UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Etienne da Silva Julia

Um estudo comparativo entre algoritmos de classicação para Fluxos Contínuos de Dados

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Etienne da Silva Julia

Um estudo comparativo entre algoritmos de classicação para Fluxos Contínuos de Dados

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ci ncia da Computação.

Orientador: Prof^a Dr^a Elaine Ribeiro de Faria Paiva

Universidade Federal de Uberlândia UFU

Faculdade de Computação

Bacharelado em Ci ncia da Computação

Uberlândia, Brasil

2018

Etienne da Silva Julia

Um estudo comparativo entre algoritmos de classicação para Fluxos Contínuos de Dados

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Ci ncia da Computação.

Prof[a] Dr[a] Elaine Ribeiro de Faria Paiva
Orientadora

Prof[a] Dr[a] Maria Adriana Vidigal de Lima

Prof Dr Rodrigo Sanches Miani

Uberlândia, Brasil
2018

Resumo

A proposta desse projeto de pesquisa é a realização de um estudo analítico e comparativo entre alguns algoritmos de classicação que já existem em uxos contínuos de dados (do ingl s data streams), com o objetivo de julgar e determinar as vantagens e desvantagens de cada um deles para bases de dados reais e articiais. A comparação foi realizado por meio da ferramenta MOA(*Massive Online Analysis*), que é, atualmente, o software *open source* mais utilizado para mineração de uxos contínuos de dados. Os resultados dos experimentos indicaram que a incorporação de medidas para tratar as especicidades associadas à tarefa de classicação em Fluxos Contínuos de Dados melhora signicativamente o desempenho de um classicador, tal como corroborado pelos resultados obtidos pelo algoritmo **Very Fast Decision Tree**.

Palavras-chave: Mineração de Dados, Classicação, Algoritmo, Fluxos Contínuo de Dados, Comparação.

Lista de ilustrações

Figura 1 Exemplo Árvore de Decisão 21
Figura 2 Árvore de decisão 1: representa o desempenho dos funcionários de uma empresa com base nos atributos sexo, experi ncia e escolaridade. . . . 22
Figura 3 Árvore de decisão 2: representa o desempenho dos funcionários de uma empresa com base nos atributos experi ncia e escolaridade. 23
Figura 4 Árvore de decisão 3: representa o desempenho dos funcionários de uma empresa com base nos atributos escolaridade e experi ncia. 25
Figura 5 KNN com diferentes valores de **K** 29
Figura 6 **KNN** com **K** muito grande 30
Figura 7 Janela deslizante do KNN 35
Figura 8 **MOA**: Interface do Classicador 38
Figura 9 **MOA**: Conguração 39
Figura 10 Grácos dos resultados do *Evaluate Prequential* para bases reais. 45
Figura 11 Grácos dos resultados do *Interleaved Chunks* para bases reais. 47
Figura 12 Grácos dos resultados do *evaluate prequential* para bases articiais com poucos atributos. 50
Figura 13 Grácos dos resultados do *Interleaved Chunks* para bases articiais com poucos atributos. 52
Figura 14 Grácos dos resultados do *Evaluate Prequential* para bases articiais com muitos atributos. 55
Figura 15 Grácos dos resultados do *Interleaved Chunks* para bases articiais com muitos atributos. 57
Figura 16 Grácos dos resultados do *Evaluate Prequential* para bases articiais com ruído e/ou mudança de conceito. 60
Figura 17 Grácos dos resultados do *Interleaved Chunks* para bases articiais com ruído e/ou mudança de conceito. 62

Lista de tabelas

Tabela 1 Animais vertebrados, adaptada de (TAN; STEINBACH; KUMAR, 2005) 20
Tabela 2 Funcionários da empresa 22
Tabela 3 Ramo 1 23
Tabela 4 Ramo 2 23
Tabela 5 Ramo 3 23
Tabela 6 Tabela de Jogo 27
Tabela 7 Exemplo dia 27
Tabela 8 P[Ensolarado|Sim] 27
Tabela 9 Classicação dia E 28
Tabela 10 Características das bases reais 43
Tabela 11 Características das bases articiais 44
Tabela 12 Tabela com os resultados da medida de avaliação **precisão** nas bases reais, usando o método *evaluate prequential*. Resultados em % 46
Tabela 13 Tabela com os resultados da medida de avaliação **kappa** nas bases reais, usando o método *evaluate prequential*. Resultados em % 46
Tabela 14 Tabela com os resultados da medida de avaliação tempo nas bases reais, usando o método *evaluate prequential*. Resultados em segundos 46
Tabela 15 Tabela com os resultados da medida de avaliação **precisão** nas bases reais, usando o método *interleaved chunks*. Resultados em % 46
Tabela 16 Tabela com os resultados da medida de avaliação *kappa* nas bases reais, usando o método *interleaved chunks*. Resultados em % 48
Tabela 17 Tabela com os resultados da medida de avaliação tempo nas bases reais, usando o método *interleaved chunks*. Resultados em % 48
Tabela 18 Tabela com os resultados da medida de avaliação precisão nas bases articiais com poucos atributos, usando o método *evaluate prequential*. Resultados em % 49
Tabela 19 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com poucos atributos, usando o método *evaluate prequential*. Resultados em % 51
Tabela 20 Tabela com os resultados da medida de avaliação tempo nas bases articiais com poucos atributos, usando o método *evaluate prequential*. Resultados em segundos 51
Tabela 21 Tabela com os resultados da medida de avaliação precisão nas bases articiais com poucos atributos, usando o método *interleaved chunks*. Resultados em % 51

Tabela 22 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com poucos atributos, usando o método *interleaved chunks*. Resultados em % 53
Tabela 23 Tabela com os resultados da medida de avaliação tempo nas bases articiais com poucos atributos, usando o método *interleaved chunks*. Resultados em segundos 53
Tabela 24 Tabela com os resultados da medida de avaliação precisão nas bases articiais com muitos atributos, usando o método *evaluate prequential*. Resultados em % 54
Tabela 25 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com muitos atributos, usando o método *evaluate prequential*. Resultados em % 54
Tabela 26 Tabela com os resultados da medida de avaliação tempo nas bases articiais com muitos atributos, usando o método *evaluate prequential*. Resultados em % 56
Tabela 27 Tabela com os resultados da medida de avaliação precisão nas bases articiais com muitos atributos, usando o método *interleaved chunks*. Resultados em % 56
Tabela 28 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com muitos atributos, usando o método *interleaved chunks*. Resultados em % 56
Tabela 29 Tabela com os resultados da medida de avaliação tempo nas bases articiais com muitos atributos, usando o método *interleaved chunks*. Resultados em segundos 58
Tabela 30 Tabela com os resultados da medida de avaliação precisão nas bases articiais com mudança de conceito e/ou ruído, usando o método *evaluate prequential*. Resultados em % 59
Tabela 31 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com mudança de conceito e/ou ruído, usando o método *evaluate prequential*. Resultados em % 59
Tabela 32 Tabela com os resultados da medida de avaliação tempo nas bases articiais com mudança de conceito e/ou ruído, usando o método *evaluate prequential*. Resultados em segundos 59
Tabela 33 Tabela com os resultados da medida de avaliação precisão nas bases articiais com mudança de conceito e/ou ruído, usando o método *interleaved chunks*. Resultados em % 61
Tabela 34 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com mudança de conceito e/ou ruído, usando o método *interleaved chunks*. Resultados em % 61

