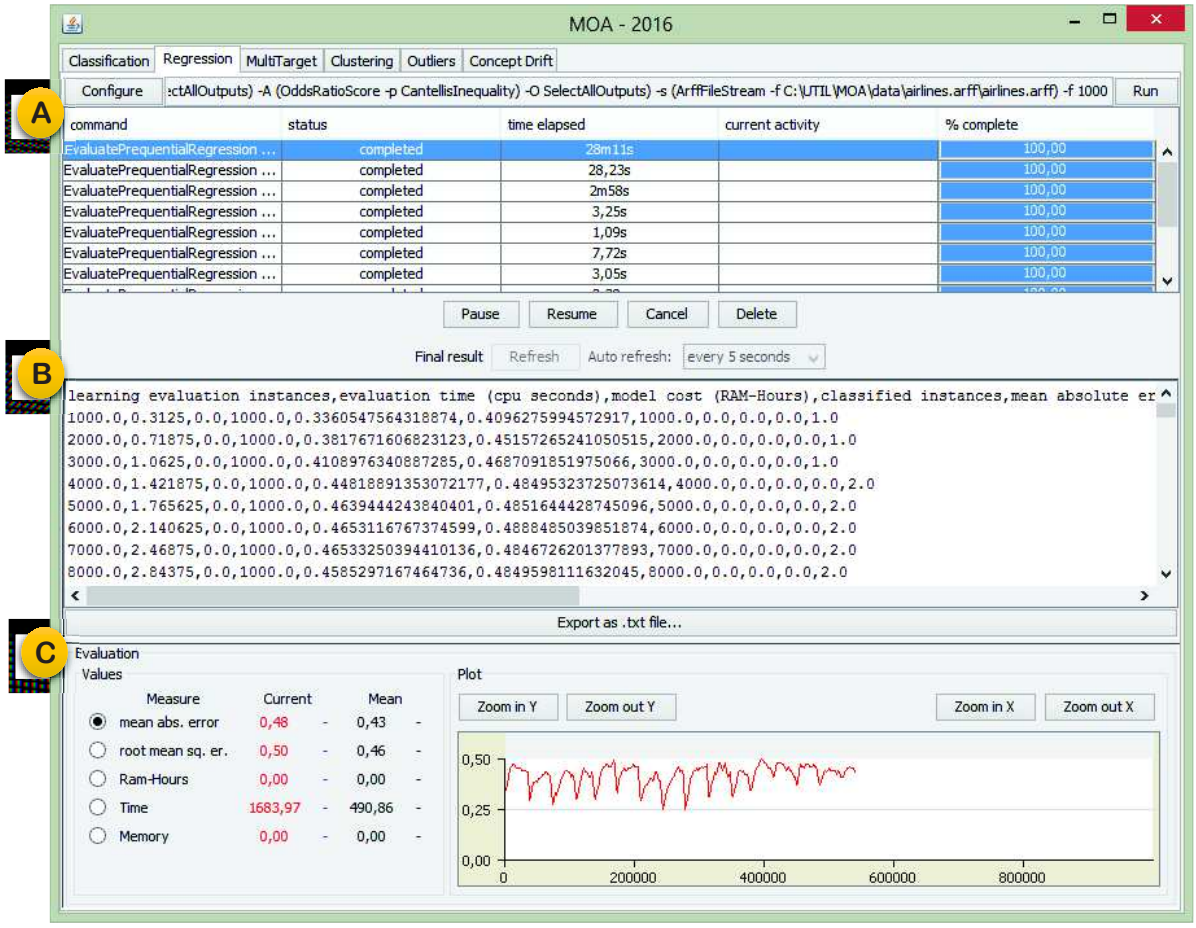
[16] SFNRegressor: https://sites.google.com/site/moasocialbasedalgorithms/home

[17] IBLStreams: http://www.uni-marburg.de/fb12/kebi/research/software/iblstreams

[18] Documentação do MOA: http://moa.cms.waikato.ac.nz/moa-extensions/

resultado das tarefas, em formato de texto, dados esses que são utilizados na avaliação deste trabalho. Já na região C são exibidos os indicadores de erro, tempo e memória projetando um gráfico à direita.

Figura 7: Interface gráfica do MOA.



Fonte: Elaborado pelo autor.

Os resultados gerados, além de impressos na tela principal da ferramenta, também podem ser salvos em arquivos de texto. Assim como um arquivo com o resultado da predição executada, permitindo avaliar o valor esperado e valor predito. Essas configurações podem ser parametrizadas tanto na interface gráfica quanto por linha de comando.

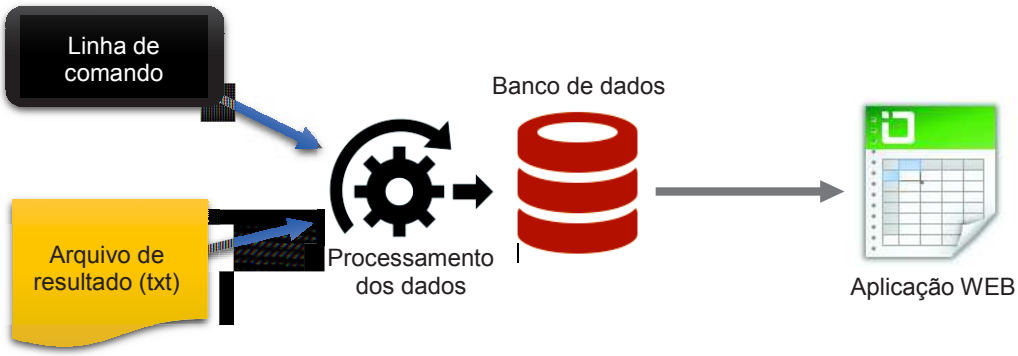
É importante destacar que, apesar da proposta deste *framework* de trabalhar com *Big Data Stream*, o mesmo não trabalha com paralelismo. Como alternativa para trabalhar em ambientes distribuídos há outro *framework*, o SAMOA (*Scalable Advanced Massive Online Analysis*) de Morales e Bifet (2015), que não é abordado neste trabalho, visto que ainda não apresenta uma versão estável, sendo limitada a um sistema operacional específico. Entretanto, é importante considerá-lo para trabalhos futuros.

4.1.2 *Data Stream Regression Results Analysis* (DSRRA)

Uma aplicação, denominada *Data Stream Regression Results Analysis*, foi desenvolvida pelo autor desta dissertação para oferecer apoio na avaliação dos resultados, onde o objetivo principal é disponibilizar uma leitura simplificada dos resultados dos experimentos gerados

pelo MOA. Sendo assim, para cada experimento executado foram salvos todos os parâmetros definidos e o arquivo com o resultado da classificação. A Figura 8 resume a arquitetura da aplicação, onde os diferentes resultados convergem para uma estrutura genérica de atributos, possibilitando análises comparativas.

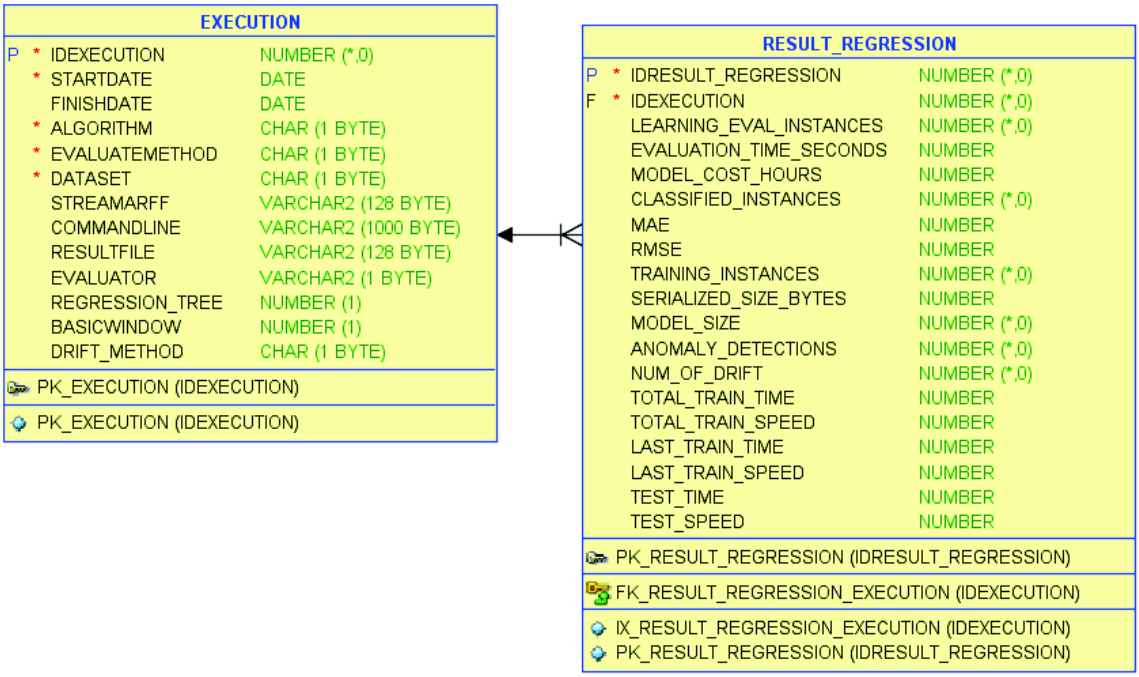
Figura 8: Arquitetura da ferramenta de apoio DSRRA.



Fonte: Elaborador pelo autor.

A partir da entrada da parametrização utilizada (linha de comando) e o arquivo com os resultados, as informações são processadas e armazenadas em um banco de dados Oracle 11g XE[19]. Sendo organizadas em duas tabelas relacionais, conforme a Figura 9. A aplicação de interface *web* foi desenvolvida com o Oracle Apex [20] (*Application Express*), que é uma ferramenta de desenvolvimento rápido para *web*, utilizando as linguagens PL/SQL [21] e Javascript.

Figura 9: Diagrama entidade relacionamento das duas principais tabelas da aplicação.

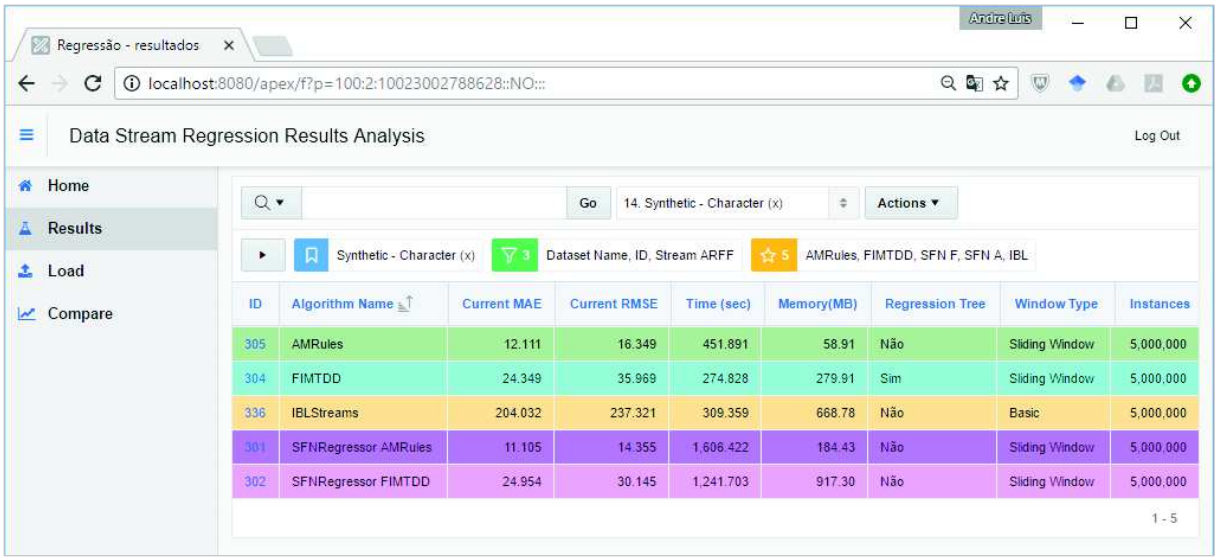


Fonte: Elaborador pelo autor.

[19] Oracle 11g XE: banco de dados relacional, *freeware*. Disponível em: http://www.oracle.com/.
[20] Oracle Apex: https://apex.oracle.com
[21] PL/SQL: *Procedural Language/Structured Query Language*, é uma extensão da linguagem SQL da Oracle.

Como resultado do processo de importação, a ferramenta disponibiliza uma consulta de forma dinâmica, oferecendo opções de filtros, ordenação e marcação de textos. Na Figura 10 é ilustrado um exemplo da visualização final dos experimentos para uma base de dados, onde está filtrado para exibir um resultado de cada algoritmo. O critério adotado para exibição considera o menor erro (MAE) e também o menor tempo e memória.

Figura 10: Tela de consulta dos experimentos.



Fonte: Elaborador pelo autor.

É importante destacar que esta aplicação não é o foco deste trabalho. No Apêndice A são disponibilizadas mais informações dessa ferramenta, que está publicada no GitHub[22] para que possa receber contribuições da comunidade científica.

4.2 Bases de Dados

Com o objetivo de utilizar bases de dados com características de *data stream*, foram selecionadas três fontes originais, considerando o número mínimo de 500 mil instâncias. Um total de oito bases foram geradas a partir das fontes principais, cada uma com diferentes propósitos, detalhados a seguir.

4.2.1 Airlines

Esta base de dados foi apresentada na competição 2009 *Data Expo*[23], sendo usada também por Ikonomovska, Gama e Džeroski (2011) e Almeida, Ferreira e Gama (2013). Representa uma base não estacionária, do mundo real, com dados de voos comerciais dos EUA, do período de outubro de 1987 à abril de 2008. Para este trabalho foi utilizado o intervalo de 1988 à 2008. A base tem dados das partidas e chegadas dos voos, como horário de saída, chegada, local de origem e destino, totalizando 14 atributos. A informação a ser predita nesta

[22] DSRRA: disponível no endereço https://github.com/nunes-andre/dsrra/
[23] http://stat-computing.org/dataexpo/2009/

base é o tempo de atraso de voo. O volume de dados é aproximadamente 113 milhões de instâncias.

O propósito desta base é avaliar o comportamento dos algoritmos com grande volume de dados, o tempo de execução e também as métricas de erro. A base completa dos dados é chamada neste trabalho de **Airlines** apenas, outras duas variações foram geradas: uma base com dados apenas do último ano (2008) – **AirlinesL1Y**, com 5,8 milhões de instâncias – e outra com os dois últimos anos (2007-2008) – **AirlinesL2Y**, com 13 milhões de instâncias. A intenção desta abordagem com amostragens menores é avaliar o poder de generalização e capacidade de aprendizado apenas com dados mais recentes na comparação com os resultados da base completa.

4.2.2 YearPredictionMSD

Esta base de dados é parte de uma coleção de características de músicas do repositório *Million Song Dataset*[24] (BERTIN-MAHIEUX et al. 2011), sendo disponibilizada em Lichman (2013). Consiste na predição do ano de lançamento da música a partir de características do aúdio, são músicas lançadas entre 1922 e 2011. Sendo 515 mil registros, composta por 90 atributos, todos de tipo numérico, 12 indicando médias de timbres e 78 a convariância dos timbres.

O objetivo desta base é avaliar a capacidade dos algoritmos em trabalhar com um grande número de atributos. Adicionalmente, uma nova base foi gerada a partir da duplicação da original. Com isso, possibilitando uma avaliação dos resultados e comportamento onde os padrões já aprendidos anteriormente são submetidos novamente de forma recorrente, buscando assim, que na segunda passagem os erros sejam inferiores a primeira.

4.2.3 Friedman

Uma solução com dados artificiais é adotada para a avaliação de ocorrências de *concept drift*, devido a dificuldade em controlar e identificar tais características em bases reais. Com a clássica fórmula de Friedman (1991), foi gerado uma base de dados com 10 atributos numéricos (x_d) de valores independentes, considerando apenas cinco para a geração do valor a ser predito (y). Entretanto, algumas variações são aplicadas para geração dos dados. A fórmula original é definida na equação (16), que gera quatro diferentes configurações de intervalo de valores, cada uma definindo um conceito (Cn), com 1 milhão de instâncias. A Tabela 4 mostra a configuração de intervalo de valores utilizada para cada conceito (*concept*).

$$y = 10\sin(\pi x_1 x_2) + 20\,(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1) \qquad (16)$$

As regras definidas para a configuração de cada conceito são definidas como:

- C1: a fórmula original é aplicada;
- C2: a fórmula original é alterada, substituindo x_3 por x_8, gerando variação na seleção de atributos;
- C3: a fórmula original é alterada, substituindo x_2 e x_3 por x_7 e x_8 respectivamente, também gerando variação na seleção de atributos;

[24] http://labrosa.ee.columbia.edu/millionsong/

- C4: a fórmula original é aplicada, com alterações apenas no intervalo de valores mínimos e máximos.