Tabela 35 Tabela com os resultados da medida de avaliação tempo nas bases artificiais com mudança de conceito e/ou ruído, usando o método *interleaved chunks*. Resultados em segundos 61
Tabela 36 *Rank* dos algoritmos para bases reais em relação ao valor *kappa* 64
Tabela 37 Resultados do teste de signicância estatística dos algoritmos para bases reais em relação ao valor *kappa* 64
Tabela 38 *Rank* dos algoritmos em relação ao valor *kappa* 65
Tabela 39 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa* 65
Tabela 40 *Rank* dos algoritmos em relação ao valor *kappa* 66
Tabela 41 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa* 66
Tabela 42 *Rank* dos algoritmos em relação ao valor *kappa* 67
Tabela 43 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa* 67
Tabela 44 *Rank* dos algoritmos em relação ao valor *kappa* 68
Tabela 45 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa* 68

Lista de abreviaturas e siglas

Sumário

1 INTRODUÇÃO . . . 12
1.1 Contextualização . . . 12
1.2 Justicativa . . . 13
1.3 Objetivos . . . 14
1.3.1 Gerais . . . 14
1.3.2 Especícos . . . 14
1.4 Organização do trabalho . . . 14

2 REVISÃO BIBLIOGRÁFICA . . . 16
2.1 Introdução . . . 16
2.2 Denições Importantes . . . 16
2.3 Mineração de Dados . . . 16
2.4 Fluxos Contínuos de Dados . . . 18
2.5 Classicação . . . 18
2.6 Classicação em cenários estáticos . . . 19
2.6.1 Árvore de Decisão . . . 20
2.6.2 *Naive Bayes* . . . 25
2.6.3 AdaBoost . . . 28
2.6.4 K vizinhos mais próximos . . . 29
2.6.5 Classicador ativo . . . 30
2.7 Classicação em Fluxos Contínuos de Dados . . . 31
2.7.1 Mudança de conceito . . . 31
2.7.2 Algoritmos para classicação em uxos contínuos de dados . . . 32
2.7.2.1 Very Fast Decision Tree . . . 32
2.7.2.2 Naive Bayes . . . 33
2.7.2.3 OzaBoost Adwin . . . 34
2.7.2.4 K vizinhos mais próximos . . . 35
2.7.2.5 Classicador ativo . . . 35

3 METODOLOGIA . . . 36
3.1 Introdução . . . 36
3.2 Algoritmos Selecionados . . . 36
3.2.1 Naive Bayes . . . 36
3.2.2 K-Nearest Neighbors . . . 36
3.2.3 Active Classier . . . 36
3.2.4 OzaBoost Adwin . . . 37

3.2.5 Hoeding Tree . . . 37
3.3 Massive Online Analysis Framework . . . 37
3.3.1 Processamento dos dados . . . 39
3.4 Bases de dados . . . 40
3.4.1 Bases Reais . . . 40
3.4.2 Bases Articiais . . . 40
3.5 Estratégias de avaliação . . . 41
3.6 Medidas de avaliação . . . 41
3.7 Teste estatístico . . . 41
3.7.1 Friedman test . . . 41
3.8 Considerações nais . . . 42

4 EXPERIMENTOS E RESULTADOS . . . 43
4.1 Introdução . . . 43
4.2 Bases de Dados . . . 43
4.3 Experimentos com bases reais . . . 44
4.3.1 Evaluate Prequential . . . 44
4.3.2 Interleaved Chunks . . . 46
4.3.3 Discussão dos resultados . . . 48
4.4 Experimentos com bases articiais . . . 49
4.4.1 Experimentos com bases articiais com poucos atributos . . . 49
4.4.1.1 Evaluate Prequential . . . 49
4.4.1.2 Interleaved Chunks . . . 51
4.4.1.3 Discussão dos resultados . . . 53
4.4.2 Experimentos com bases articiais com muitos atributos . . . 53
4.4.2.1 Evaluate Prequential . . . 54
4.4.2.2 Interleaved Chunks . . . 56
4.4.2.3 Discussão dos resultados . . . 58
4.4.3 Experimentos com bases articiais com ruído e mudança de conceito . . . 58
4.4.3.1 Evaluate Prequential . . . 59
4.4.3.2 Interleaved Chunks . . . 61
4.4.4 Discussão dos resultados . . . 63
4.5 Teste estatístico . . . 63
4.6 Bases Reais . . . 63
4.7 Bases articiais . . . 64
4.7.1 Bases articiais com pouco atributos . . . 65
4.7.2 Bases articiais com muitos atributos . . . 66
4.7.3 Bases com mudança de conceito e/ou ruído . . . 67
4.7.4 Todas as bases . . . 67
4.8 Considerações nais do capítulo . . . 68

5 CONCLUSÃO 69
5.1 Contribuições 69
5.2 Conclusões obtidas 69
5.3 Trabalhos futuros 69

REFER NCIAS 71

1 Introdução

1.1 Contextualização

A mineração de dados é um processo que lida, essencialmente, com a descoberta automática de padrões, bem como com a previsão e a classicação de resultados em grandes conjuntos de dados. Tal processo requer recursos técnicos e cientícos provenientes da Estatística e Intelig ncia Articial (TAN; STEINBACH; KUMAR, 2005), a saber:

Estatística:

Amostragem

Estimativa

Teste de Hipótese

Intelig ncia Articial:

Algoritmos de busca

Técnicas de modelagem

Teorias de aprendizagem da Intelig ncia Articial

Reconhecimento de padrões

Aprendizado de Máquina

Nas últimas tr s décadas, a mineração de dados esteve focada para cenários de lote (*batch*), no qual toda a base de dados está disponível para o sistema desde o início (GAMA, 2010). Entretanto, nos últimos anos surgiu a necessidade de se encontrar novos métodos de processamento e armazenamento de dados que operem em ambientes dinâmicos. Isso ocorreu devido ao fato de que algumas novas aplicações geram suas saídas em tempo real, na forma de um uxo de dados com tamanho possivelmente innito (GOLAB; ÖZSU, 2003).

Diante dessa nova necessidade, surgiu o estudo sobre mineração em uxos contínuos de dados (*Data Streams*). Essa área de pesquisa trabalha com algoritmos que possuem mecanismos de aprendizado incremental, capazes de incorporar mudança de conceito e de esquecer dados ultrapassados (GAMA, 2010).

Um uxo contínuo de dados é caracterizado por uma sequ ncia de elementos, possivelmente innita, recebida pelo sistema ao longo do tempo, sem que se tenha controle sobre a ordem e a velocidade de chegada deles (GOLAB; ÖZSU, 2003).

A tarefa de classicação é uma das áreas mais estudadas em Mineração de Dados (FIDELIS; LOPES; FREITAS, 2000). Ela consiste em atribuir a cada elemento de um conjunto de dados uma classe pré-determinada, com base na descoberta de relações entre os atributos dos itens analisados e as classicações já existentes.

Quando se analisa a tarefa de classicação em Fluxos Contínuos de Dados, é importante ressaltar os problemas que surgem ao se trabalhar em um ambiente dinâmico. Além do problema da mudança de conceito, que signica uma mudança na distribuição dos dados, outros podem surgir dependendo do tipo de aplicação em questão, tais como: detecção de novidade, incorporação de novos conhecimentos, velocidade de chegada etc.

Neste cenário, a presente monograa tem como principal objetivo efetuar um estudo comparativo entre alguns dos principais algoritmos de classicação para a mineração de Fluxos Contínuos de Dados.

1.2 Justicativa

No cenário atual, grandes quantidades de dados são geradas e armazenadas em diversas bases de dados a cada segundo. Nesses dados, existem informações que podem ser de suma importância no tratamento de diversos problemas cientícos e práticos. A Mineração de Dados é uma área de estudo que tem como objetivo resgatar tais informações utilizando, como principais métodos, a Classicação e o Agrupamento de Dados (*Clustering*).

Devido à importância da classicação, diversos métodos foram estudados ao longo dos anos para otimizar essa tarefa (principalmente para cenários de lote), tais como: árvores de decisão (TAN; STEINBACH; KUMAR, 2005), *naive bayes* (TAN; STEINBACH; KUMAR, 2005), *k-nearest neighbors* (TAN; STEINBACH; KUMAR, 2005). Entretanto, no caso de uxos contínuos de dados, o grande volume de informações, a inviabilidade de acessar os mesmos dados diversas vezes, a rápida taxa de chegada e o fato de a distribuição dos dados gerados poder mudar rapidamente, fazem com que o processo de aprendizado nesse ambiente dinâmico seja diferente do aprendizado em lote (STEFANOWSKI; BRZEZINSKI, 2017). É um exemplo de algoritmo de classicação desenvolvido para uxos contínuos de dados a VFDT - *Very Fast Decision Tree*. Todos esses algoritmos serão estudados detalhadamente no capítulo 2.

Como atualmente a geração dos dados é majoritariamente automática, diversas aplicações se beneciam de uma modelagem baseada em Fluxos Contínuos de Dados

(GAMA, 2010), tais como:

Monitoramento de Redes

Mineração web

Redes de sensores

Aplicações nanceiras

Os diversos cenários de aplicação tornam importante a exist ncia de estudos comparativos entre os algoritmos de classicação em Fluxos Contínuos de Dados, uma vez que eles permitem que se estime a eci ncia de certos métodos para diferentes cenários de mineração de dados.

1.3 Objetivos

1.3.1 Gerais

Comparar algoritmos para a classicação de uxos contínuos de dados selecionados da ferramente MOA utilizando diversas bases reais e articiais. O objetivo da comparação é determinar, por meio do teste estatístico de *Friedman*, se alguns dos algoritmos selecionados possuem um desempenho signicantemente melhor do que outros com relação às medidas de avaliação adequadas para o contexto.

1.3.2 Especícos

1. Gerar bases de dados articiais usando o gerador de dados do *framework* MOA, a m de simular diferentes cenários de uxos contínuos de dados a serem utilizados nos experimentos.
2. Determinar empiricamente os aspectos positivos e negativos de cada algoritmo analisado utilizando de bases de dados reais e articiais.
3. Obter um contraste entre os resultados obtidos com bases reais e articiais.
4. Usar testes estatísticos para determinar o nível de signicância nos resultados obtidos usando as bases de dados.

1.4 Organização do trabalho

O capítulo 2 apresenta e explica os fundamentos teóricos que foram necessários para a realização deste trabalho.

No capítulo 3 são descritas as ferramentas que foram utilizadas para a realização dos experimentos conduzidos neste trabalho, assim como as bases de dados e parâmetros de avaliação que foram selecionados.

O capítulo 4 apresenta os experimentos realizados e seus resultados e o capítulo 5 as conclusões obtidas, assim como as contribuições feitas e propostas para trabalhos futuros.

2 Revisão Bibliográca

2.1 Introdução

Neste capítulo, serão explicados os fundamentos teóricos envolvidos na realização desta monograa. Os tópicos a serem apresentados são: denição de Fluxos Contínuos de Dados e Classicação, sendo que esta última incluirá a descrição dos algoritmos aqui selecionados para lidar, tanto com o cenários de lote, quanto com o de Fluxos Contínuos de Dados.

2.2 Denições Importantes

Nesta seção serão estabelecidas algumas denições que serão utilizadas ao longo desta monograa.

Treinamento: no contexto deste trabalho, o treinamento corresponde à tarefa de testar um algoritmo sobre um conjunto de dados, com o intuito de incorporar os erros e acertos cometidos para melhorar o resultado de futuras execuções.

Teste: o processo de teste é feito, usualmente, após o treinamento e consiste em executar o algoritmo para um novo conjunto de dados e observar os resultados.

Instância: uma instância corresponde a um elemento de um conjunto de dados.

Atributo: um atributo corresponde à uma característica de uma instância. Por exemplo, em uma base de dados de clientes, alguns atributos podem ser: idade, cpf, altura, etc.

Rótulo: o rótulo é o valor de um atributo que se deseja descobrir. Dessa forma, se um algoritmo tem como objetivo determinar a espécie dos animal de uma base de dados, e um Poodle for uma das instâncias dessa base, o rótulo dele será de Canino.

Ruído: o ruído é um termo utilizado para referenciar instâncias que foram corrompidas em uma base de dados e que, por isso, possuem atributos com valores diferentes dos reais.

2.3 Mineração de Dados

A Mineração de Dados consiste, fundamentalmente, na detecção automática de padrões dentro de grandes bases de dados. As tarefas associadas ao processo de mineração

de dados podem ser divididas em duas categorias principais: previsão e descrição. As tarefas de previsão recebem uma instância e tentam prever o valor de algum atributo com base nos demais, enquanto que as tarefas descritivas tentam derivar padrões entre os dados que representem as relações entre eles (TAN; STEINBACH; KUMAR, 2005). Dessas duas categorias é possível descrever as 4 principais tarefas de mineração de dados.

Modelo de previsão: essa tarefa tem como objetivo gerar um modelo para a previsão do valor de uma instância com base em seus atributos, por exemplo, determinar a espécie de um animal com base nas suas características físicas. Os dois principais tipos de tarefas de previsão são a **Classicação** , que será abordada em mais detalhes na Seção 2.5, e a **Regressão**. A regressão consiste fundamentalmente em um processo que tenta prever, ou estimar, resultados com base em uma função matemática, enquanto a classicação busca atribuir rótulos a elementos desconhecidos, ou seja, determinar a qual grupo (classe) cada elemento pertence.