Com isso, quatro milhões de instâncias são geradas. Com a replicação de C1 para o final da base, gerando um C5, então cinco milhões de instâncias são geradas, denominando **Fried** como a primeira base sintética deste trabalho. O propósito desta estratégia é proporcionar num ambiente controlado, a avaliação do comportamento dos algoritmos expostos à necessidade de várias adaptações no modelo, avaliando a capacidade de generalização e aprendizado. É esperado que o erro para C5 seja igual ou inferior à C1.

Tabela 4: Configuração dos intervalos de valores para cada conceito.

	C1			C2			C3			C4		
Atributo	**Mín.**	**Máx.**	**Decimais**	**Mín.**	**Máx.**	**Decimais**	**Mín.**	**Máx.**	**Decimais**	**Mín.**	**Máx.**	**Decimais**
x_1	1	2	1	1	2	1	5	7	1	1	3	1
x_2	0	5	2	0	5	2	0	5	2	1	5	1
x_3	1	5	1	5	10	1	5	9	1	5	9	1
x_4	1	10	0	1	10	0	1	10	0	1	10	0
x_5	-2	3	1	-2	3	1	-2	3	1	-2	3	1
x_6	-10	20	2	-10	20	2	-10	20	2	-10	20	2
x_7	1	5	2	1	5	2	4	6	1	4	6	1
x_8	0	10	2	1	5	2	1	5	1	1	5	1
x_9	-2	12	2	-2	12	2	-2	12	2	-2	12	2
x_{10}	1	10	2	1	10	2	1	10	2	1	10	2

Fonte: Elaborador pelo autor.

A partir desta base sintética inicial outras duas abordagens são geradas, com objetivo de investigar outros aspectos. Na primeira adaptação, o valor numérico do atributo x_4 foi substituído por um caractere, de acordo com a seguinte condição: 1=A, 2=B, 3=C, ... até 10=J. Demais valores permanecem iguais, reforçando que este atributo é utilizado na geração do y para todos os conceitos. Nesse trabalho esta base é chamada de **FriedChar** (*Friedman with Character*). A segunda adaptação gerada é a remoção do atributo x_4 da base de dados, denominando a base **FriedWOx4** (*Friedman Without* x_4). O objetivo dessas adaptações é verificar o comportamento na comparação com as demais abordagens, principalmente a dependência do atributo alvo (y) do x_4.

4.3 Configurações das Experimentações

Um resumo das configurações utilizadas neste trabalho é apresentado nesta seção. Em geral, a maioria das configurações são similares entre os algoritmos, entretanto existem algumas particularidades. Em relação a tamanho da amostragem utilizada, para todos os experimentos foi utilizado 100 mil, a exceção das bases *YearPredictionMSD*, que também foram avaliadas com amostragens de 10 mil. Em razão do menor número de instâncias, visto que o resultado da classificação é dado após percorrer o tamanho da amostragem, então para não ter apenas 5 saídas de resultados na base original, foram utilizadas outras configurações. O resumo das características de cada base utilizada é exibida na Tabela 5.

Em relação à parametrização do número mínimo de instâncias (*grace period*), a definição foi mantida padrão, para os resultados apresentados, $N_{min} = 200$. Duas opções foram utilizadas para o algoritmo FIMT-DD, abordagens definindo a geração de árvore de regressão (RegressionTree=true) e árvore de decisão tradicional (RegressionTree=false), este parâmetro altera o formato da árvore gerado durante o aprendizado na geração do modelo.

Para as configurações de detecção de *concept drift* foram utilizados valores para $\lambda = 50$, para as técnicas PHT e ADWIN. Para o número de instâncias foram utilizados dois valores 30 e 50, conforme os trabalhos relacionados. Também foram realizados experimentos com variações nessa configuração, entretanto por não apresentarem bons resultados não são detalhados no Capítulo 5, porém os resultados obtidos são expostos no Apêndice B, assim como a configuração empregada.

Tabela 5: Resumo das bases de dados utilizadas.

Nome da base	Atributos	Instâncias
Airlines	14	113 M
AirlinesL1Y	14	5.8 M
AirlinesL2Y	14	13 M
YearPredictionMSD	90	515 K
YearPredictionMSD Duplicada	90	1 M
Fried	10	5 M
FriedChar	10	5 M
FriedWOx4	9	5 M

Fonte: Elaborador pelo autor.

Todas as experimentações foram executadas em um Intel Core 5i-6200, CPU 2.4GHz com 8GB de RAM, disco SSD, sistema operacional Windows 10 Pro. Para todos os experimentos foi utilizado o *framework* do MOA na versão 2016.04, exceto para o experimentos com o algoritmo IBLStreams, onde a versão disponibilizada da biblioteca não é compatível, sendo necessário o uso da versão 2011.10.

5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos a partir de avaliações empíricas, analisando aspectos gerais de *data stream* focados na tarefa de regressão. Considerando a métrica de erro, tempo de execução e memória utilizada, sem especificar qual é mais relevante.

Na seção 5.1 são apresentados os resultados com a maior base de dados deste trabalho, Airlines, expondo o comportamento de cada algoritmo durante o processo de aprendizado, principalmente no critério tempo de execução. A seção 5.2 demonstra os resultados com a base de dados YearPredictionMSD, com o foco na avaliação do poder de generalização através de dados recorrentes. Na seção 5.3 são apresentados os resultados explorando *concept drift*, considerando o poder de adaptação de cada algoritmo, assim como um comparativo entre os principais métodos de detecção de *concept drift*. Por fim, na seção 5.4 são apresentadas as considerações dos resultados obtidos, expondo o desempenho de cada algoritmo.

5.1 Airlines

Na avaliação dos atributos fornecidos pela base é possível identificar que para uma precisão maior na predição seria necessário mais informações, tais como condições climáticas, número de passageiros, dados das aeronaves, aeroportos, entre outras informações. Entretanto, este trabalho visa estudar outras métricas, que vão além do erro apenas. Sendo assim, apesar de de buscar avaliar o tempo de execução, principalmente, foi aplicado um cálculo de correlação entre os atributos sobre uma amostragem dos dados, considerando apenas o último ano, o resultado é exibido no Quadro 2.

Quadro 2: Correlação dos atributos da base AirlinesL1Y.

	Year	Month	DayofMonth	DayofWeek	CRSDepTime	CRSArrTime	FlightNum	ActualElapsedTime	Distance	ArrDelay
Year	1,00									
Month		1,00	-	-	- 0,01	-	-	- 0,02	-	**- 0,08**
DayofMonth		-	1,00	- 0,01	-	-	-	-	-	**- 0,02**
DayofWeek		-	- 0,01	1,00	-	-	-	0,01	0,02	0,01
CRSDepTime		- 0,01	-	-	1,00	0,78	- 0,01	- 0,01	- 0,01	**0,12**
CRSArrTime		-	-	-	0,78	1,00	- 0,03	0,05	0,04	**0,12**
FlightNum		-	-	-	- 0,01	- 0,03	1,00	- 0,32	- 0,35	0,01
ActualElapsedTime		- 0,02	-	0,01	- 0,01	0,05	- 0,32	1,00	0,97	**0,10**
Distance		-	-	0,02	- 0,01	0,04	- 0,35	0,97	1,00	0,01
ArrDelay		**- 0,08**	**- 0,02**	0,01	**0,12**	**0,12**	0,01	**0,10**	0,01	1,00

Fonte: Elaborador pelo autor.

O resultado da correlação indicou que alguns atributos não apresentaram relação, enquanto atributos como horário de partida (CRSDepTime) e chegada (CRSArrTime), juntamente com tempo de voo (ActualElapsedTime) e dia (DayofMonth) e mês (Month) também apresentaram alguma relação. Esta análise em bases caracterizadas por *concept drift* pode apresentar resultados distorcidos considerando o cálculo de correlação. Conforme Fan, Han e Liu (2014), a distribuição tende a mudar ao longo do tempo em bases não estacionárias.

Na tarefa de aprendizado a análise com a base Airlines apresentou resultados relevantes, principalmente relacionados com o tempo de execução. Considerando a base de dados completa, a diferença de tempo é grande, com o menor tempo executando em 12 minutos e a mais demorada em 6,5 horas, enquanto o erro não apresentou grande diferença. A Tabela 6 apresenta o resumo dos resultados obtidos com os experimentos realizados. Avaliando os experimentos com as bases AirlinesL1Y e AirlinesL2Y é possível verificar que os resultados apresentados são muito próximos, na avaliação do erro, comprovando que apenas os dados mais recentes são importantes para o aprendizado.

O algoritmo SFNRegressor com AMRules apresentou o menor erro (MAE) nos experimentos com as três bases. Entretanto, também exigiu mais tempo, demandando 20 vezes mais tempo do que o mais rápido, que foi o FIMT-DD. Outra constatação relevante fica com o algoritmo IBLStreams que exigiu mais memória que os demais, além de não suportar a base completa, gerando erro por falta de memória.

Tabela 6: Resumo dos resultados obtidos com as bases de *Airlines*.

	AirlinesL1Y					AirlinesL2Y					Airlines Completa				
Algoritmo	**MAE**	**RMSE**	**Tempo (seg)**	**Mem. (MB)**	**Média MAE**	**MAE**	**RMSE**	**Tempo (seg)**	**Mem. (MB)**	**Média MAE**	**MAE**	**RMSE**	**Tempo (seg)**	**Mem. (MB)**	**Média MAE**
AMRules	11,39	**17,72**	432,72	25,73	18,23	11,69	17,79	1036,72	38,49	20,55	12,54	18,90	24924,27	125,26	14,95
FIMT-DD	11,65	17,87	**58,66**	**8,29**	19,85	12,02	18,95	**109,41**	**10,04**	21,74	11,26	17,74	**725,17**	**10,08**	16,13
IBLStreams	13,64	18,76	202,92	917,08	20,23	13,64	18,76	2202,30	2036,87	21,23	-	-	-	-	-
SFN AMRules	**11,18**	17,85	1.487,36	95,30	**17,73**	**10,92**	**17,68**	2858,39	94,89	**20,07**	**10,97**	17,88	23293,98	76,49	**14,85**
SFN FIMT-DD	11,32	17,86	899,30	147,24	18,41	11,24	**17,61**	803,38	97,18	21,37	11,29	**17,63**	4898,53	93,19	16,01

Fonte: Elaborador pelo autor.

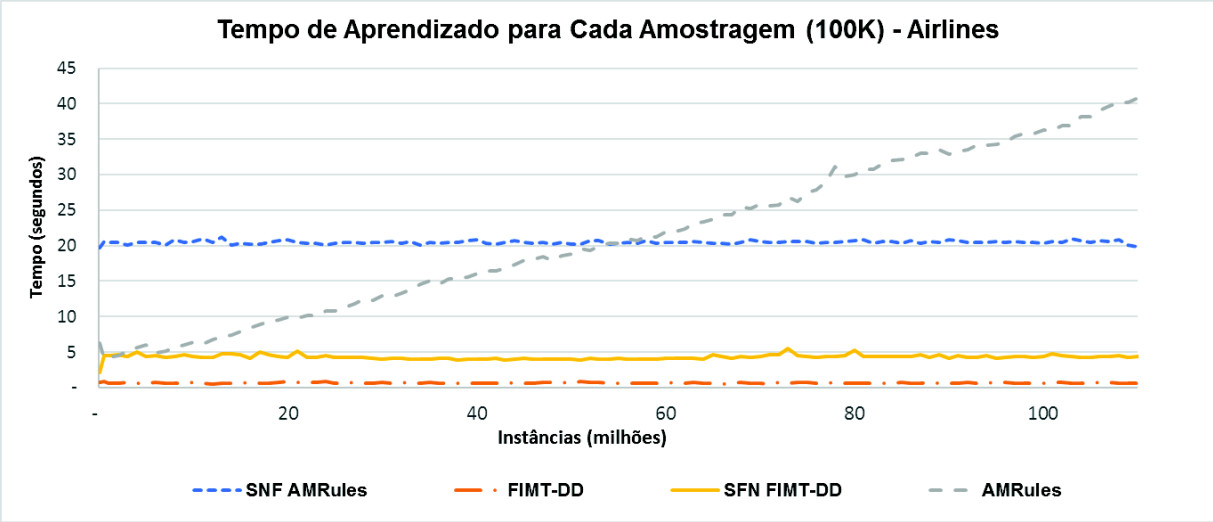
Conforme as configurações utilizadas para estes experimentos, as saídas são geradas a cada amostragem de 100.000 instâncias, as colunas MAE e RMSE (Tabela 6) indicam o resultado da última amostragem, enquanto a coluna Média MAE indica a média de todas as saídas durante o processo de aprendizado. Não é dada muita importância para a média porque no início de cada treinamento o erro geralmente é alto, diminuindo ao longo do aprendizado.

Seguindo a premissa de que em ambientes dinâmicos não há tempo para a etapa de pré-processamento, foi avaliado a capacidade dos algoritmos na seleção de atributos, utilizando a base de dados AirlinesL1Y. Para isso foram realizados experimentos incluindo atributos com valores independentes sem relação com o atributo alvo, gerando um ruído nos dados. O resultado demonstrou que o erro não foi impactado, apresentando o mesmo valor que a base sem atributos não relacionados, ou seja, a seleção de atributos dos algoritmos foi eficiente.

A ocorrência de degradação, em relação ao tempo, durante o treinamento de um grande conjunto de dados foi investigada. Os resultados demonstraram que o tempo de execução para cada amostragem é constante na maioria dos algoritmos, a exceção foi o AMRules, onde o tempo teve um crescimento ao longo da base, onde na primeira amostragem executou em 6 segundos e precisou de 41 segundos para a última amostragem. O motivo desse aumento de tempo está vinculado a atualização do modelo, que está sendo incremental. A Figura 11

apresenta um gráfico, demonstrando essa análise. Pode-se evidenciar que o SFNRegressor utilizando AMRules consegue estabilizar o tempo, mantendo-se constante até o final da base, apesar de apresentar um tempo muito maior para cada amostragem.

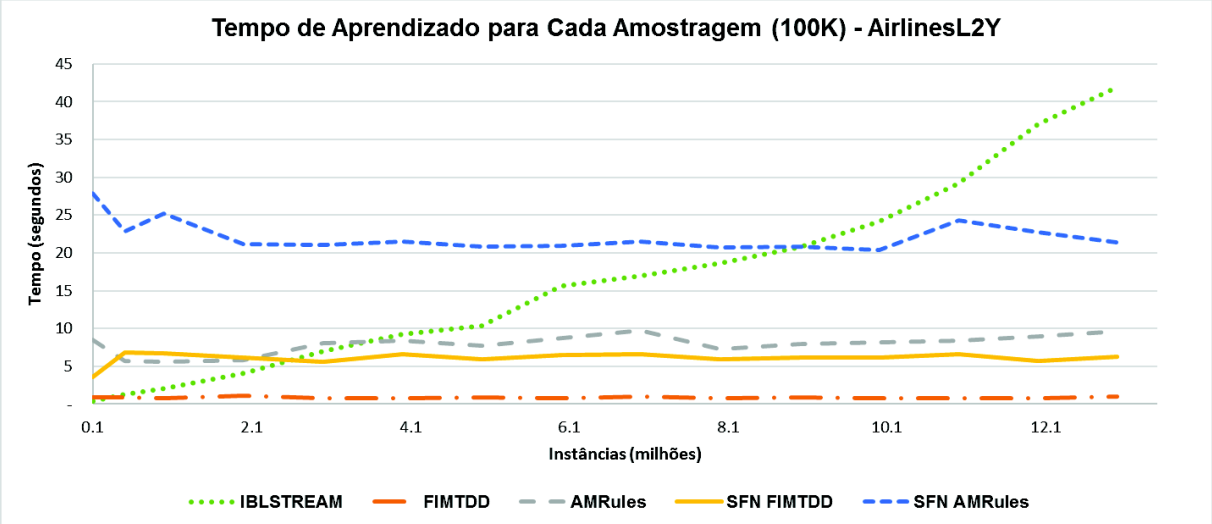
Figura 11: Tempo de aprendizado para cada amostragem ao longo da base Airlines.



Fonte: Elaborador pelo autor.

Dando sequência à investigação, o comportamento do algoritmo IBLStreams também foi explorado através dos experimentos com a base AirlinesL2Y. A Figura 12 ilustra os resultados, onde o tempo do IBLStreams apresenta um grande crescimento. Enquanto os demais algoritmos não apresentam significativa diferença. Embora este experimento utilize um grande volume de dados – 13 milhões de instâncias – o resultado final não mostra claramente o mesmo crescimento na comparação com a análise para a base completa (Airlines).

Figura 12: Tempo de aprendizado para cada amostragem ao longo da base AirlinesL2Y.