Análise por associação: visa encontrar os padrões mais interessantes entre os elementos de um conjunto de dados da maneira mais eciente possível.

Análise de grupos (*Clusters*): visa agrupar um conjunto de dados com base nos valores de atributos, de forma que elementos mais similares estejam mais perto uns dos outros e, dessa forma, façam parte do mesmo grupo.

Detecção de Anomalia: tem como objetivo encontrar instâncias em um conjunto que possuam características muito diferentes dos demais (anomalias) enquanto evita rotular um exemplo normal como uma anomalia. Exemplo: detecção de *spam*.

Outro aspecto importante a ser ressaltado no âmbito da mineração de dados é a distinção entre métodos supervisionados e métodos não-supervisionados. No aprendizado supervisionado, os agentes observam exemplos de treino rotulados e, a partir deles, tentam gerar uma modelagem dos dados para mapear a entrada e saída de novas instâncias. Já no aprendizado não-supervisionado, os agentes devem descobrir padrões entre dados sem receber assist ncia do sistema, como é feito na tarefa de agrupamento (RUSSELL; NORVIG, 2009).

As tarefas referentes à mineração de dados podem ser realizadas em dois tipos de cenários diferentes: o estático e o dinâmico. No estático, o conjunto de dados está previamente disponível para os algoritmos de treinamento, o que caracteriza um processamento em lote onde os dados são passíveis de serem processados uma ou mais vezes. No dinâmico, novas informações são introduzias no sistema com o decorrer do tempo (GAMA, 2010). Nesta situação, se tal inserção ocorrer em altas frequ ncias e grandes volumes, o processamento repetido dos mesmos dados pode se tornar inviável. Nessa circunstância,

é imprescindível que o algoritmo seja capaz de incorporar novos dados ao seu modelo de decisão de maneira eciente, de modo a obter um bom desempenho.

2.4 Fluxos Contínuos de Dados

Um Fluxo Contínuo de Dados é um conjunto possivelmente ilimitado de dados que é gerado de maneira sequencial. Essa estrutura é extremamente relevante atualmente, pois pode ser utilizada para representar diversos processos que geram dados automaticamente de forma constante e rápida, tais como sistemas de *GPS*, celulares, correio eletrônico, redes de sensores e redes sociais (GAMA, 2010).

Ao se tratar de Fluxos Contínuos de Dados, alguns dos fatores a serem considerados mais relevantes em seu processo de mineração são:

incorporação de novas informações;

detecção de novidade, que possibilita a detecção de novos conceitos e o renamento do modelo para representá-los;

velocidade de chegada, que exige que os sistemas sejam capazes de receber e processar novos dados de maneira adequada;

mudança de conceito (*Concept Drift*) que são mudanças estatísticas que podem ocorrer nos valores das variáveis;

remoção de conceitos ultrapassados, que consiste na capacidade que o modelo tem de descartar informações que ele considerar desatualizadas;

a inviabilidade de processar a mesma informação múltiplas vezes, devido ao grande volume de dados e à restrição de tempo e memória;

frequentemente devem ser minerados de maneira distribuída, ou seja, um sistema pode ter diversos núcleos de processamentos trabalhando em um mesmo uxo (AGGARWAL, 2007);

Conforme introduzido anteriormente, convém salientar que essa monograa abordará, fundamentalmente, a mineração de uxos contínuos de dados efetuada por meio da tarefa de classicação, conforme explicado nas próximas seções.

2.5 Classicação

A tarefa de classicação é um dos aspectos mais estudados na área de Mineração de Dados. Ela consiste em atribuir a cada elemento de um conjunto de dados uma classe

previamente denida. Isso é feito por meio de comparações entre os valores dos atributos dos elementos analisados e as especicidades de cada classe(FIDELIS; LOPES; FREITAS, 2000).

Na atividade de Classicação, é fundamental compreender a relação entre os conceitos de Variância e de viés. A variância mede o quão esparsos estão os dados de um conjunto em relação à média dos valores. A grosso modo, uma alta variância em um conjunto indica que existem muitos elementos nele que são muito diferentes entre si, enquanto uma baixa variância indica que os elementos do conjunto são similares. Em um contexto de classicação, uma alta variância afeta diretamente o desempenho de um modelo de decisão, uma vez que, quanto maior a diferença entre os elementos de uma classe, mais regras são necessárias para representá-la e, quanto mais regras existem no modelo para denir suas classes, maior a chance de novos exemplos não se encaixarem bem em nenhuma delas (problema do *overting*) e, embora um aumento na quantidade de regras diminua o erro para a base de treino, o erro aumenta consideravelmente quando o modelo é aplicado a uma base de teste. Já um alto viés indica o uso de um modelo simples e generalizável, que representa as classes de um conjunto de teste utilizando uma quantidade insuciente de regras, ou seja, as classes são representadas fracamente pelo modelo de decisão (*undertting*), o que causa um aumento muito grande na taxa de erro da classicação, tanto para a base de teste, quanto para a base de treino. Dessa forma, a qualidade dos algoritmos de classicação está diretamente associada a sua habilidade de balancear o viés e a variância.

Outro aspecto importante para o desempenho de um algoritmo classicador é a capacidade de incorporação de novos dados ao seu modelo de decisão. Embora esta funcionalidade já esteja presente em algoritmos estáticos de mineração, como *K-nearest neighbors* e *naive bayes*, em situações onde podem ocorrer mudanças estatísticas nos valores das variáveis a serem previstos pelo modelo (*Concept Drift*), pode ser necessário uma atenção maior para a implementação de mecanismos que permitam a incorporação de novas informações relevantes e o esquecimento de informações desatualizadas (GAMA, 2010).

2.6 Classicação em cenários estáticos

Como os outros processos de mineração de dados, a tarefa de classicação pode ser realizada tanto em cenários estáticos, quanto dinâmicos, contando que sejam feitas as devidas adaptações aos algoritmos utilizados. Nessa seção serão apresentadas, detalhadamente, as versões estáticas dos algoritmos selecionados para este trabalho.

2.6.1 Árvore de Decisão

Uma árvore de decisão classicadora é um grafo em forma de árvore que classica registros em função dos valores de um conjunto de atributos. Assim sendo, a raiz e cada nó interno representam um teste relativo de um dado atributo, ao passo que os nós terminais (folhas) representam as classes envolvidas no problema. Esse tipo de algoritmo é conhecido como *eager learner*, pois utiliza do pré-processamento dos dados para gerar um modelo que é responsável por mapear os atributos de uma instância à uma respectiva classe assim que tal instância é recebida (TAN; STEINBACH; KUMAR, 2005).

A título de exemplo introdutório, considere-se a Tabela 1 contendo registros relativos a animais vertebrados e informando a classe onde cada um se aloca.