Fonte: Elaborador pelo autor.

Esta avaliação comprova a importância de utilizar bases com um grande volume de dados, uma vez que uma das principais características de *data stream* é atuar sem limite de tamanho, considerando bases não estacionárias. Experimentos com pequenas amostragens de

dados podem esconder problemas reais e fornecer interpretações equivocadas. A Figura 12 mostra que o algoritmo FIMT-DD é o mais estável, considerando o tempo de execução, não excedendo 5 segundos para o aprendizado. A Tabela 7 apresenta a estatística dos experimentos com as bases Airlines e AirlinesL2Y.

Tabela 7: Estatísticas do tempo (segundos) de aprendizado utilizando as bases Airlines e AirlinesL2Y.

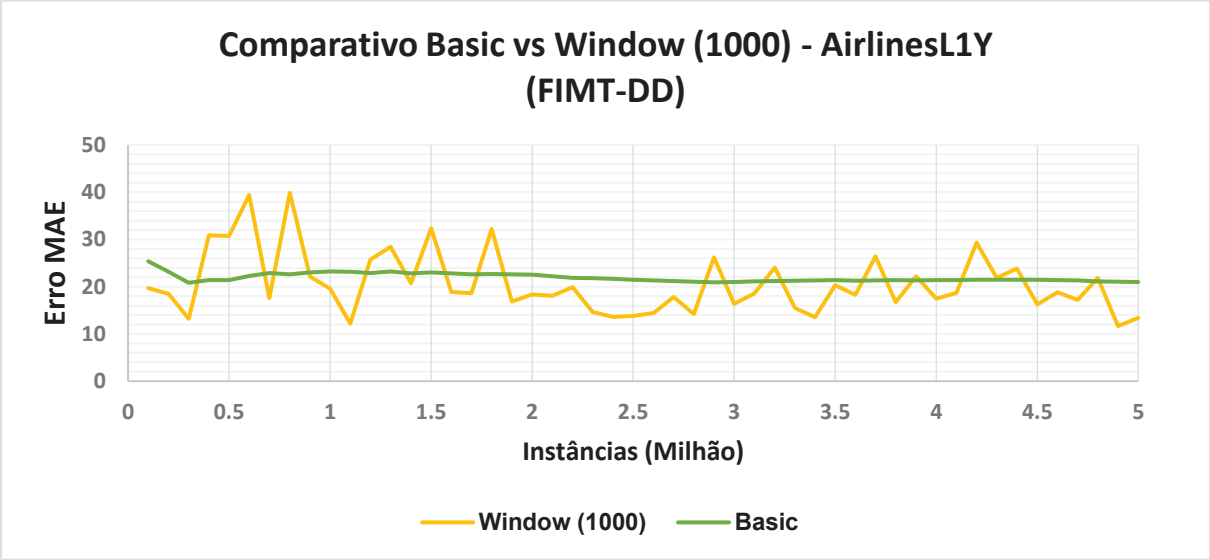
Algoritmo	Média		Mínimo		Máximo		Desvio Padrão	
Base	AirlinesL2Y	Airlines	AirlinesL2Y	Airlines	AirlinesL2Y	Airlines	AirlinesL2Y	Airlines
AMRules	7,91	21,90	5,31	3,98	10,25	42,77	1,16	11,07
FIMT-DD	**0,84**	**0,64**	**0,64**	**0,48**	**1,23**	**1,16**	**0,10**	**0,09**
IBLStreams	16,81	-	0,38	-	41,95	-	11,54	-
SFN AMRules	21,82	20,47	17,91	16,88	31,22	23,59	1,89	0,34
SFN FIMT-DD	6,13	4,30	3,61	2,09	7,39	5,92	0,43	0,28

Fonte: Elaborador pelo autor.

O algoritmo AMRules teve o maior desvio padrão para a base completa (Airlines), já no contexto da AirlinesL2Y o algoritmo IBLStreams teve o maior desvio. A memória utilizada não apresentou maiores problemas nos cenários avaliados. Experimentos com o SFNRegressor utilizando AMRules obtiveram menores erros (MAE), entretanto, o tempo necessário para completar o processo foi muito maior do que as abordagens com FIMT-DD e AMRules. Embora o FIMT-DD tenha obtido os melhores resultados é preciso salientar que o parâmetro que define o tipo de árvore de decisão utilizada, RegressionTree, foi selecionado.

No comparativo entre as opções *Basic* e *Window* foi possível constatar que a primeira opção é mais estável, enquanto a segunda opção com tamanhos de janelas de 1.000 apresenta maiores variações. Na Figura 13 é apresentado um comparativo entre as duas opções com a AirlinesL1Y.

Figura 13: Comparativo do erro MAE entre *Basic* e *Window* (1000) com o algoritmo FIMT-DD.



Fonte: Elaborador pelo autor.

Na avaliação detalhada dos resultados obtidos também foi possível identificar que quanto maior o tamanho da janela, menor é o tempo de execução. Porém, apresenta maiores variações, ainda assim resultando em menores erros para a maioria das amostragens. Os outros

algoritmos apresentaram resultados semelhantes, sendo que na maioria das execuções a opção com *Window* apresentou menores erros e menores tempos de execução.

5.2 YearPredictionMSD

Nos experimentos com YearPredictionMSD foi explorado o comportamento com a base original e duplicada, com o objetivo de avaliar a classificação com dados recorrentes. Os mecanismos de esquecimentos são importantes no contexto de adaptação de modelos, principalmente para atacar o *concept drift*, entretanto no mundo real existem cenários com dados recorrentes e séries temporais. Os resultados mostram que o algoritmo AMRules obteve o menor erro e também o menor tempo, a Tabela 8 resume os resultados dos experimentos.

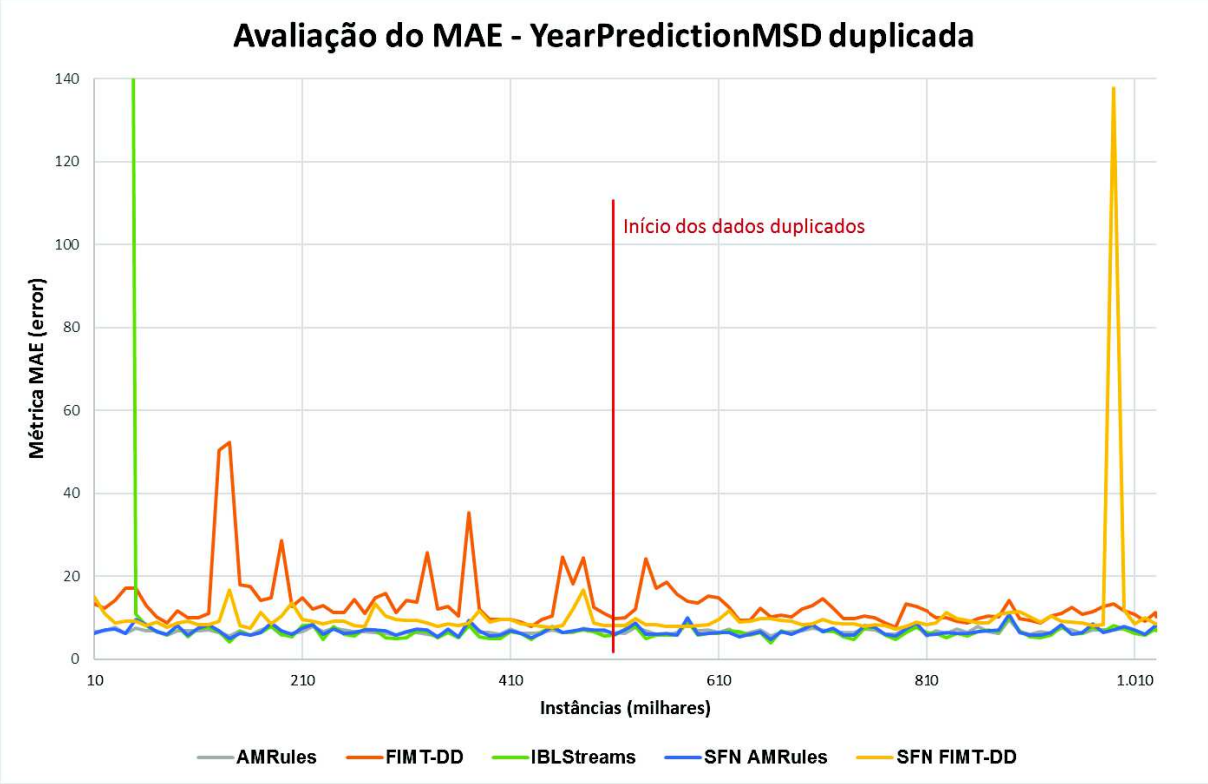
Tabela 8: Resultados dos experimentos com a base YearPredictionMSD.

Algoritmo	YearPredictionMSD Original					YearPredictionMSD Duplicada				
	MAE	RMSE	Tempo (seg)	Memória (MB)	Média MAE	MAE	RMSE	Tempo (seg)	Memória (MB)	Média MAE
AMRules	**6,67**	**9,39**	**141,41**	30,12	**6,73**	**6,65**	**9,37**	**346,06**	51,06	**6,70**
FIMT-DD	9,31	13,88	302,58	320,84	11,89	10,21	14,83	731,06	545,33	13,59
IBLStreams	6,86	10,99	2676,39	**10,46**	44,70	6,93	11,27	5659,52	**18,28**	60,93
SFN AMRules	8,19	12,69	569,56	86,70	7,03	8,10	12,71	716,47	93,93	6,83
SFN FIMT-DD	6,70	9,49	582,31	66,56	**6,73**	8,54	11,74	2953,13	1060,34	10,56

Fonte: Elaborador pelo autor.

O algoritmo IBLStreams apresentou erros com valores muito altos nas quatro primeiras amostragens, com valores como 1997, 1137 e 500, a partir da quinta amostragem apresentou resultados próximos dos demais algoritmos. Na tentativa de eliminar essas diferenças foram realizadas diversas abordagens na configuração, sendo possível eliminar esse erro inicial, iniciando com valores de erros menores, entretanto o tempo de execução foi muito superior, passando de 44 minutos para 2 horas. O detalhamento da configuração utilizada pode ser visto no Apêndice C. A Figura 14 apresenta o resultado da avaliação da base YearPredictionMSD duplicada destacando o ponto que inicia a duplicação dos dados.

Figura 14: Avaliação do erro (MAE) utilizando a base YearPredictionMSD duplicada.

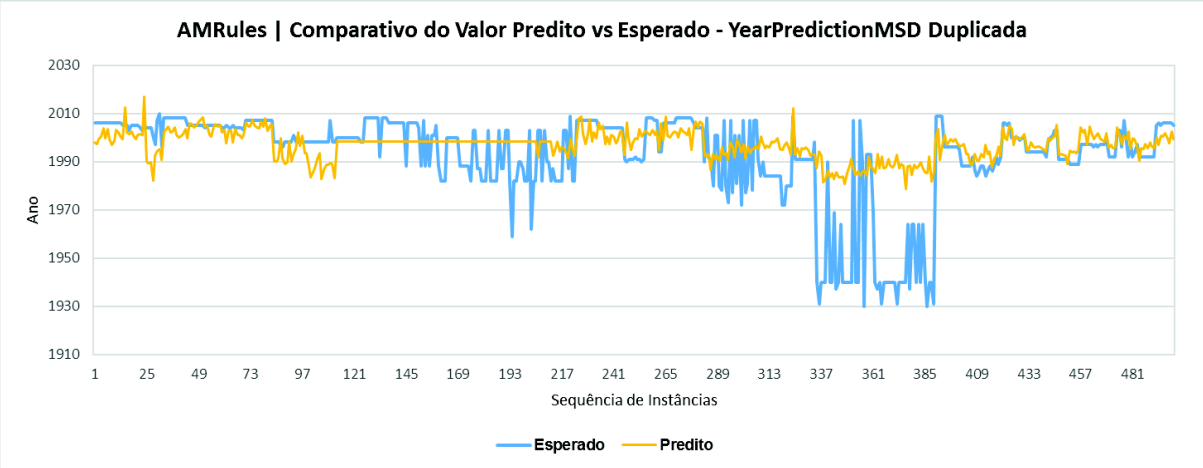


Fonte: Elaborador pelo autor.

Em geral, após o ponto inicial dos dados duplicados o erro diminuiu, apesar de que o tamanho da base não seja tão grande (1 milhão), o número de atributos é considerável (90) e os valores são todos numéricos com cinco decimais, incluindo valores negativos. Estas características tornam o processo de aprendizado mais difícil e justifica mais tempo para execução. Outro aspecto importante de ressaltar dessa base se refere aos dados não estarem ordenados, permitindo avaliar aspectos importantes de *data stream* sem a atuação do pré-processamento.

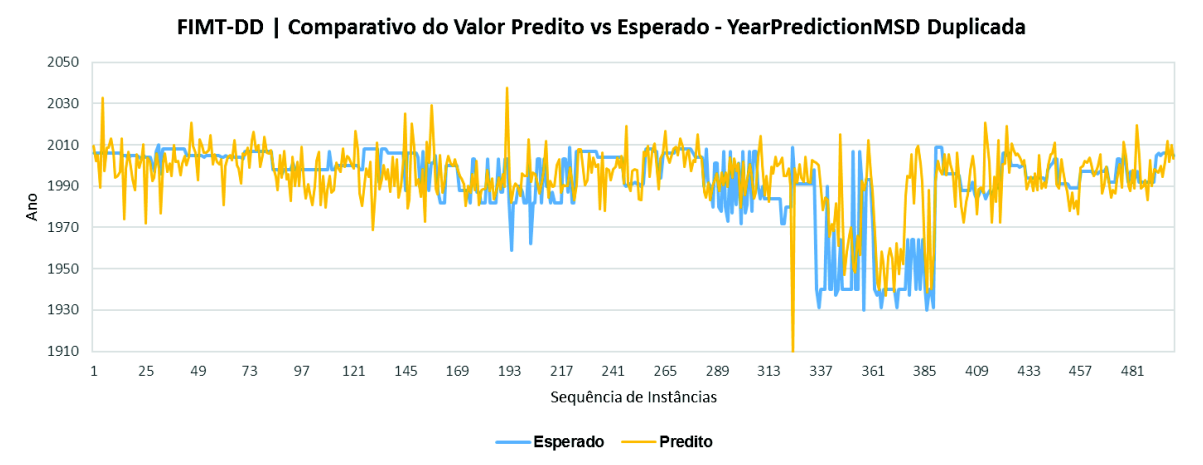
No sentido de avaliar o erro na comparação entre as duas métricas mais utilizadas, MAE e RMSE, buscou-se verificar também o valor predito em relação ao esperado. Nas Figuras 15, 16, 17 e 18 são apresentadas as comparações entre os algoritmos, onde foi utilizado uma amostragem dos últimos 500 registros avaliados. O aspecto negativo dessa avaliação foi o IBLStreams que sempre resultou em zero a predição, impossibilitando fazer um comparativo com os demais algoritmos nesta abordagem. Não foi possível identificar a causa desse comportamento, pois nesta versão do MOA não é listado outro algoritmo para a tarefa de regressão, em avaliações com algoritmos e bases para a tarefa de regressão foi gerado o valor predito.

Figura 15: Comparativo entre o valor esperado e o valor predito utilizando as últimas 500 instâncias de amostragem com o algoritmo AMRules para a base YearPredictionMSD duplicada.



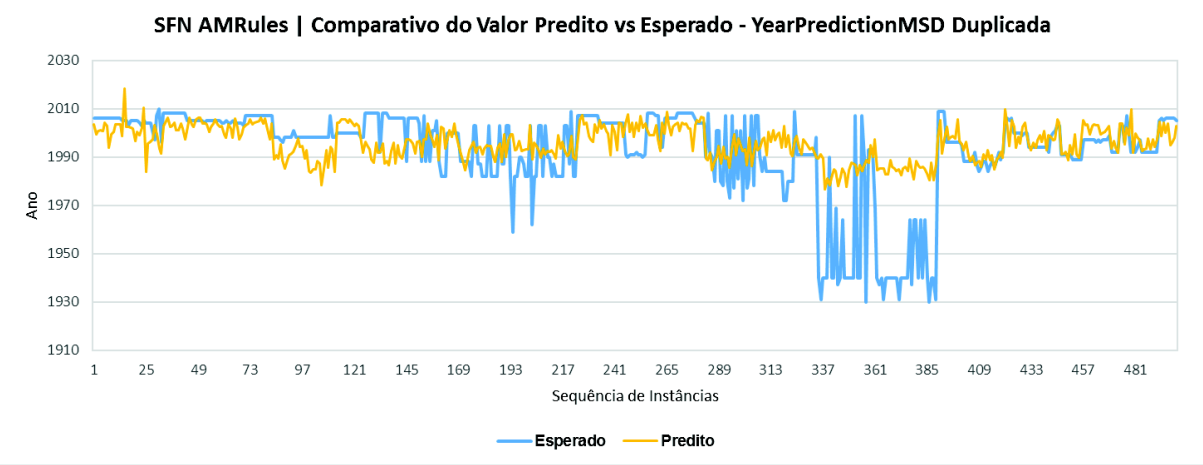
Fonte: Elaborador pelo autor.