Tabela 1 Animais vertebrados, adaptada de (TAN; STEINBACH; KUMAR, 2005)

Nome	Temperatura Corporal	Cobertura da Pele	Parto	Aquático	Aérea	Tem Pernas	Hiberna	Classicação
Humano	Quente	Pelos	Sim	Não	Não	Sim	Não	Mamífero
Píton	Fria	Escamas	Não	Não	Não	Não	Sim	Réptil
Salmão	Fria	Escamas	Não	Sim	Não	Não	Não	Peixe
Baleia	Quente	Pelos	Sim	Sim	Não	Não	Não	Mamífero
Sapo	Fria	Nenhuma	Não	Misto	Não	Sim	Sim	Anfíbio
Dragão-de-komodo	Fria	Escamas	Não	Não	Não	Sim	Não	Réptil
Morcego	Quente	Pelos	Sim	Não	Sim	Sim	Sim	Mamífero
Pombo	Quente	Penas	Não	Não	Sim	Sim	Não	Ave
Gato	Quente	Pelos	Sim	Não	Não	Sim	Não	Mamífero
Tubarão-leopardo	Fria	Escamas	Sim	Sim	Não	Não	Não	Peixe
Tartaruga	Fria	Escamas	Não	Misto	Não	Sim	Não	Réptil
Pinguim	Quente	Penas	Não	Misto	Não	Sim	Não	Ave
Porco-espinho	Quente	Espinhos	Sim	Não	Não	Sim	Sim	Mamífero
Enguia	Fria	Escamas	Não	Sim	Não	Não	Não	Peixe
Salamandra	Fria	Nenhuma	Não	Misto	Não	Sim	Sim	Anfíbio

Na Tabela 1 pode-se observar que os registros possuem diversos atributos e uma classe nal. Percebe-se que exemplos com diferentes valores de atributos podem ser atribuídos a mesma classe. Isso ocorre porque determinados atributos podem ser mais relevantes do que outros na classicação nal. A partir desses dados é possível montar uma árvore de decisão para classicar novos registros com as informações já disponíveis. Na Figura 1 pode-se ver um modelo de árvore de decisão para classicar mamíferos, utilizando **Temperatura Corporal** para separar vertebrados de corpo quente e vertebrados de corpo frio.

Como todos os animais de corpo frio não são mamíferos, é criado um nó folha logo abaixo do atributo **Temperatura Corporal**. Para os vertebrados de corpo quente, o modelo separa os registros por meio do atributo **Parto**, que é utilizado para diferenciar os mamíferos dos demais tipos de animais, que também possuem a temperatura corporal

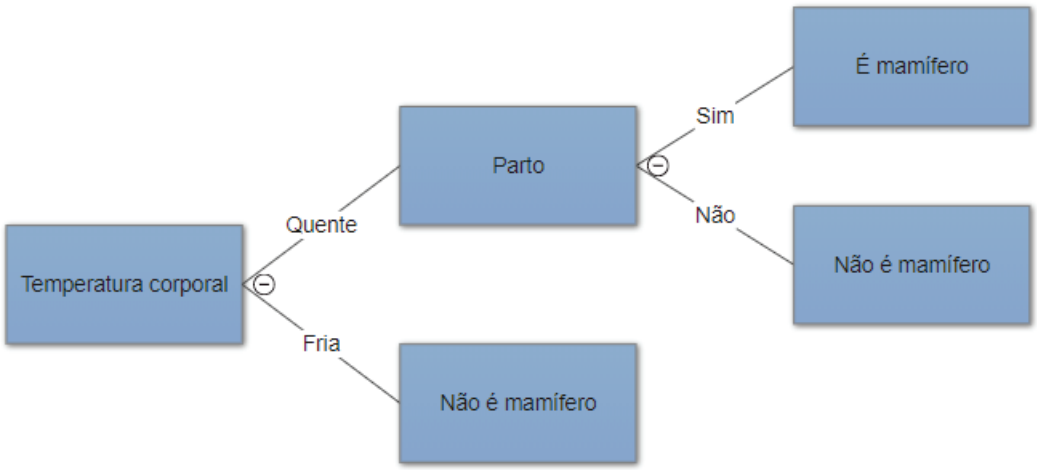


Figura 1 Exemplo Árvore de Decisão

quente.

Um dos aspectos mais importantes em uma árvore de decisão é maneira com que ela é construída, uma vez que os atributos escolhidos para efetuar a ramicação afetam diretamente o tamanho e a complexidade do modelo em questão. Encontrar uma árvore de decisão ótima é computacionalmente inviável, devido ao tamanho exponencial do espaço de pesquisa (TAN; STEINBACH; KUMAR, 2005). Entretanto, é imprescindível que se conte com um método capaz de prover uma ordem apropriada de escolha de atributos que permita a construção de uma árvore de decisão mais simples. A aus ncia de tal método causaria um aumento no custo de construção que poderia vir a inviabilizar o uso desta estrutura em certas situações. Na literatura cientíca, encontram-se diversos algoritmos relevantes para gerar árvores precisas de forma eciente, tais como o IDE3, C4.5, CART e SLIQ (para uma visão detalhada desses, recomenda-se a leitura de (ANYANWU; SHIVA, 2009)). Um critério de ordenação de atributos bastante relevante usado por algoritmos de construção de árvores de decisão (tal como o IDE3) é o conceito conhecido como Entropia.

A Entropia é uma grandeza responsável por medir a desordem, tanto de objetos físicos, quanto de informações. Uma boa maneira de visualizar melhor essa grandeza é imaginar um conjunto de gavetas, onde cada gaveta armazena lápis de uma mesma cor, por exemplo: uma gaveta armazena todos os lápis azuis, outra todos vermelhos, e assim por diante. Em um sistema perfeitamente ordenado se pegarmos um lápis da gaveta responsável por armazenar a cor preta, temos a certeza, mesmo sem olhar, que esse lápis será preto. Nesse caso, a entropia desse sistema é nula, pois tudo está em perfeita ordem. Para esclarecer a maneira com que tal conceito é aplicado na construção de uma árvore de decisão, segue um exemplo adaptado do livro (ARARIBóIA, 1988).

Considere que uma empresa deseja criar uma árvore de decisão que classique o provável desempenho de futuros funcionários, utilizando o sexo, a experi ncia, a escolaridade e o desempenho dos funcionários que já estão na empresa como base.