Figura 16: Comparativo entre o valor esperado e o valor predito utilizando as últimas 500 instâncias de amostragem com o algoritmo FIMT-DD para a base YearPredictionMSD duplicada.



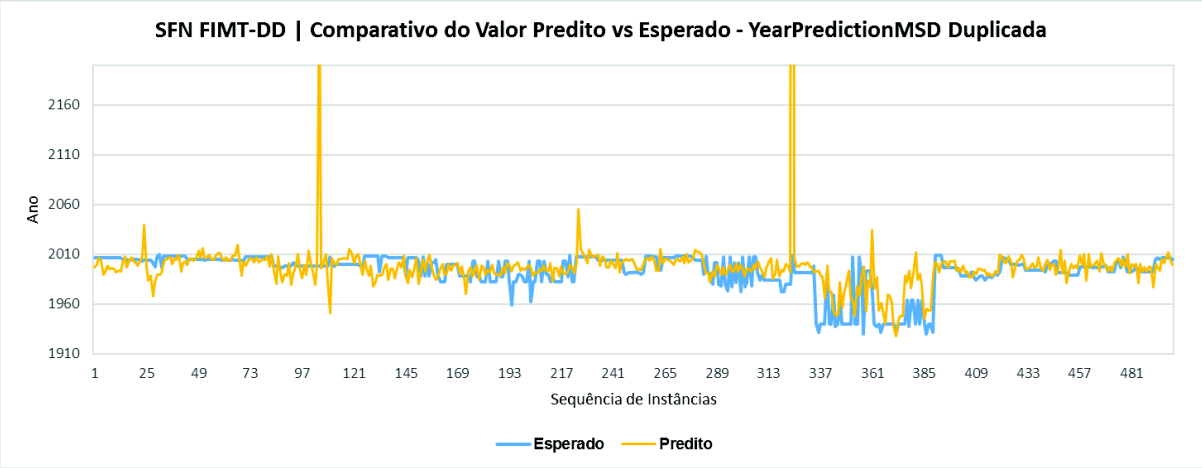
Fonte: Elaborador pelo autor.

Figura 17: Comparativo entre o valor esperado e o valor predito utilizando as últimas 500 instâncias de amostragem com o algoritmo SFNRegressor com AMRules para a base YearPredictionMSD duplicada.



Fonte: Elaborador pelo autor.

Figura 18: Comparativo entre o valor esperado e o valor predito utilizando as últimas 500 instâncias de amostragem com o algoritmo SFNRegressor com FIMT-DD para a base YearPredictionMSD duplicada.



Fonte: Elaborador pelo autor.

Quando realizada uma análise mais detalhada sobre esses resultados é possível verificar que após a segunda passagem o número de acertos foi maior em três abordagens, apenas com o algoritmo AMRules foi inferior. A Tabela 9 apresenta o número de instâncias corretas em cada algoritmo, considerando uma diferença inferior a um (1) entre predito e esperado, a base original tem 515.345 instâncias. Deve ser observado que o resultado da predição é baseado em cálculos sobre regras associativas, árvores de decisão entre outras técnicas, isso possibilita que valores preditos sejam diferentes do intervalo de valores utilizados no aprendizado.

Tabela 9: Número de acertos na predição da base YearPredictionMSD.

Base de dados	Indicador	Algoritmos			
		AMRules	FIMT-DD	SFN AMRules	SFN FIMT-DD
Original	Acertos	**55.338**	32.742	57.916	45.698
	Menor valor	1.641	-207.452	1.752	-38.962
	Maior valor	2.162	80.958	2.206	30.697
Duplicada (apenas)	Acertos	55.146	**36.737**	**58.570**	**46.186**
	Menor valor	1.927	-12.091	1.921	-45.757
	Maior valor	2.049	9.312	2.133	573.238

Fonte: Elaborador pelo autor.

Avaliando o resultado dos valores preditos é possível constatar que os algoritmos com base no AMRules, considerando a abordagem apenas com ele e através do SFNRegressor apresentaram melhores resultados, no sentido de mais acertos, observa-se também que o intervalo de valores preditos (entre o menor e o maior) diminui na segunda passagem dos dados. Para definição de acertos foi calculado que a diferença entre o predito e o esperado fosse inferior a um (1). Enquanto as abordagens com FIMT-DD resultam em valores bem distantes do desejado.

5.3 Concept Drift

Para a avaliar a capacidade de adaptação com *concept drift* um ambiente controlado, de bases sintéticas com inserção de *drifts* foram empregadas. Com isso foram utilizados diferentes abordagens e configurações, sendo explorados os principais métodos para detecção de *drift*: ADWIN e PHT. A Tabela 10 mostra um resumo das execuções com as três bases de dados utilizadas, onde o menor erro é obtido com o algoritmo SFNRegressor com AMRules. Entretanto, em contraste com esse resultado, o tempo de execução para atingir esses resultados foi cinco vezes maior que o algoritmo FIMT-DD.

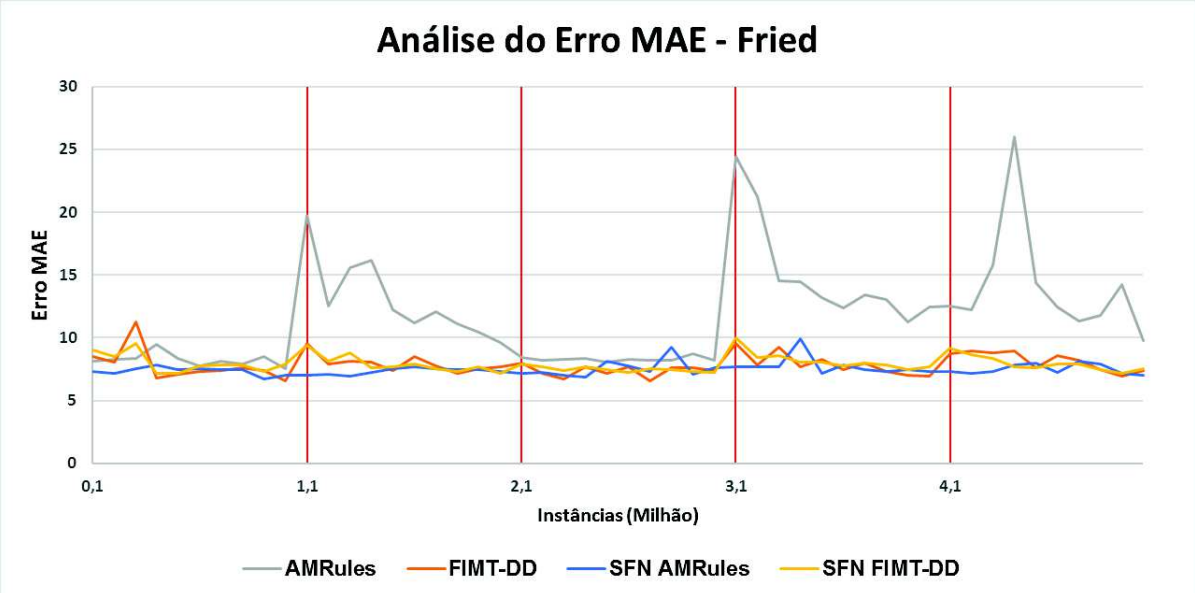
Tabela 10: Resumo dos resultados obtidos nos experimentos com as três bases sintéticas.

Algoritmo	Fried					FriedChar					FriedWOx4				
	MAE	RMSE	Tempo (seg)	Mem. (MB)	Média MAE	MAE	RMSE	Tempo (seg)	Mem. (MB)	Média MAE	MAE	RMSE	Tempo (seg)	Mem. (MB)	Média MAE
AMRules	9,76	12,77	498,98	**45,49**	11,73	12,11	16,35	451,89	**58,91**	14,68	23,48	27,72	284,22	**19,04**	24,03
FIMT-DD	7,36	21,11	**328,34**	344,55	7,84	24,35	35,97	**274,83**	279,91	24,01	24,35	35,97	**251,16**	279,83	24,01
IBLStreams	202,19	234,57	376,38	1387,03	204,94	204,03	237,32	309,36	668,78	204,16	202,19	234,57	406,66	1348,88	204,94
SFN AMRules	**7,01**	**8,45**	1737,20	158,42	**7,49**	**11,11**	**14,36**	1606,42	184,43	**12,12**	**22,67**	**26,72**	1111,06	105,97	**23,18**
SFN FIMT-DD	7,55	21,43	471,25	416,52	7,91	24,95	30,15	1241,70	917,30	24,86	24,14	31,56	998,78	912,67	23,89

Fonte: Elaborador pelo autor.

Nestes experimentos foram excluídos os resultados do algoritmo IBLStreams, devido ao elevado erro MAE apresentado, com números inesperados (superiores à 200), mantendo-se estável, apesar de diversas tentativas de parametrizações. Sendo assim optou-se por não exibir esses resultados. A Figura 19 apresenta a evolução do erro MAE durante o processo de aprendizado sobre a base Fried, destacando os pontos onde há *concept drift*, a cada 1M de instâncias (conforme definido na seção 4.2.3). Após 4M é esperado que o erro seja inferior na comparação com intervalo de 1 à 1M. Porém, ocorre o contrário, após *concept drift* C3 o erro não diminui para a maioria das amostragens e algoritmos. Este comportamento é repetido em experimentos com as bases FriedChar e FriedWOx4.

Figura 19: Análise do erro MAE durante o aprendizado sobre a base Fried.



Fonte: Elaborador pelo autor.

A Figura 19 destaca que o marcador está em 100K após o *concept drift*, (1,1M; 2,1M; 3,1M e 4,1M). Isto ocorre porque no ponto 1M ocorre a mudança, mas será preciso submeter a próxima amostragem (100K) para que o modelo seja atualizado e a saída tenha os dados com esse novo conceito. Em uma perspectiva diferente, a avaliação dos resultados de 1 á 1M (C1) comparados com os resultados de 4M à 5M (C5), que resultam em 10 amostragens, são projetados na Tabela 11, mostrando o erro MAE, onde o SFNRegressor obteve melhores resultados para C5.

Tabela 11: Comparativo do erro MAE entre os conceitos C1 e C5 com a base Fried.

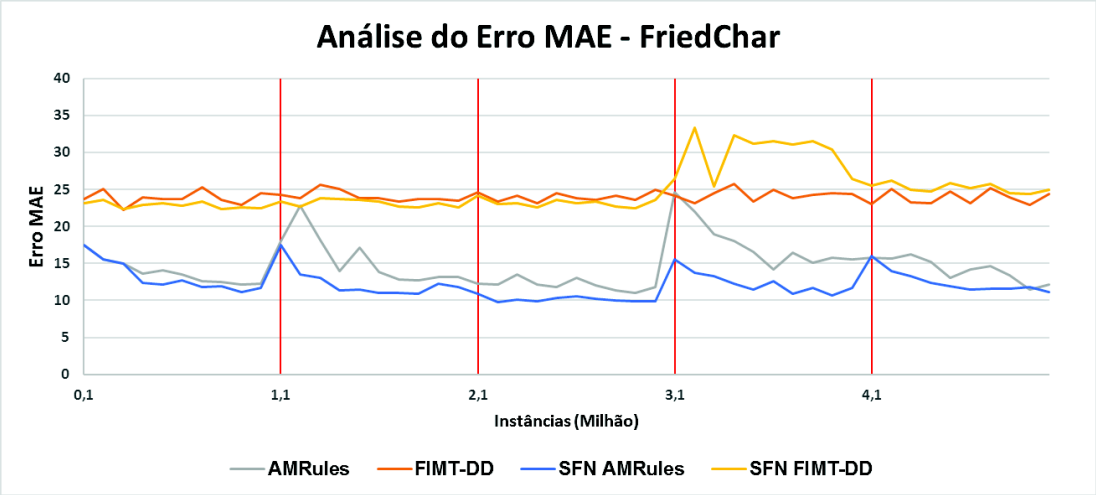
Intervalo de instâncias avaliadas	AMRules		FIMT-DD		SFN AMRules		SFN FIMT-DD	
	C1	C5	C1	C5	C1	C5	C1	C5
1-100K	**8,0984**	12,4928	**8,4633**	8,7548	7,3062	**7,2982**	**9,0542**	9,1891
100K-200K	**8,2825**	12,2446	**8,0712**	8,9275	**7,1443**	7,1496	**8,4966**	8,6277
200K-300K	**8,3579**	15,7587	11,2868	**8,7826**	7,4971	**7,2763**	9,5185	**8,3435**
300K-400K	**9,4816**	26,0131	**6,7653**	8,9678	7,8651	**7,8181**	**7,1407**	7,6740
400K-500K	**8,3361**	14,3701	**7,1029**	7,6051	**7,4790**	8,0105	**7,1735**	7,5890
500K-600K	**7,7847**	12,4642	**7,3169**	8,6110	7,5573	**7,1983**	**7,7921**	7,8985
600K-700K	**8,1056**	11,3151	**7,3949**	8,1879	**7,4900**	8,1268	**7,8162**	7,9015
700K-800K	**7,9197**	11,7470	7,6371	**7,4806**	**7,4253**	7,9310	7,7997	**7,4873**
800K-900K	**8,4985**	14,1953	7,3941	**6,9298**	**6,6857**	7,1667	7,3136	**7,1789**
900K-1M	**7,5162**	9,7553	**6,6003**	7,3562	**6,9757**	7,0142	7,8818	**7,5478**

Fonte: Elaborador pelo autor.

Os resultados mostram que o algoritmo AMRules acertou mais na primeira vez (C1) que avaliou os dados, apresentando um erro maior na segunda passagem (C5). Enquanto o algoritmo SFNRegressor apresentou um maior equilíbrio nos resultados, obtendo menores erros na segunda avaliação.

Nos experimentos com a base FriedChar é possível verificar um aumento no erro MAE, principalmente com os algoritmos FIMT-DD e SFNRegressor com FIMT-DD. Nessas abordagens foram exploradas alternativas na parametrização, avaliando tanto o modelo com a opção de árvore de regressão (RegressionTree) quanto a árvore de decisão tradicional. A Figura 20 apresenta o gráfico de evolução do erro de aprendizado.

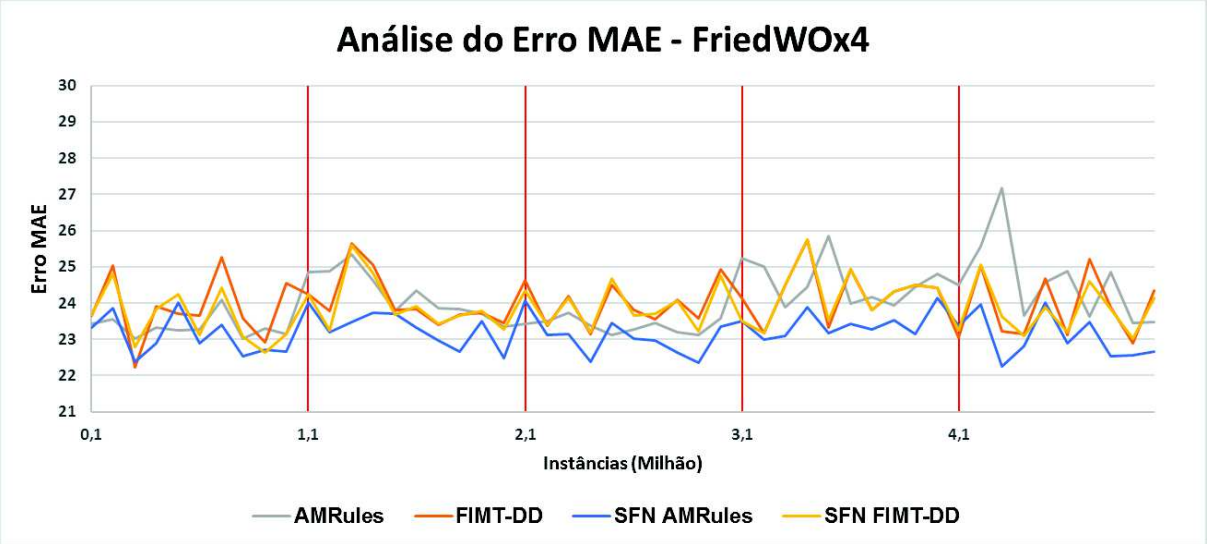
Figura 20: Análise do erro MAE durante o aprendizado sobre a base FriedChar.



Fonte: Elaborador pelo autor.