Tabela 2 Funcionários da empresa

	Escolaridade	Experi ncia	Sexo	Desempenho
1	Superior	Experiente	Feminino	Bom
2	Técnica	Inexperiente	Feminino	Ruim
3	Superior	Experiente	Masculino	Bom
4	Superior	Inexperiente	Masculino	Bom
5	Técnica	Experiente	Masculino	Bom
6	Técnica	Inexperiente	Masculino	Ruim
7	Técnica	Inexperiente	Masculino	Ruim
8	Superior	Inexperiente	Feminino	Bom
9	Colegial	Inexperiente	Masculino	Ruim
10	Colegial	Inexperiente	Feminino	Ruim

Supondo que a escolha da ordem dos atributos seja feita de maneira aleatória, e que os atributos selecionados sejam, na seguinte ordem: sexo, experi ncia e escolaridade. A árvore gerada seria:

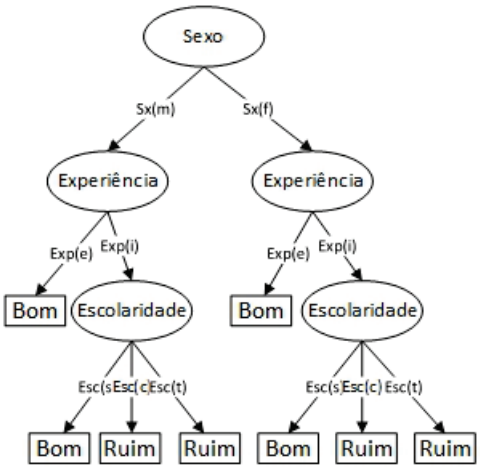


Figura 2 Árvore de decisão 1: representa o desempenho dos funcionários de uma empresa com base nos atributos sexo, experi ncia e escolaridade.

Agora em um segundo caso, seguindo uma ordem diferente de atributos, tal como: Experi ncia, Escolaridade e Sexo:

Percebe-se que tanto a árvore um quanto a dois são capazes de classicar qualquer entrada, entretanto, a segunda árvore necessitou apenas de dois atributos em sua criação, o que faz com que ela seja uma solução melhor do que a primeira. Por isso, é fundamental que se tenha uma maneira de encontrar a melhor solução possível. A incorporação do cálculo da entropia no processo de geração da árvore possibilita que essa solução seja encontrada sem que seja necessário testar todas as possibilidades.

No processo de criação é razoavelmente intuitivo que devem ser escolhidos os atributos que produzam as ramicações com menos desordem, ou seja, os que possuam a menor entropia. Entretanto, para esse m, é necessário o conhecimento de como calcular a entropia de cada possível atributo. Tal cálculo será apresentado e ilustrado aqui por meio da avaliação da entropia do atributo **Escolaridade**. Para tanto, as Tabelas 3, 4 e

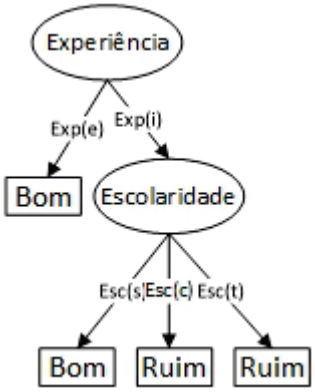


Figura 3 Árvore de decisão 2: representa o desempenho dos funcionários de uma empresa com base nos atributos experi ncia e escolaridade.

5 mostram a distribuição dos registros usados como base, respectivamente, segundo os seguintes valores de tal atributo: Colegial, Superior e Técnica.

Tabela 3 Ramo 1

	Escolaridade colegial		
	Sexo	Experi ncia	Desempenho
9	Masculino	Inexperiente	Ruim
10	Feminino	Inexperiente	Ruim

Tabela 4 Ramo 2

	Escolaridade Superior		
	Sexo	Experi ncia	Desempenho
1	Feminino	Experiente	Bom
3	Masculino	Experiente	Bom
4	Masculino	Inexperiente	Bom
8	Feminino	Inexperiente	Bom

Tabela 5 Ramo 3

	Escolaridade técnica		
	Sexo	Experi ncia	Desempenho
2	Feminino	Inexperiente	Ruim
5	Masculino	Experiente	Bom
6	Masculino	Inexperiente	Ruim
7	Masculino	Inexperiente	Ruim

Pode-se perceber que os ramos 1 (Figura 3) e 2 (Figura 4) possuem apenas indivíduos com o mesmo tipo de desempenho, ou seja, ambos estão perfeitamente ordenados. Isso é o suciente, baseado no que foi denido anteriormente, para se concluir que ambos estes ramos possuem entropia nula. Entretanto, o ramo 5 possui indivíduos tanto de desempenho **Bom**, quanto **Ruim**. Diante disso, a entropia do ramo pode ser determinada como:

$$PD1 \quad log(PD1)/log(2) \quad PD2 \quad log(PD2)/log(2)... \tag{2.1}$$

Onde **PD1** e **PD2** representam a proporção em que cada diagnóstico aparece em um determinado ramo. No caso do ramo 3, pode-se considerar que PD1 = 3/4 e PD2 = 1/4. PD1 representa a proporção de indivíduos com diagnóstico **Ruim** e PD2 a proporção de indivíduos com diagnóstico **Bom**, então, aplicando-se a equação 2.1, tem-se que:

$$(3/4) \quad log(3/4)/log(2) \quad (1/4) \quad log(1/4)/log(2) = 0.81127812 \tag{2.2}$$

Ou seja, a entropia do ramo 3 tem valor 0.81127812. Entretanto, foi calculado apenas a entropia de cada um dos ramos do atributo **Escolaridade** e não do atributo como um todo, de tal forma que Ramo1 = 0, Ramo2 = 0 e Ramo3 = 0.81127812. Para se calcular a entropia desse conjunto basta aplicar a fórmula:

$$N1 \quad E1/Num + N2 \quad E2/Num + N3 \quad E3/Num... \tag{2.3}$$

Seja N1, N2 e N3 a quantidade de registros em cada ramo; Num a quantidade total de registros em questão e E1, E2 e E3 a entropia de cada ramo, o resultado nal da entropia para o atributo **Escolaridade** é:

$$(2 \quad 0)/10 + (4 \quad 0)/10 + (4 \quad 0.81127812)/10 = 0.32451125 \tag{2.4}$$

Agora que se tem a entropia do atributo **Escolaridade**, deve-se realizar o mesmo processo para cada um dos demais atributos da base representada pela Tabela 2 e selecionar os atributos para a criação da árvore, priorizando, sempre, os de menor entropia. Nesse exemplo, o valor resultante para o atributo **Experi ncia** é 0.60418198442 e para o **Sexo** é 1. Então, a árvore de decisão ideal é:

É importante ressaltar que mesmo que a árvore tr s (Figura 4) e a dois (Figura 3) tenham o mesmo tamanho, a última delas ainda é melhor por priorizar o atributo de menor entropia. Nesse caso pode-se pensar que a escolha do atributo **Escolaridade** tinha mais chance de gerar uma árvore menor do que a escolha do atributo **Experi ncia** .

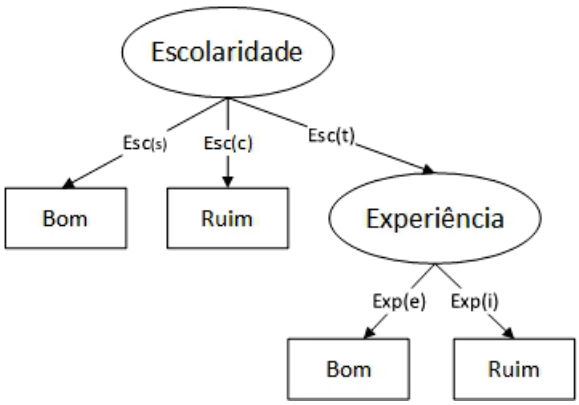


Figura 4 Árvore de decisão 3: representa o desempenho dos funcionários de uma empresa com base nos atributos escolaridade e experi ncia.

2.6.2 *Naive Bayes*

Esta seção visa apresentar a maneira com que o teorema de *Bayes* pode ser utilizado como um classicador de dados. O teorema descreve como calcular a probabilidade de uma hipótese em função de alguma evid ncia. Mesmo sendo um modelo simples, ele é surpreendentemente ecaz quando utilizado como um algoritmo de classicação. Para isso, é necessário que seja feita a suposição de que as probabilidades de ocorr ncia dos atributos sejam independentes entre si, ou seja, o valor de cada atributo não inuencia, nem é inuenciado, por nenhum dos outros. Mesmo que essa suposição normalmente seja falsa, quando aplicada à tarefa de classicação ela permite que o algoritmo apresente um desempenho comparável ao desempenho de classicadores mais sosticados (RISH, 2001).

A fórmula geral do teorema é representada por:

$$P[H|E= \ ] \ P[E|H] \quad P[H]/P[E] \tag{2.5}$$

para a explicação de cada termo e a visualização de como o teorema de *Bayes* pode ser aplicado, considere o seguinte exemplo: Maria apresenta os sintomas de uma doença muito rara, que afeta apenas 1 a cada 1000 indivíduos. Ela vai ao médico e realiza um exame que determina, com 99% de precisão, se ela tem ou não a doença e o resultado é positivo. Pode parecer intuitivo que, como o resultado foi positivo, Maria tem 99% de chance de ter a doença, entretanto, essa suposição está completamente incorreta. Uma maneira de se perceber isso é considerar um caso em que esse exame é realizado em 1000 pessoas aleatórias. Dessas 1000 pessoas, estatisticamente apenas uma terá a doença e essa pessoa provavelmente testará positivo. Entretanto, 10 dos testes terão um resultado falso-positivo. Então um resultado positivo colocaria uma pessoa em um grupo de 11 pessoas, onde apenas uma de fato tem a doença. Ou seja, cada uma dessas 11 pessoas teria uma chance de apenas 1/11 (9%) de estarem doentes, mesmo com um resultado positivo. O teorema de *Bayes* permite que esse cálculo seja realizado de maneira direta, para qualquer nova informação que for introduzida ao sistema, no exemplo de Maria:

Probabilidade geral de Maria ter a doença:

$$P[H] = 1/1000 \tag{2.6}$$

Probabilidade geral do teste ser positivo (é representada pela probabilidade de se ter a doença multiplicada pela probabilidade do teste ser um positivo correto somada com a probabilidade de não se ter a doença multiplicada pela probabilidade do teste ser um falso positivo):

$$P[E= \ ] \ P[H] \quad P[E|H+ \ ] \ P[\quad H] \quad P[E| \quad H] \tag{2.7}$$

$$P[E] = 1/1000 \quad 0.99 + 0.999 \quad 1/100 = 0.01 \tag{2.8}$$

Probabilidade do teste ser positivo, **sendo que** Maria tem a doença (esse tipo de probabilidade é chamada de **probabilidade condicional** e representa a probabilidade que o primeiro evento, neste caso E, ocorra, uma vez que o segundo, H, é verdadeiro):

$$P[E|H] = 0.99 \tag{2.9}$$

Probabilidade de Maria ter a doença, **sendo que** o teste foi positivo (Teorema de *Bayes*):

$$P[H|E= \ ] \ P[E|H] \quad P[H]/P[E] \tag{2.10}$$

$$P[H|E] = 0.99 \quad 0.001/0.01 = 0.099 = 9\% \tag{2.11}$$

Agora, a maneira que esse teorema pode ser aplicado a um cenário de classicação: considere a Tabela 6 como o conjunto de dados que determinam se um jogador de golfe jogou ou não em um determinado dia.

Suponha que deseja-se calcular se o jogador vai jogar no seguinte dia E, com base na Tabela 6:

Ou seja, o que se deseja calcular é $P[Sim|E]$ (probabilidade de se jogar **sendo que** os eventos de E ocorram). Primeiramente é necessário aplicar o teorema de *Bayes* a essa probabilidade:

$$P[Sim|E= \ ] \ P[E|Sim] \quad P[Sim]/P[E] \tag{2.12}$$

Agora é necessário calcular $P[E|Sim]$, $P[Sim]$ e $P[E]$. $P[E|Sim]$ representa a probabilidade dos eventos do dia E (tempo ensolarado, temperatura fria, alta humidade)

Tabela 6 Tabela de Jogo

Tempo	Temperatura	Humidade	Ventando	Jogar
Nublado	Frio	Normal	Sim	Sim
Nublado	Quente	Alta	Não	Sim
Nublado	Quente	Normal	Não	Sim
Nublado	Amena	Alta	Sim	Sim
Chuva	Frio	Normal	Não	Sim
Chuva	Amena	Alta	Não	Sim
Chuva	Amena	Normal	Não	Sim
Ensolarado	Frio	Normal	Não	Sim
Ensolarado	Amena	Normal	Sim	Sim
Chuva	Frio	Normal	Sim	Não
Chuva	Amena	Alta	Sim	Não
Ensolarado	Quente	Alta	Não	Não
Ensolarado	Quente	Alta	Sim	Não
Ensolarado	Amena	Alta	Não	Não

Tabela 7 Exemplo dia

Tempo	Temperatura	Humidade	Vento	Jogar
Ensolarado	Frio	Alta	Sim	?

ocorrerem sendo que houve jogo naquele dia. Como o teorema supõe as probabilidades destes eventos são independentes entre si, basta calcular cada uma separadamente e multiplicá-las em seguida.

Para encontrar probabilidade $P[Ensolarado|Sim]$, é necessário observar quantos dos dias em que houve jogo tinham tempo ensolarado:

Tabela 8 P[Ensolarado|Sim]

	Jogar	
	Sim	Não
Ensolarado	2	3
Nublado	4	0
Chuva	3	2

Então, $P[Ensolarado|Sim] = 2/9$. Ao replicar esse processo para os demais atributos do dia E, tem-se que: $P[Fria|Sim] = 3/9$, $P[HumidadeAlta|Sim] = 3/9$ e $P[Ventando|Sim=$] 3/9. Então:

$$P[E|Sim] = 2/9 \quad 3/9 \quad 3/9 \quad 3/9 = 0.0082 \tag{2.13}$$

Em seguida, deve-se calcular o valor de *P[Sim]*, que é simplesmente a divisão do número de dias em que houve jogo pela quantidade total de dias analisados, ou seja:

$P[Sim] = 9/14 = 0.64$.