No último experimento, utilizando a base FriedWOx4, é possível observar que os erros são superiores a abordagens com as bases Fried e FriedChar. Isso permite confirmar a importância do atributo x_4 no processo de aprendizado. A Figura 21 apresenta o gráfico de evolução do erro durante o processo de aprendizado.

Figura 21: Análise do erro MAE durante o aprendizado sobre a base FriedWOx4.

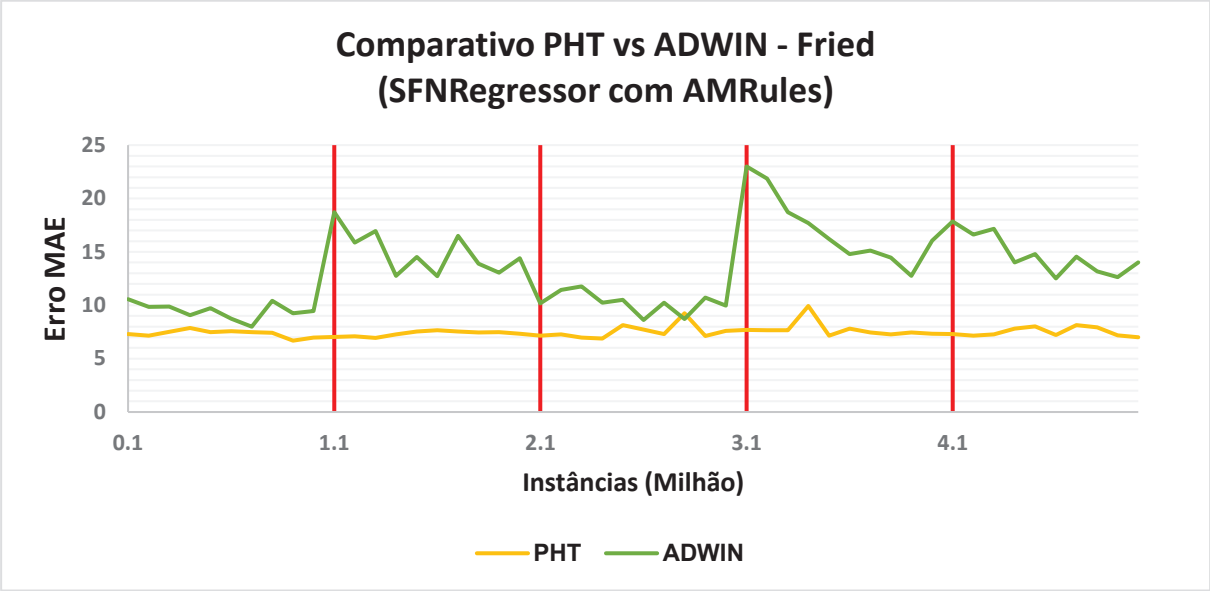


Fonte: Elaborador pelo autor.

Nesses experimentos o algoritmo SFNRegressor com AMRules apresentou menores erros, sendo inferior ao AMRules e mais estável. Mas exigiu muito mais tempo para obter esses resultados, com 28,9 minutos contra 8,3 minutos para o AMRules (Tabela 10). A avaliação com a abordagem de *Window*, definindo tamanhos entre 1K e 10K apresentaram melhores resultados que *Basic*, principalmente no contexto de *concept drift*, mais informações são exibidas no Apêndice D.

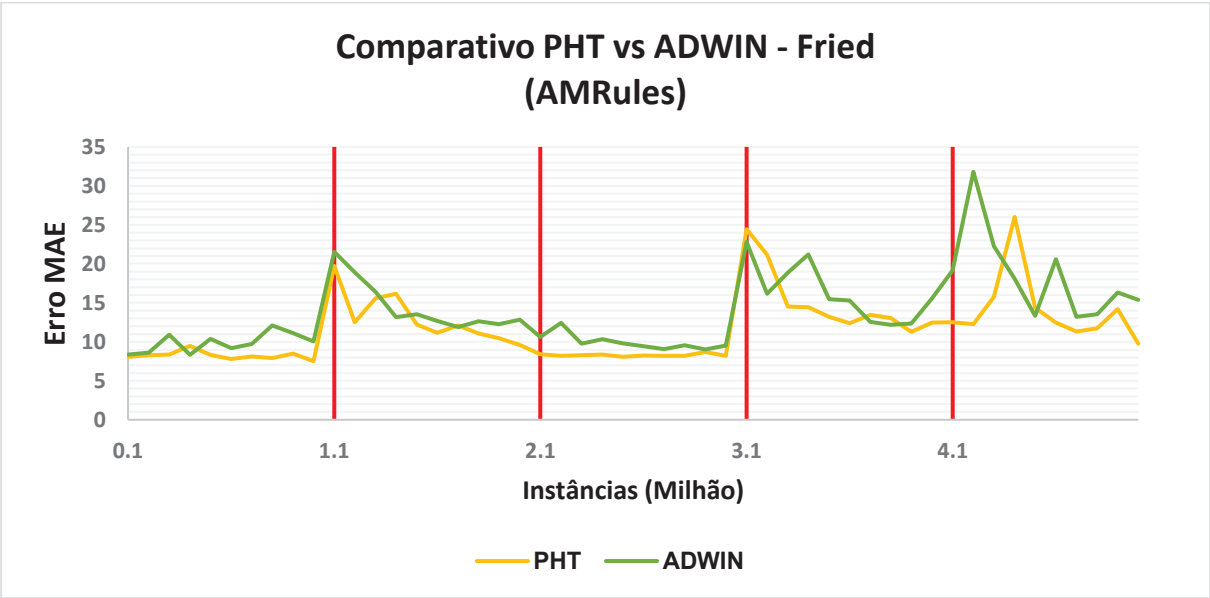
Na comparação dos métodos de detecção e adaptação, avaliando PHT e ADWIN, foram investigadas situações em que ambos apresentaram resultados promissores, sendo que não é possível determinar a melhor situação de forma absoluta. Principalmente pelo fato de que alguns algoritmos, como o FIMT-DD e IBLStreams, não permitem a utilização do método ADWIN. Nas Figuras 22 e 23 são projetados os comparativos utilizando o algoritmo SFNRegressor com AMRules e AMRules apenas, respectivamente.

Figura 22: Comparação entre PHT e ADWIN para detecção de *concept drift* utilizando o algoritmo SFNRegressor com AMRules.



Fonte: Elaborador pelo autor.

Figura 23: Comparação entre PHT e ADWIN para detecção de *concept drift* utilizando o algoritmo AMRules.

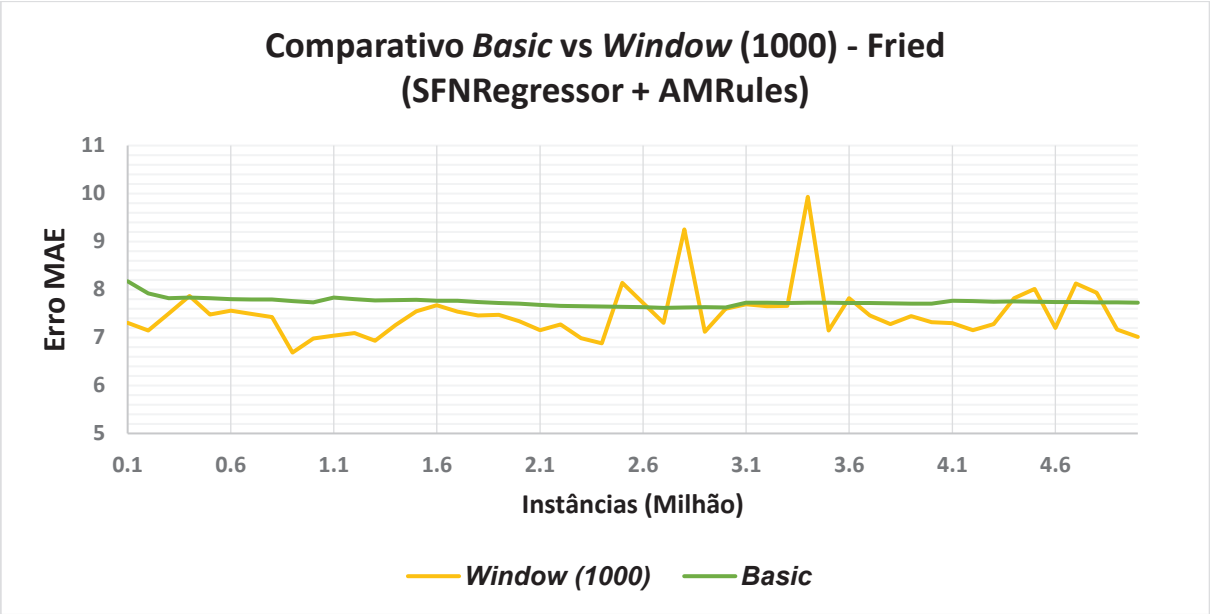


Fonte: Elaborador pelo autor.

Avaliando aspectos como a comparação entre o PHT e ADWIN para o algoritmo SFNRegressor foi visto que o primeiro método apresenta menores erros, entretanto exige mais tempo no processo de aprendizado. O aprendizado com ADWIN foi executado em 11 minutos e 28 segundos, enquanto o PHT executou em 28 minutos e 57 segundos. Na análise do mesmo comparativo com o algoritmo AMRules não foi identificado o mesmo comportamento, apresentando resultados muito próximos entre os métodos, tanto de erro quanto de tempo de execução.

Outro aspecto explorado nesses experimentos foi a opção de avaliação, realizando o comparativo, entre os tipos *Basic* e *Window*. Onde foram aplicadas diferentes configurações para o tamanho (w) da janela, conforme gráfico ilustrado na Figura 24.

Figura 24: Comparativo do erro MAE entre *Basic* e *Window* com o algoritmo SFNRegressor + AMRules.



Fonte: Elaborador pelo autor.

Os resultados repetem o mesmo comportamento em relação a outras execuções, onde o *Basic* segue estável, sem grandes oscilações. Enquanto a opção *Window* oferece possibilidade de menores erros, avaliações com tamanho da janela superior a 1000, como 10.000 e 20.000 apresentaram maiores oscilações, entretanto com erros superiores.

5.4 Considerações

Os resultados mostraram que, em geral, a memória é um aspecto que não preocupa, pois na maioria das experimentações não excedeu um grande volume. Sendo possível verificar que os algoritmos avaliados atendem a um dos fundamentos de *data stream mining*, os algoritmos que mais utilizam memória nos seus modelos foram IBLStreams e FIMT-DD, ressaltando que este último não gerou nenhum erro ou falta de espaço no gerenciamento de memória.

Nas avaliações com grandes bases de dados, como a Airlines, o algoritmo SFNRegressor apresentou os menores erros, mas em contrapartida exigiu mais tempo para executar, superando em 30 vezes o tempo de execução do algoritmo FIMT-DD. Sendo que a diferença do erro foi mínima. Com essa mesma base, mas em outra perspectiva, avaliando o tempo de aprendizado ao longo do processo, foi identificado que o algoritmo AMRules incrementa o tempo de aprendizado para cada amostragem, isto foi possível identificar apenas a partir de um grande volume de instâncias (superior a 15 milhões), em cenários com poucas instâncias não é perceptível este comportamento. Já o algoritmo IBLStreams se destacou como algoritmo que exigiu mais memória entre todos neste experimento, característica que impossibilitou execuções com a base completa da Airlines.

Na abordagem com a base YearPredictionMSD, onde foi explorado o comportamento dos algoritmos com dados recorrentes, a partir da replicação da base. Os resultados mostraram que na maioria dos casos a segunda passagem pelos dados não teve um ganho significativo, em alguns algoritmos gerou um erro maior, principalmente com o algoritmo AMRules. Apesar desse aspecto negativo, o algoritmo que apresentou menores erros foi o SFNRegressor utilizando o AMRules. Já o FIMT-DD se mostrou instável, com grandes oscilações entre a primeira e a segunda passagem, ainda que tenha apresentado mais acertos na segunda passagem, os erros foram maiores.

Quando incluído atributos que aumentam o ruído dos dados, foi visto que estes não interferem no resultado final. Constatação obtida através do uso da base sintética gerada a partir da fórmula de Friedman (1991), onde cinco atributos são utilizados sem interferir no resultado e os mesmos não influenciam negativamente no aprendizado, considerando a métrica de erro. Embora, possa aumentar o tempo do aprendizado, apesar de ainda assim se mostrar eficiente para a contexto de *data stream*, onde não há a etapa de pré-processamento.

Quando avaliado o *concept drift* foi constatado que apenas o algoritmo IBLStreams não permitiu detectar a ocorrência, onde apresentou erros muito acima dos demais, sendo excluídos dos resultados. Os demais algoritmos se mostraram capazes de identificar a mudança, comprovando que os métodos PHT e ADWIN são eficientes para tratar esse tipo de comportamento, apesar de não ser possível destacar que um seja superior ao outro de forma absoluta, embora nas execuções com o algoritmo SFNRegressor o método ADWIN seja muito mais rápido que o PHT, exigindo menos de 50% do tempo. Salienta-se também que o algoritmo FIMT-DD suporta apenas o PHT, enquanto o IBLStreams não oferece este mecanismo.

Nos experimentos foram analisados resultados com as duas opções de avaliação, *Basic* e *Window*. Sendo possível verificar que o primeiro atua de forma estável, demorando mais para baixar o erro, porém não oscila, seguindo estável. Enquanto o método utilizando de avanço com janelas, empregados em diferentes configurações permitiu um erro menor, porém oscilou mais, sendo este um comportamento que depende do domínio explorado. Neste contexto, outros métodos para evolução do treinamento poderiam ser utilizados, entretanto o MOA não oferece suporte a um mecanismo como o fator de esquecimento (*fading factor*) para a tarefa de regressão, apenas para classificação.

Na maioria dos experimentos o algoritmo SFNRegressor apresentou os menores erros, entretanto pesa na avaliação o fato de exigir muito mais tempo para o aprendizado. Como aspecto positivo foi possível verificar que o mesmo atua de forma estável com grande volume de dados, no aspecto tempo de execução. Dentro dos recursos computacionais, a memória não exigiu grandes atenções, por outro lado, o recurso mais exigido nas execuções, porém não monitorado, foi o de processamento (CPU).

6 CONCLUSÃO

Neste trabalho, uma avaliação empírica de algoritmos de mineração de *data stream* para regressão foi empregada gerando um estudo investigativo, considerando os principais aspectos indicados pela literatura, como memória, tempo de execução e erro, onde foram explorados quatro algoritmos: FIMT-DD, AMRules, IBLStreams e SFNRegressor. Muitos trabalhos têm sido divulgados sobre este tema, a maioria deles focado apenas sobre uma métrica, o erro. Com isso utilizam abordagens com bases de dados estacionárias, com um pequeno número de amostras (inferior a 100 mil). Em outros casos aplicam comparativos com algoritmos de mineração de dados tradicional. Conforme a literatura avaliada, esse tipo de abordagem pode gerar resultados passíveis de questionamentos, considerando que as técnicas e conceitos apresentam significativas diferenças. Dessa forma, os resultados deste trabalho demonstram a importância de utilizar cenários com bases não-estacionárias, além de destacar a relevância da limitação de recursos computacionais. Também foi explorada a capacidade de detecção e adaptação de *concept drift*, característica fundamental de *data stream*.

Como principais contribuições deste trabalho são destacadas: a avaliação de critérios fundamentais, memória, tempo de execução e poder de generalização, relacionados a regressão para *data stream*; a produção de uma análise crítica dos algoritmos investigados; e a possibilidade de reprodução e extensão dos estudos realizados pela disponibilização das parametrizações empregadas.

Sendo assim, nos comparativos que buscavam avaliar a métrica de tempo de execução foram obtidos significativos indicadores, entre os quais destaca-se o tempo de execução ao longo do processo de aprendizado. Esse aspecto é apresentado de forma inédita em relação aos trabalhos relacionados, onde foi demonstrado que no emprego de bases massivas não-estacionárias, os resultados finais apresentam diferentes comportamentos, exaltando pontos que podem ser otimizados, principalmente nos algoritmos AMRules e IBLStreams que apresentaram um aumento acumulativo no tempo de execução para cada amostragem.

A replicação de dados, quando empregada para gerar recorrência de padrões sem *concept drift* explícito (YearPredictionMSD), demonstrou que os algoritmos conseguem classificar melhor no aspecto de erro gerado. No entanto, quando explorada a replicação de dados com a ocorrência de *concept drift* (sintética), foi visto que os algoritmos não apresentaram o mesmo desempenho, principalmente o algoritmo AMRules, resultando em erros muito superiores.

Os resultados obtidos com os experimentos explorando a presença de *concept drift* não demonstraram qual dos métodos oferece melhor classificação. Embora o método ADWIN tenha apresentado um desempenho superior na métrica tempo de execução, considerando o algoritmo SFNRegressor com AMRules. Importante ressaltar que neste comparativo apenas duas abordagens permitiram utilizar os métodos PHT e ADWIN.

De uma forma geral, a investigação demonstrou que o tempo de aprendizado é o fator mais crítico entre as métricas avaliadas. Sendo comprovado que tal conclusão foi possível somente empregando bases massivas com características não-estacionárias, enquanto o gerenciamento de memória se mostrou eficaz em quase todos os cenários, a exceção do algoritmo IBLStreams. Este trabalho buscou detalhar todos os experimentos realizados, expondo configurações utilizadas de forma que possibilite a reprodução das execuções. Os resultados obtidos foram satisfatórios para o propósito pretendido, ainda assim existe espaço para aprimoramentos.

Para trabalhos futuros são indicados: a extensão dos estudos com os algoritmos utilizados, como ajustes nas configurações dos métodos de detecção de *concept drift*. Espera-se que após a publicação de uma versão mais recente do algoritmo IBLStreams possibilite que o mesmo seja comparado em todos os aspectos como os demais. Também sugere-se realizar o mesmo estudo com o SAMOA, que oferece uma plataforma para paralelismo computacional. Assim como estudos com bases *multi-targets* e o uso de séries temporais.

REFERÊNCIAS

ABRAMOWITZ, M.; STEGUN, I. A. **Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables**. Courier Corporation, v. 55, 1964.

AGGARWAL, C. C. **On biased reservoir sampling in the presence of stream evolution**. In Proceedings of the 32nd International Conference on Very large data bases. VLDB Endowment, 2006, p. 607-618.

AHA, D. W.; KIBLER, D.; ALBERT, M. K. **Instance-based learning algorithms.** Machine Learning, v. 6, n. 1, p. 37-66, 1991.

ALBERT, R.; BARABÁSI, A. L. **Statistical mechanics of complex networks**. Reviews of Modern Physics, v. 74, n. 1, p. 47, 2002.

ALMEIDA, E.; FERREIRA, C.A.; GAMA, J. **Learning Model Rules From High-Speed Data Streams**. In Proceedings of the 3rd International Conference on Ubiquitous Data Mining-Volume 1088. CEUR-WS, 2013, p.10.

APTÉ, C.; WEISS, S. **Data mining with decision trees and decision rules**. Future Generation Computer systems, v. 13, n.2, p.197-210, 1997.

ARMSTRONG, J. S.; COLLOPY, F. Error measures for generalizing about forecasting methods: Empirical comparisons. **International Journal of Forecasting**, v.8, n.1, p.69-80, 1992.

BABCOCK, B.; BABU, S.; DATAR, M.; MOTWANI, R.; WIDOM, J. **Models and issues in data stream systems**. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2002, p. 1-16.

BARDDAL, J.P.; GOMES, H.M.; ENEMBRECK, F. Advances on Concept Drift Detection in Regression Tasks Using Social Networks Theory. **International Journal of Natural Computing Research** (IJNCR), v. 5, n. 1, p.26-41, 2015.

BASSEVILLE, M.; E NIKIFOROV, I. V. **Detection of abrupt changes: theory and application**. Englewood Cliffs: Prentice Hall, v.104, 1993.

BENNETT, N. D.; CROKE, B. F.; GUARISO, G.; GUILLAUME, J. H.; HAMILTON, S. H.; JAKEMAN, A. J.; PIERCE, S. A. **Characterising performance of environmental models.** Environmental Modelling & Software, v. 40, p.1-20, 2013.

BIFET, A.; DE FRANCISCI MORALES, G.; READ, J.; HOLMES, G. e PFAHRINGER, B.. **Efficient Online Evaluation of Big Data Stream Classifiers**. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015, p. 59-68.

BIFET, A.; GAVALDA, R.. **Learning from Time-Changing Data with Adaptive Windowing**. In Proceedings of the 2007 SIAM International Conference on Data Mining, v. 7, 2007.

BIFET, A.; HOLMES, G.; KIRKBY, R. e PFAHRINGER, B. Moa: Massive online analysis. **The Journal of Machine Learning Research**, v. 11, p.1601-1604, 2010.

BIFET, A.; HOLMES, G.; PFAHRINGER, B.; KIRKBY, R. e GAVALDÀ, R. **New ensemble methods for evolving data streams**. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2009, p. 139-148.

CLEARWATER, S. H.; CHENG, T. P.; HIRSH, H.; BUCHANAN, B. G. **Incremental batch learning**. In Proceedings of the sixth international workshop on Machine learning. Morgan Kaufmann Publishers Inc., 1989, p. 366-370.

DASARATHY, B. V. **Nearest neighbor (NN) norms: NN pattern classification techniques.** Los Alamitos: IEEE Computer Society Press v.1, 1990.

DEAN, J. **Big data, data mining, and machine learning: value creation for business leaders and practitioners**. John Wiley & Sons, Hoboken, New Jersey. 2014.

DOMINGOS, P.; HULTEN, G. **Mining high-speed data streams**. In Proceedings of The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2000, p.71-80.

DOMINGOS, P.; HULTEN, G. **A general method for scaling up machine learning algorithms and its application to clustering**. In International Conference on Machine Learning,Vol. 1, 2001a, p. 106-113.

DOMINGOS, P.; HULTEN, G. **Catching up with the Data: Research Issues in Mining Data Streams**. In DMKDb, 2001b.

DIETTERICH, T. G.; KONG, E. B. **Machine Learning Bias, Statistical Bias, and Statistical Variance of Decision Tree Algorithms**. Technical Report, Department of Computer Science, Oregon State University, 1995.

FAN, J.; HAN, F. e LIU, H. **Challenges of Big Data analysis**. National Science review, 2014, v. 1, p. 293-314.

FAYYAD, U.; PIATETSKY-SHAPIRO, G. e SMYTH, P. . **The KDD process for extracting useful knowledge from volumes of data**. Communications of the ACM, 1996, p. 27-34.

GABER, M. M. **Advances in data stream mining**. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, Elsivier, DOI: 10.1017/CBO9781107415324.004, v. 1, 2012 p.79-85.

GABER, M.M.; ZASLAVSKY, A. e KRISHNASWAMY, S. **Mining data streams: a review.** ACM Sigmod Record, v. 34, n.2. p.18-26, 2005.

GAMA, J. **Knowledge discovery from data streams**. CRC Press. 2010.

GAMA, J. e PEDERSEN, R.U. **Predictive Learning in Sensor Network**. Learning from data streams. Springer-Verlag Berlin Heidelberg, 2007, p. 143-163.

GAMA, J.; ROCHA, R.; MEDAS, P. **Accurate decision trees for mining high-speed data streams**. In Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2003, p. 523-528.

GAMA, J. e RODRIGUES, P.P. **Data Stream Processing.** Learning From Data Streams. Springer-Verlag Berlin Heidelberg, 2007, p. 25-38.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. **Issues in evaluation of stream learning algorithms**. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2009, p. 329-338.

GAMA, J.; ŽLIOBAITĖ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. **A survey on concept drift adaptation**. ACM Computing Surveys (CSUR), v. 46, n. 4, p. 44, 2014.

GANTZ, J.; REINSEL, D. **The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east**. IDC iView: IDC Analyze the future 2007, 2012. p. 1-16.

GAO, J.; FAN, W.; HAN, J. e PHILIP, S.Y. **A General Framework for Mining Concept-Drifting Data Streams with Skewed Distributions**. In Proceedings of the 2007 SIAM International Conference on Data Mining, 2007, p. 3-14.

GELMAN, A.; HILL, J. **Data analysis using regression and multilevel/hierarchical models.** Cambridge University Press, 2006.

HALL, M. A. e HOLMES, G. **Benchmarking attribute selection techniques for discrete class data mining**. Knowledge and Data Engineering, IEEE Transactions on, v. 15, n. 6, 1437-1447, 2003.

HALL, M.; WITTEN, I.; FRANK, E. **Data mining: Practical machine learning tools and techniques.** Morgan Kaufmann, Burlington. 2011.

HAN, J. e KAMBER, M. **Data mining: concepts and techniques.** Elsevier, San Francisco, CA, EUA, 2006. p. 8-9.

HAND, D.J.; MANNILA, H. e SMYTH, P. **Principles of data mining**. MIT Press, 2001.

HOEFFDING, W. Probability inequalities for sums of bounded random variables. **Journal of the American statistical association**, v. 58, n. 301, p.13-30, 1963.

HOGG, R. e CRAIG, T. **Introduction to mathematical statistics.** Upper Saddle River, New Jersey: Prentice Hall, 1995.

IKONOMOVSKA, E.; GAMA, J.; DŽEROSKI, S. **Learning model trees from evolving data streams.** Data mining and knowledge discovery, v. 23, p. 128-168, 2011.

IKONOMOVSKA E, GAMA J e DŽEROSKI S. **Online tree-based ensembles and option trees for regression on evolving data streams.** Neurocomputing, v. 150, p. 458-470, 2015.

IKONOMOVSKA, E.; GAMA, J.; ZENKO, B. e DZEROSKI, S. **Speeding-up Hoeffding-Based Regression Trees with Options**. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), 2011, p. 537-544.

JAPKOWICZ, N.; SHAH, M. **Evaluating Learning Algorithms: a Classification Perspective.** Cambridge University Press, 2011.

JORDAN, M. I.; MITCHELL, T. M. **Machine Learning: Trends, Perspectives, and Prospects**. Science, v. 349, n. 6245, p.255-260, 2015.

KLINKENBERG, R. **Learning drifting concepts: Example selection vs. example weighting**. Intelligent Data Analysis, v. 8, n. 3, p.281-300, 2004.

LANEY, D. **3D data management: Controlling data volume, velocity and variety**. META Group Research Note, v. 6. p.70, 2001.

LOSSIO-VENTURA J.A.; ALATRISTA-SALAS H. Overview of SIMBig 2015: 2nd Annual International Symposium on Information Management and Big Data. **2nd Annual International Symposium on Information Management and Big Data**, 2015. p. 10.

LUGHOFER, E. D. **FLEXFIS: a robust incremental learning approach for evolving Takagi–Sugeno fuzzy models.** Fuzzy Systems, IEEE Transactions on, v. 16, n. 6, p. 1393-1410, 2008.

KOYCHEV, I. **Tracking changing user interests through prior-learning of context.** In Adaptive Hypermedia and Adaptive Web-Based Systems. Springer Berlin Heidelberg, 2002, p. 223-232.

MARSLAND, S. **Machine Learning: an Algorithmic Perspective**. CRC press, 2015.

MITCHELL, T. M. **The need for biases in learning generalizations**. New Jersey: Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ., 1980, p. 184-191.

MITCHELL, T. M. **Machine learning**. Machine Learning. McGraw-Hill Science/Engineering/Match, Redmond, WA (EUA) . 1997, p. 432.

MITCHELL, T. M. **Machine learning and data mining.** Communications of the ACM, 42(11), 1999, p. 30-36.

MOHAMMED, H.K.; SOLIMAN. **Data stream mining**. 2010. Disponível em: < https://www.researchgate.net/profile/Hoda_Mohamed4/publication/259647558_Data_Stream_Mining/links/00b4952d184c01f0db000000.pdf>. Acesso em: 27 março 2016.

MORALES, G. D. F.; BIFET, A. Samoa: Scalable advanced massive online analysis. **Journal of Machine Learning Research**, v. 16, p. 149-153, 2015.

NEWMAN, M. **Networks: an introduction.** Oxford Universid Press, 2010.

QUINLAN, J. R. **Induction of decision trees**. Machine learning, v. 1, n. 1, p. 81-106, 1986.

QUINLAN, J. R. **Learning with continuous classes.** In 5th Australian Joint Conference on Artificial Intelligence, 1992, p. 343-348.

SHAKER, A. e HÜLLERMEIER, E. **IBLStreams: a system for instance-based classification and regression on data streams**. Evolving Systems, v. 3, n. 4, 2012, p.235-249.

SNIJDERS, C.; MATZAT, U. e REIPS, U.D. . Big data: Big gaps of knowledge in the field of internet science. **International Journal of Internet Science**, v. 7, n.1, p.1-5, 2012.

SRIMANI, P. K.; PATIL, M. M. Regression Model using Instance based Learning Streams. **Indian Journal of Science and Technology**, v. 7, n. 6, p. 864, 2014.

TAKAGI, T.; SUGENO, M. **Fuzzy identification of systems and its applications to modeling and control.** Systems, Man and Cybernetics, IEEE Transactions on, n. 1, p. 116-132, 1985.

WEISS, S. M.; INDURKHYA, N. **Rule-based regression**. In Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993, p. 1072-1078.

WEISS, S. M.; INDURKHYA, N. Rule-based machine learning methods for functional prediction. **Journal of Artificial Intelligence Research**, v. 3, p. 383-403, 1995.

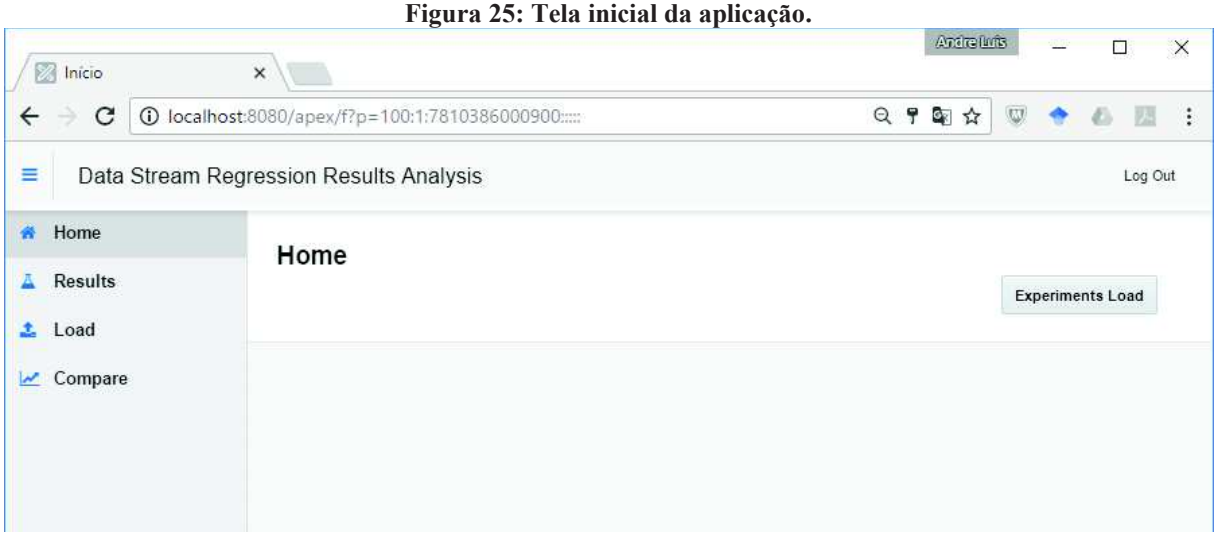
VITTER, J.S. **Random sampling with a reservoir**. ACM Transactions on Mathematical Software, v. 11, n. 1, p.37-57, 1985.

UYSAL, İ.; GÜVENIR, H. A. **An overview of regression techniques for knowledge discovery**. The Knowledge Engineering Review, v. 14, n. 4, p. 319-340, 1999.

APÊNDICE A APLICAÇÃO DSRRA

Na Figura 25 é apresentada a tela inicial da aplicação. A captura de telas foi realizada utilizando o navegador de internet Chrome.

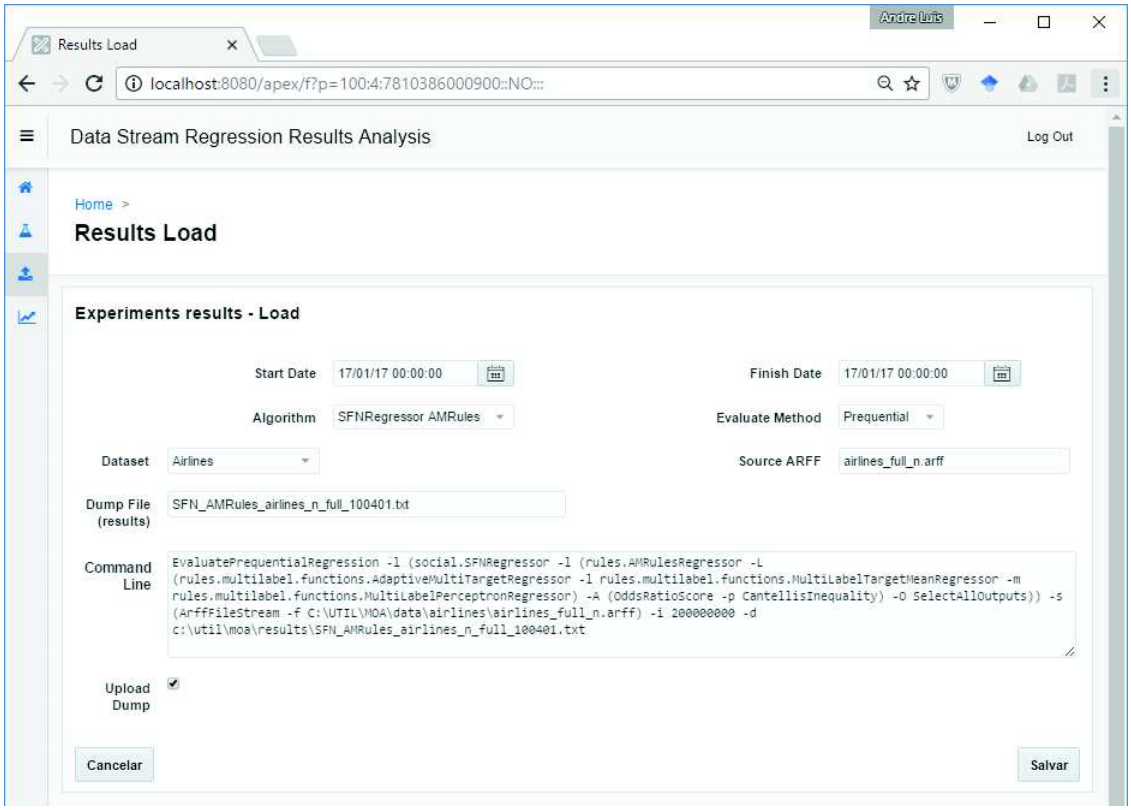
Figura 25: Tela inicial da aplicação.



Fonte: Elaborador pelo autor.

Na Figura 26 é possível visualizar a tela utilizada para realizar a entrada dos resultados de um experimento. Onde a partir da da entrada de algumas informações e a linha de comando utilizada são extraídas as configurações definidas.

Figura 26: Tela de carregar resultados dos experimentos.



Fonte: Elaborador pelo autor.

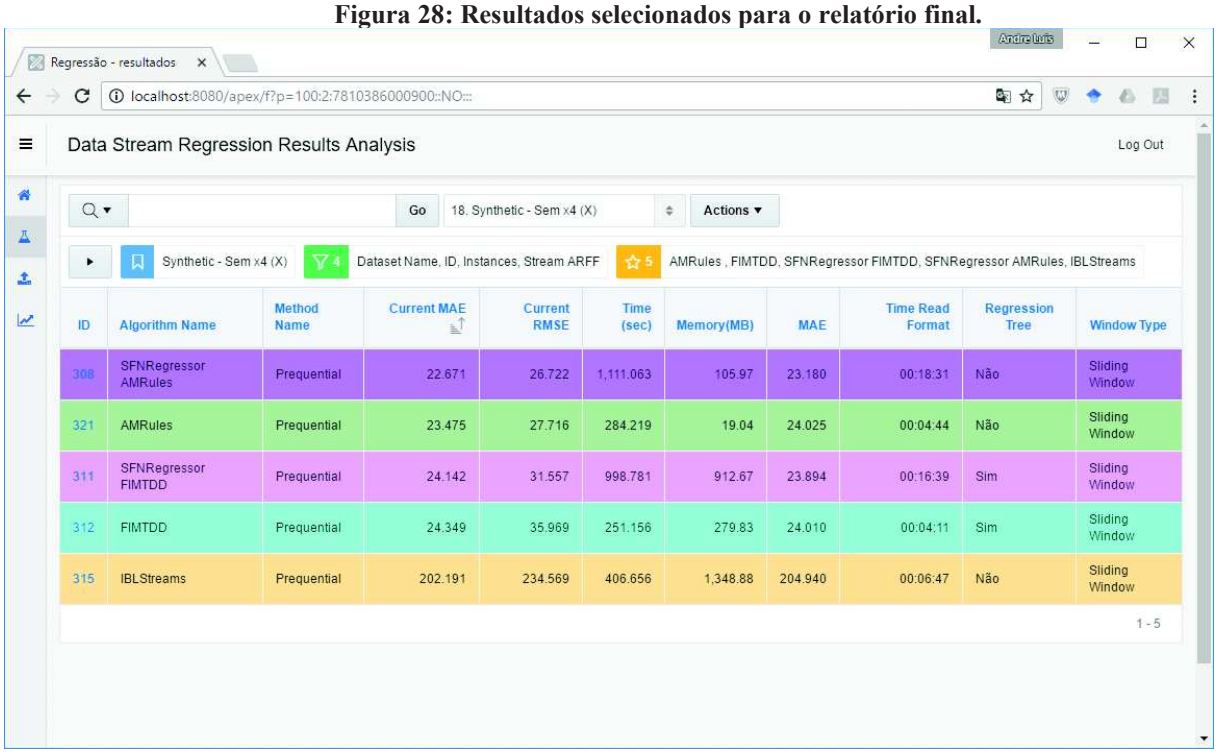
Na Figura 27 é apresenta a tela com todos os experimentos salvos. As colunas projetadas em tela são dinâmicas, permitindo definir quais são exibidas.

Figura 27: Resultados gerais dos experimentos.

Fonte: Elaborador pelo autor.

Na Figura 28 é ilustrado um relatório personalizado salvo (18.Synthetic – Sem x4(x)), onde foi configurado uma cor de fundo para cada algoritmo.

Figura 28: Resultados selecionados para o relatório final.



Fonte: Elaborador pelo autor.

Na Figura 29 é detalhado o dicionário de dados da aplicação.

Figura 29: Descrição do dicionário de dados.

Table EXECUTION

Name	Type	Comment
IDEXECUTION	NUMBER(38)	ID sequencial (autoincremento)
START_DATE	DATE	Data e hora do início da execução
FINISH_DATE	DATE	Data e hora do fim da execução
ALGORITHM	CHAR(1)	Indicador de qual algoritmo (A: AMRules; F: FIMT-DD; I: IBLStreams; S: SFN+AMRules; R: SFN+FIMTDD)
EVALUATE_METHOD	CHAR(1)	Indicador do método de avaliação (P: Prequential; H: Holdout)
DATASET	CHAR(1)	Indicador da base de dados (A: Airlines; M: YearPredictionMSD; S: Sintética)
STREAM_ARFF	VARCHAR(128)	Nome do arquivo do dataset
COMMAND_LINE	VARCHAR(1000)	Linha comando utilizado para execução do experimento
RESULT_FILE	VARCHAR(128)	Nome do arquivo de resultado
EVALUATOR	VARCHAR(1)	Tipo de janelamento (W: SlidingWindow; B: Basic)
REGRESSION_TREE	NUMBER(1)	Atributo exclusivo para aprendizados com o FIMT-DD, indica se foi gerado uma árvore de regressão
DRIFT_METHOD	CHAR(1)	Tipo de técnica utilizado (P: PHT; A:ADWIN)

Table REGRESSION_RESULT

Name	Type	Comment
IDREGRESSION_RESULT	NUMBER(38)	ID sequencial (autoincremento)
IDEXECUTION	NUMBER(38)	Referencia para o ID da tabela "Execution"
LEARNING_EVAL_INSTANCES	NUMBER(38)	Número de instâncias avaliadas no aprendizado
EVALUATION_TIME_SECONDS	NUMBER	Tempo (em segundos) demorada para o aprendizado
MODEL_COST_HOURS	NUMBER	Indicativo de custo gerado pela ferramenta MOA
CLASSIFIED_INSTANCES	NUMBER(38)	Número de instâncias classificadas
MAE	NUMBER	Métrica de erro da regressão MAE
RMSE	NUMBER	Métrica de erro da regressão RMSE
TRAINING_INSTANCES	NUMBER(38)	Número de instâncias treinadas
SERIALIZED_SIZE_BYTES	NUMBER	Total de memória utilizada no treinamento
MODEL_SIZE	NUMBER(38)	Tamanho do modelo (árvore, regras associativas, nós, etc.)
ANOMALY_DETECTIONS	NUMBER(38)	Indicador gerado por um dos algoritmos estudado (AMRules)
NUM_OF_DRIFT	NUMBER(38)	Quantidade de concept drift detectados no treinamento

Fonte: Elaborador pelo autor.

APÊNDICE B EXPERIMENTOS NÃO DETALHADOS NOS RESULTADOS

Os experimentos com resultados não considerados bons são apresentados à seguir, no Quadro 3 estão os experimentos utilizando a base Fried apenas.

Quadro 3: Experimentos não selecionados no trabalho utilizando a base Fried.

Algorithm Name	Current MAE	Current RMSE	Memory (MB)	Time	Command Line
AMRules	9.755	12.771	45.4	00:07:41	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -d AMRules_FriedBasic2_1125_01.txt -a 0.05
AMRules	11.998	19.758	45.49	00:06:33	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -e BasicRegressionPerformanceEvaluator -d AMRules_FriedBasic3_1128_06.txt -a 0.05
AMRules	12.684	17.497	28.78	00:05:42	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H (ADWINChangeDetector -a 0.01) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -i 115000000 -d AMRules_frie3_5m_1217_01.txt -w 10000 -a 0.05
AMRules	14.612	27.838	42.84	00:06:25	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H PageHinkleyDM -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -i 115000000 -d AMRules_frie3_5m_1217_03.txt -w 10000 -a 0.05
AMRules	15.389	21.076	34.35	00:07:15	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H ADWINChangeDetector -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -i 10000000 -d AMRules_Fried3_0219_003.txt -o AMRules_Fried3_0219_003.pred -a 0.03
AMRules	17.252	26.232	31.01	00:05:43	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H (ADWINChangeDetector -a 0.005) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -i 115000000 -d AMRules_frie3_5m_1217_02.txt -w 10000 -a 0.05
AMRules	50.065	63.146	0.24	00:04:28	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -c 0.0 -g 20 -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H ADWINChangeDetector -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f friedman_basic3.arff) -i 10000000 -d AMRules_Fried3_0219_001.txt -o AMRules_Fried3_0219_001.pred -a 0.03
FIMTDD	8.234	20.245	344.55	00:05:30	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f

					friedman_basic3.arff) -e BasicRegressionPerformanceEvaluator -d FIMTDD_FriedBasic3_1128_07.txt -a 0.05
SFNRegressor AMRules	7.358	9.055	160.29	00:23:34	EvaluatePrequentialRegression -l (social.SFNRegressor -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H PageHinkleyDM -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_AMRules_Syntheticl_0121_06.txt -w 10000 -a 0.05
SFNRegressor AMRules	7.724	10.82	158.38	00:23:43	EvaluatePrequentialRegression -l (social.SFNRegressor -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e BasicRegressionPerformanceEvaluator -i 10000000 -d SFNA_Fried3_0221_001.txt -a 0.03
SFNRegressor AMRules	9.15	11.796	132.43	00:23:06	EvaluatePrequentialRegression -l (social.SFNRegressor -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H ADWINChangeDetector -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_AMRules_Syntheticl_0121_05.txt -w 10000 -a 0.05
SFNRegressor AMRules	12.209	16.237	100.51	00:20:47	EvaluatePrequentialRegression -l (social.SFNRegressor -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H (ADWINChangeDetector -a 0.20261) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_AMRules_Syntheticl_0121_04.txt -w 10000 -a 0.05
SFNRegressor AMRules	13.144	18.634	99.46	00:10:48	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H (ADWINChangeDetector -a 0.015) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_AMRules_Syntheticl_0121_02.txt -w 10000 -a 0.05
SFNRegressor AMRules	14	22.574	104.14	00:11:28	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H ADWINChangeDetector -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_AMRules_Syntheticl_0121_01.txt -w 10000 -a 0.05
SFNRegressor AMRules	14	22.574	104.14	00:11:28	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H ADWINChangeDetector -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i

					115000000 -d SFN_AMRules_Syntheticl_0121_01.txt -w 10000 -a 0.05
SFNRegressor AMRules	20.1	31.723	89.39	00:10:07	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H (ADWINChangeDetector -a 0.20261) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_AMRules_Syntheticl_0121_03.txt -w 10000 -a 0.05
SFNRegressor FIMTDD	7.625	20.978	1,099.43	00:21:16	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f friedman_basic3.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d SFN_FIMTDD_Syntheticl_0217_01.txt -w 10000 -a 0.05
SFNRegressor FIMTDD	8.15	18.69	1,099.43	00:20:39	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f friedman_basic3.arff) -e BasicRegressionPerformanceEvaluator -d SFN_FIMTDD_FriedBasic3_1128_01.txt

Fonte: Elaborador pelo autor.

Os experimentos com resultados não considerados bons com a base de dados YearPrediction são apresentados no Quadro 4.

Quadro 4: Experimentos não selecionados no trabalho utilizando a base YearPrediction original.

Algorithm Name	Current MAE	Current RMSE	Memory (MB)	Time	Command Line
AMRules	6.672	9.392	30.42	00:02:32	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -i 600000 -f 10000 -q 10000 -d AMRules_Music_orig_0120_03.txt -a 0.06
AMRules	6.672	9.392	30.44	00:03:02	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -i 520000 -f 10000 -q 10000 -d AMRules_YearPredictionMSD_01.txt.csv -o AMRules_YearPredictionMSD_01.pred
AMRules	7.7	11.727	492.3	00:12:29	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 1000000 -f 10000 -q 10000
AMRules	7.979	12.356	30.43	00:03:41	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 520000 -f 10000 -q 10000 -d AMRules_YearPredictionMSD_01.txt.csv -o AMRules_YearPredictionMSD_01.pred
AMRules	8.062	12.845	81.15	00:03:58	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -c 0.02 -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m

					rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 520000 -f 10000 -q 10000 -d AMRules_YearPredictionMSD_01.txt.csv -o AMRules_YearPredictionMSD_01.pred -a 0.25
AMRules	8.102	12.81	1.07	0.001389	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -c 0.0 -t 0.025 -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -U -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 200000000 -f 10000 -q 10000 -d AMRules_MSD_0120_2.txt -a 0.12
AMRules	8.175	12.819	3.08	00:02:43	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -H SeqDrift1ChangeDetector -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 600000 -f 15000 -q 15000 -d AMRules_Music_preq_201.txt -a 0.05
FIMTDD	9.314	13.881	320.84	00:05:03	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d FIMTDD_Music_1122_04.txt -w 10000 -a 0.03
FIMTDD	9.663	14.086	486.72	00:04:13	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -c 0.021 -t 0.07 -a 0.016 -h 10) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 114000000 -f 15000 -q 15000 -d FMITDD_Music_p_206.txt -a 0.05
FIMTDD	9.833	14.15	399.79	00:05:03	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -c 0.05 -a 0.05 -h 25) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 114000000 -f 10000 -q 10000 -d FMITDD_Music_p_204.txt -a 0.025
FIMTDD	10.234	15.81	320.84	00:04:59	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 2000) -i 114000000 -f 10000 -q 10000 -d FMITDD_Music_p_203.txt -a 0.025
FIMTDD	10.234	15.81	320.84	00:04:36	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 2000) -i 114000000 -f 10000 -q 10000 -d FMITDD_Music_p_202.txt -w 2000 -a 0.025
FIMTDD	10.234	15.81	320.84	00:04:38	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 2000) -i 114000000 -f 10000 -d FMITDD_Music_p_201.txt -w 2000 -a 0.025
FIMTDD	10.451	14.875	321.52	00:03:08	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -c 0.021 -t 0.07 -a 0.016 -h 10) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 114000000 -f 15000 -q 15000 -d FMITDD_Music_p_207.txt -a 0.05
FIMTDD	10.91	15.432	470.6	00:05:45	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -c 0.021 -h 100) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 114000000 -f 10000 -q 10000 -d FMITDD_Music_p_205.txt -a 0.025
FIMTDD	11.914	190.741	320.84	0.004861	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -i 520000 -f 10000 -q 10000 -d FIMTDD_YearPredictionMSD_01_PB.csv
FIMTDD	12.708	98.414	34.87	00:01:43	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d FIMTDD_Music_1122_02.txt -w 10000 -a 0.03

FIMTDD	12.708	98.414	34.87	00:01:43	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d FIMTDD_Music_1122_02.txt -w 10000 -a 0.03
FIMTDD	12.822	90.704	34.87	00:02:06	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -f 10000 -q 10000 -d FIMTDD_Music_1128_01.txt
FIMTDD	12.822	90.704	34.87	00:01:54	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 1000000 -f 10000 -q 10000 -o FIMTDD_INSURANCE_P.PRED.pred
FIMTDD	12.881	17.82	574.9	00:18:53	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.075 -a 0.042 -h 25 -f 0.902 -l 0.035 -d 0.025) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 6000000 -f 5000 -q 10000 -d FIMTDD_music_1001.txt -a 0.025
FIMTDD	13.488	107.741	34.87	00:01:35	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -f 10000 -q 10000 -d FIMTDD_Music_1128_02.txt
FIMTDD	14.678	20.468	958.64	00:14:10	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.075 -a 0.042 -f 0.902 -l 0.035 -d 0.025) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 6000000 -f 10000 -d FIMTDD_music_1003.txt -a 0.025
FIMTDD	14.678	20.468	720.78	00:02:52	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.075 -a 0.042 -f 0.902 -l 0.035 -d 0.025) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 6000000 -d FIMTDD_music_1002.txt -a 0.025
FIMTDD	21.098	155.61	49.93	00:02:24	EvaluatePrequentialRegression -l (trees.FIMTDD -s InfoGainSplitCriterion -t 0.055 -a 0.015 -e) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 1000000 -f 10000 -q 10000 -o FIMTDD_INSURANCE_P.PRED.pred
FIMTDD	27.022	1,763.37	163.43	00:02:02	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.075 -a 0.042 -f 0.902 -l 0.135 -d 0.225) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -i 6000000 -d FIMTDD_music_1004.txt -a 0.025
IBLStreams	6.855	10.985	10.46	00:44:36	EvaluateInterleavedTestThenTrain -l (IBLStreams -i 10000 -s WMeanReg -w GaussianKernel -m 10000 -j 28 -k 256 -l 800 -q 0.4989753) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e WindowRegressionPerformanceEvaluator -i 520000 -f 10000 -q 10000 -d IBLStreams_holdout_music_03.txt
IBLStreams	1,997.39	1,997.43	384.08	00:01:01	EvaluatePrequential -l (IBLStreams -i 600000 -s WMeanReg -a AdaptSigma -w GaussianKernel -m 600000 -j 16 -k 128 -l 512) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -f 15000 -q 15000 -d IBLStreams_music_basic_100.txt -a 0.049999999999999996
SFNRegressor FIMTDD	9.739	13.995	480.36	00:10:48	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -c 0.01 -a 0.025 -y 250 -l 0.035 -d 0.005)) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -i 600000 -f 15000 -q 15000 -d SFN_F_Music_preq_203.txt -a 0.05
SFNRegressor FIMTDD	10.708	47.084	124.73	00:07:07	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -f 10000 -q 10000 -d SFN_FIMTDD_Music_1128_04.txt
SFNRegressor FIMTDD	13.866	92.692	124.73	00:06:35	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -f 10000 -q 10000 -d FIMTDD_Music_1128_03.txt
SFNRegressor FIMTDD	14.47	96.855	124.73	00:06:35	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d SFN_F_Music_1122_02.txt -w 10000 -a 0.03
SFNRegressor FIMTDD	14.47	96.855	124.73	00:06:35	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d SFN_F_Music_1122_02.txt -w 10000 -a 0.03

SFNRegressor FIMTDD	98.892	12,290.42	480.35	00:11:06	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -c 0.01 -a 0.025 -y 250 -l 0.035 -d 0.005)) -s (ArffFileStream -f YearPredictionMSD.arff -c 1) -e BasicRegressionPerformanceEvaluator -i 600000 -f 15000 -q 15000 -d SFN_F_Music_preq_208.txt -a 0.05

Fonte: Elaborador pelo autor.

Os experimentos com resultados não considerados bons com a base de dados AirlinesL2Y são apresentados no Quadro 5.

Quadro 5: Experimentos não selecionados no trabalho utilizando a base AirlinesL2Y original.

Algorithm Name	Current MAE	Current RMSE	Memory (MB)	Time	Command Line
AMRules	11.394	17.72	25.73	00:07:40	EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O StdDevThreshold) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -f 50000 -d AMRules_Airlines08_full_11.txt -o AMRules_Airlines08_full_11.pred -a 0.055
FIMTDD	11.645	17.868	8.31	00:00:43	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f airlines2008.arff) -i 10000000 -d FIMTDD_Airlines08_0221_002.txt -a 0.03
FIMTDD	11.778	17.922	16.46	00:02:04	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d FIMTDD_Airlines08_full_1015_01.txt -a 0.055
FIMTDD	11.778	17.922	16.48	00:02:13	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d FIMTDD_Airlines08_full_10.txt -o FIMTDD_Airlines08_full_10.pred -a 0.055
FIMTDD	12.863	19.127	15.68	00:02:13	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -l 0.031) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d FIMTDD_Airlines08_full_11.txt -o FIMTDD_Airlines08_full_11.pred -w 5000 -a 0.055
FIMTDD	14.883	19.846	7.74	00:01:29	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.035 -l 0.030000000000000002) -s (ArffFileStream -f airlines2008.arff) -i 200000000 -d FIMTDD_airlines_2008_1021_05.txt -a 0.049999999999999996
FIMTDD	15.081	23.265	7.74	00:01:30	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.035 -l 0.030000000000000002) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -i 200000000 -d FIMTDD_airlines_2008_1021_04.txt -a 0.049999999999999996
FIMTDD	15.72	21.544	7.74	00:01:32	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.035 -l 0.030000000000000002) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 200000000 -d FIMTDD_airlines_2008_1021_03.txt -a 0.049999999999999996
FIMTDD	15.945	20.383	2.55	00:01:56	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f airlines2008.arff) -d FIMTDD_Airlines08_100k_1028_02.txt
FIMTDD	15.945	20.383	2.55	00:02:18	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f airlines2008.arff) -f 10000 -q 10000 -d FIMTDD_Airlines08_10k_1028_01.txt
FIMTDD	16.514	22.019	2.55	00:01:31	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i

					200000000 -d FIMTDD_airlines_2008_1020_01.txt -a 0.049999999999999996
FIMTDD	19.314	24.68	2.49	00:01:35	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -t 0.035 -l 0.030000000000000002 -d 0.005) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 200000000 -d C -a 0.049999999999999996
FIMTDD	20.023	36.003	8.29	00:00:48	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s (ArffFileStream -f airlines2008.arff) -e BasicRegressionPerformanceEvaluator -i 10000000 -d FIMTDD_Airlines08_0221_001.txt -a 0.03
FIMTDD	33.739	39.092	2.3	00:01:38	EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -d 0.05) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 200000000 -d FIMTDD_airlines_2008_1021_01.txt -a 0.049999999999999996
IBLStreams	13.644	18.759	917.08	00:06:17	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -m 6000000 -j 8 -l 64) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_airlines2008_1106_09.txt
IBLStreams	13.644	18.759	917.1	00:06:16	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -w GaussianKernel -m 6000000 -j 8 -l 64) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_airlines2008_1106_11.txt
IBLStreams	13.644	18.759	917.1	00:06:19	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -w exponentialKernel -m 6000000 -j 8 -l 64) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_airlines2008_1106_10.txt
IBLStreams	13.644	18.759	917.1	00:06:24	EvaluatePrequential -l (IBLStreams -i 6000000 -s LocLinReg -m 6000000 -j 8 -l 64) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_airlines2008_1106_12.txt
IBLStreams	13.644	18.759	917.08	00:06:36	EvaluatePrequential -l (IBLStreams -i 6000000 -s LocLinReg -a AdaptSigma -w GaussianKernel -m 6000000 -k 32 -l 56) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_airlines2008_1107_01.txt -a 0.03
IBLStreams	13.644	18.759	917.1	00:07:33	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -w GaussianKernel -m 10000 -j 56 -k 512 -l 1024 -q 10.004500000000002) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_Airlines08_11.txt -o IBLStreams_Airlines08_11.pred
IBLStreams	13.644	18.759	917.08	00:07:55	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -a AdaptSigma -w GaussianKernel -m 10000 -j 56 -k 512 -l 512) -s (ArffFileStream -f airlines2008.arff) -e WindowRegressionPerformanceEvaluator -i 6000000 -d IBLStreams_Airlines08_10.txt -o IBLStreams_Airlines08_10.pred
IBLStreams	14.772	20.918	917.08	00:06:03	EvaluatePrequential -l (IBLStreams -i 14000000 -s WMeanReg -a AdaptSigma -w GaussianKernel -m 14000000 -j 16 -k 64 -l 128) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 14000000 -d IBLStreams_airlines08_1106_07.txt -w 5000 -a 0.09 -b 1100
IBLStreams	14.772	20.918	917.08	00:06:03	EvaluatePrequential -l (IBLStreams -i 14000000 -s WMeanReg -a AdaptSigma -w GaussianKernel -m 14000000 -j 16 -k 64 -l 128) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 14000000 -d IBLStreams_airlines08_1106_07.txt -w 5000 -a 0.09 -b 1100
IBLStreams	14.772	20.918	917.08	00:06:16	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -w exponentialKernel -m 6000000 -j 8 -k 32 -l 56) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 6000000 -b 1200 -d IBLStreams_airlines2008_1106_13.txt -w 5000 -a 0.05
IBLStreams	14.772	20.918	917.1	00:06:20	EvaluatePrequential -l (IBLStreams -i 6000000 -s WMeanReg -w exponentialKernel -m 6000000 -j 8 -k 32 -l 56) -s (ArffFileStream -f airlines2008.arff) -e

					(WindowRegressionPerformanceEvaluator -w 5000) -i 6000000 -b 1200 -d IBLStreams_airlines2008_1106_14.txt -w 5000 -a 0.009
IBLStreams	14.772	20.918	917.08	00:06:30	EvaluatePrequential -l (IBLStreams -i 14000000 -s WMeanReg -a AdaptSigma -w GaussianKernel -m 14000000 -j 16 -k 64 -l 128) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 5000) -i 14000000 -d IBLStreams_airlines08_1106_06.txt -w 5000 -a 0.05 -b 1100
SFNRegressor AMRules	11.641	17.777	150.66	00:53:18	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (rules.AMRulesRegressor -g 400 -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l rules.multilabel.functions.MultiLabelTargetMeanRegressor -m rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p CantellisInequality) -O SelectAllOutputs) -k 6) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d SFNRegressor_AMRules_Airlines08_full_200.txt -o SFNRegressor_AMRules_Airlines08_full_200.pred -w 5000 -a 0.055
SFNRegressor FIMTDD	11.421	18.093	129.76	00:09:54	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion)) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d SFNRegressorF_Airlines08_full_10.txt -o SFNRegressorF_Airlines08_full_10.pred -w 5000 -a 0.055
SFNRegressor FIMTDD	11.583	17.674	121.37	00:09:43	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -k 6) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d SFNRegressor_FIMTDD_Airlines08_2001_01.txt -w 5000 -a 0.055
SFNRegressor FIMTDD	13.082	18.441	110.8	00:11:11	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.ORTO -s VarianceReductionSplitCriterion)) -s (ArffFileStream -f airlines2008.arff) -i 6000000 -d SFNRegressor_ORTO_Airlines08_full_10.txt -o SFNRegressor_ORTO_Airlines08_full_10.pred -w 5000 -a 0.055
SFNRegressor FIMTDD	13.225	22.447	104.2	00:06:30	EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -m Pagerank) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -d SFN_F_airlines2008_1121_06.txt -w 10000 -a 0.03
SFNRegressor FIMTDD	13.62	22.928	96.02	00:05:08	EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f airlines2008.arff) -e (WindowRegressionPerformanceEvaluator -w 10000) -d SFN_F_airlines2008_1121_07.txt -w 10000 -a 0.03

Fonte: Elaborador pelo autor.

APÊNDICE C LINHAS DE COMANDO UTILIZADOS NOS EXPERIMENTOS

Linhas de comandos utilizadas para as experimentações com a base Airlines:

Quadro 6: Linhas de comandos utilizados dentro do MOA para os experimentos com a base Airlines.

```
# SFN AMRULES
EvaluatePrequentialRegression -l (social.SFNRegressor -l (rules.AMRulesRegressor -L
     (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l
     rules.multilabel.functions.MultiLabelTargetMeanRegressor -m
     rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p
     CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f airlines_full_n.arff) -i
     200000000 -d SFN_AMRules_airlines_n_full_100401.txt

# FIMT-DD
EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s
     (ArffFileStream -f airlines_full_n.arff) -i 115000000 -d FIMTDD_airlines_full_n_1126_03.txt

# SFN FIMT-DD
EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s
     VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f airlines_full_n.arff) -i 115000000 -d
     SFN_FIMTDD_airlines_full_n_1126_05.txt

# AMRULES
EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L
     (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l
     rules.multilabel.functions.MultiLabelTargetMeanRegressor -m
     rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p
     CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f airlines_full_n.arff) -e
     (WindowRegressionPerformanceEvaluator -w 2000) -i 114000000 -d
     AMRules_airlines_full_numeric_010.txt -w 2000 -a 0.025
```

Fonte: Elaborador pelo autor.

Linhas de comandos utilizadas para as experimentações com base Fried:

Quadro 7: Linhas de comandos utilizados dentro do MOA para os experimentos com a base Fried.

```
# AMRules
EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L
     (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l
     rules.multilabel.functions.MultiLabelTargetMeanRegressor -m
     rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p
     CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f Fried.arff) -d
     AMRules_FriedBasic3_1128_05.txt -a 0.05

# FIMT-DD
EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -e) -s
     (ArffFileStream -f Fried.arff) -d FIMTDD_FriedBasic3_1128_08.txt -a 0.05

# IBLSTREAMS
EvaluatePrequential -l (IBLStreams -i 5000000 -s WMeanReg -w GaussianKernel -m 5000000) -s
     (ArffFileStream -f Fried.arff) -e WindowRegressionPerformanceEvaluator -b 2000 -d
     IBLStreams_friedman5_c5_1125_05.txt -a 0.03

# SFN AMRules
EvaluatePrequentialRegression -l (social.SFNRegressor -l (rules.AMRulesRegressor -L
     (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l
     rules.multilabel.functions.MultiLabelTargetMeanRegressor -m
     rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p
     CantellisInequality) -O SelectAllOutputs)) -s (ArffFileStream -f Fried.arff) -d
     SFN_AMRules_FriedBasic3_1128_04.txt -a 0.05

# SFN FIMTDD
EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -l (trees.FIMTDD -s
     VarianceReductionSplitCriterion -e)) -s (ArffFileStream -f Fried.arff) -e
     (WindowRegressionPerformanceEvaluator -w 10000) -i 115000000 -d
     SFN_FIMTDD_Synthetic1_0120_05.txt -w 10000 -a 0.05
```

Fonte: Elaborador pelo autor.

Linhas de comandos utilizadas para as experimentações com base YearPredictionMSD:

Quadro 8: Linhas de comandos utilizados dentro do MOA para os experimentos com a base YearPredictionMSD.

```
# AMRULES
EvaluatePrequentialRegression -l (rules.AMRulesRegressor -L
     (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l
     rules.multilabel.functions.MultiLabelTargetMeanRegressor -m
     rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p
     CantellisInequality) -O SelectAllOutputs) -s (ArffFileStream -f YearPredictionMSD_dup.arff -c
     1) -e (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d
     AMRules_Music_dup_0120_05.txt -a 0.06

# SFN FIMT-DD
EvaluatePrequentialRegression -l (trees.FIMTDD -s VarianceReductionSplitCriterion -l 0.05 -d 0.02)
     -s (ArffFileStream -f YearPredictionMSD_dup.arff -c 1) -i 200000000 -f 10000 -q 10000 -d
     SFN_FIMTDD_MSD_dup_1020_02.txt -a 0.049999999999999996

# IBLSTREAMS
EvaluatePrequential -l (IBLStreams -i 20000 -s WMeanReg -w GaussianKernel -m 10000 -j 40 -k 512 -l
     800 -q 0.4999752500000001) -s (ArffFileStream -f YearPredictionMSD_dup.arff -c 1) -e
     WindowRegressionPerformanceEvaluator -i 1200000 -f 10000 -q 10000 -d
     IBLStreams_YearPredictionMSD_dup_1110_01.csv -a 0.052

# SFN AMRules
EvaluatePrequentialRegression -l (social.SFNRegressor -d ADWIN -e 0.049999999999999996 -l
     (rules.AMRulesRegressor -L (rules.multilabel.functions.AdaptiveMultiTargetRegressor -l
     rules.multilabel.functions.MultiLabelTargetMeanRegressor -m
     rules.multilabel.functions.MultiLabelPerceptronRegressor) -A (OddsRatioScore -p
     CantellisInequality) -O SelectAllOutputs) -k 35) -s (ArffFileStream -f
     YearPredictionMSD_dup.arff -c 1) -i 200000000 -f 10000 -q 10000 -d
     SFN_AMRules_MSD_dup_1010_1.txt -a 0.09

# FIMTDD
EvaluatePrequentialRegression -l (social.SFNRegressor -l (trees.FIMTDD -s
     VarianceReductionSplitCriterion)) -s (ArffFileStream -f YearPredictionMSD_dup.arff -c 1) -e
     (WindowRegressionPerformanceEvaluator -w 10000) -f 10000 -q 10000 -d
     SFN_F_MusicDup_1122_12.txt -a 0.05
```

Fonte: Elaborador pelo autor.