O valor de $P[E]$ representa a probabilidade dos eventos do dia E ocorrerem juntos, que é obtida pela multiplicação das probabilidades de cada um dos eventos do dia ocorrer separadamente: $P[E=\ ]\ P[Ensolarado]\quad P[Fria]\quad P[HumidadeAlta]\quad P[Ventando]$

Estes valores são obtidos observando a base inicial e dividindo a quantidade de dias em que um evento ocorreu pela quantidade total de dias analisados, por exemplo: $P[Ensolarado]$ é a quantidade de dias ensolarados dividida pela quantidade total de dias, ou seja, 5/14. Dessa forma: $P[E] = 5/14\quad 4/14\quad 7/14\quad 6/14 = 0.022$

Então, ao substituir os valores calculados no teorema 2.12 obtemos: $P[Sim|E=\ ]$ $0.0082\quad 0.64/0.022 = 0.24$

Tem-se que, para o dia E (Tabela 7), a probabilidade de ter jogo é de 24% e a de não ter jogo é de 76%. Dessa forma, a classicação nal é:

Tabela 9 Classicação dia E

Tempo	Temperatura	Humidade	Vento	Jogar
Ensolarado	Frio	Alta	Sim	Não

2.6.3 AdaBoost

O *AdaBoost* (SCHAPIRE; FREUND, 2012) é um algoritmo de *boosting*, que faz parte dos métodos de *ensemble* para dados de lote. A tarefa de *ensemble* consiste em derivar, de um único conjunto de treino T, vários modelos de decisões $M_1, M_2, M_3, ..., M_n$ (OZA; RUSSELL, 2001). No método de *bagging*, que é uma versão simplicada do *boosting*, parte dos dados de T são selecionados e aplicados no treinamento de algum dos modelos, M_1 por exemplo. Em seguida, novos dados de T são selecionados e utilizados para o treinamento de M_2, e assim por diante. Ao nal do treinamento dos n modelos, sempre que o método receber um novo exemplo para ser rotulado ele irá enviá-lo para todos os modelos e a sua rotulação nal será a que tiver maior representatividade dentro dos resultados obtidos. A diferença entre os métodos de *bagging* e *boosting* é a maneira como os dados de treino do conjunto T são selecionados. Enquanto o *bagging* escolhe de maneira aleatória, o *boosting* atribui pesos a cada elemento de T e prioriza a escolha de elementos com maior peso. Sempre que o treino de algum dos modelos termina, ele é testado para todos os elementos de T, sendo que cada elemento classicado de maneira incorreta tem seu peso incrementado antes do treino do próximo modelo, de forma a garantir que os modelos subsequentes sejam treinados com exemplos que os modelos anteriores não conseguiram classicar bem (SCHAPIRE; FREUND, 2012).

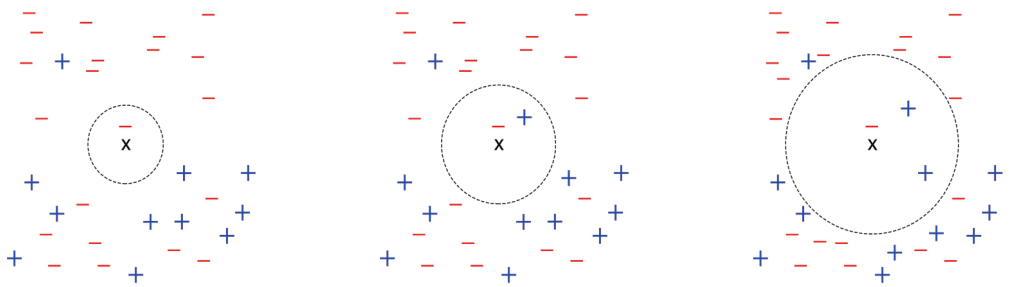
2.6.4 K vizinhos mais próximos

O algoritmo K vizinhos mais próximos (KNN) faz parte de um conjunto de classicadores conhecidos como *lazy learners*. Esses classicadores t m como principal característica a estratégia de modelar os dados depois que uma nova instância é recebida, ao contrário dos *eager learners*, como a árvore de decisão, que utilizam de um modelo gerado anteriormente à chegada da instância para realizar a classicação (TAN; STEINBACH; KUMAR, 2005). Um exemplo de *lazy leaner* que representa bem esse tipo de aprendizado é o *Rote classier* , que compara uma nova instância com toda a base de dados e a classica apenas se seus atributos forem exatamente iguais a algum elemento já classicado.

A principal fragilidade a ser destacado no *Rote Learner* é essa sua inexibilidade. Nessa direção, o KNN utiliza fundamentalmente a mesma abordagem, entretanto, ele procura os **K** elementos mais similares à instância que deve ser classicada e determina sua classe como sendo a que possui maior representatividade dentre esses elementos.

O cálculo da similaridade entre os elementos pode ser realizado de diversas maneiras dependendo do tipo dos atributos. Quando as instâncias possuem atributos numéricos, por exemplo, pode-se utilizar a **Distância Euclidiana**: $d(x, y) = d(y, x) = \sqrt{\sum_{i=1}^{n}(x_i \quad y_i)^2}$.

O aspecto mais importante para um bom funcionamento do algoritmo é a escolha do valor **K**. Se ele for pequeno demais, o classicador pode apresentar problemas de *Overtting* devido a ruído nos dados de treinamento e, se ele for grande demais, ele pode levar em consideração dados distantes demais da instância na classicação (TAN; STEINBACH; KUMAR, 2005). Nas Figuras 5 e 6, pode-se observar uma representação de como a variação do valor **K** impacta a seleção dos vizinhos pelo algoritmo.



(a) 1-Nearest Neigbor (b) 2-Nearest Neighbor (c) 3-Nearest Neighbor

Figura 5 KNN com diferentes valores de **K**

Uma característica relativa aos algoritmos *lazy* que vale a pena ressaltar é que devido a aus ncia de um modelo de decisão, eles devem calcular a proximidade de cada nova instância de teste com todos as de treino. Isso pode fazer com que o custo da classicação seja muito alto em comparação com os *eager learners*, que gastam a maior

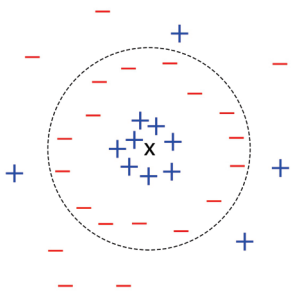


Figura 6 **KNN** com **K** muito grande

parte do tempo construindo um modelo, mas que classicam novas instâncias de forma extremamente rápida depois disso.

2.6.5 Classicador ativo

O classicador ativo é um classicador que utiliza de uma metodologia de aprendizado ativo para criar e renar seu modelo. O método de aprendizado é semi-supervisionado, ou seja, ele utiliza de dados não rotulados em seu processo de treinamento. O objetivo do algoritmo é gerar um modelo de decisão preciso utilizando o menor número possível de rótulos. Para isso, ele utiliza de um sistema de consultas (*querys*) interativo, que seleciona os dados mais incertos e solicita seus rótulos (SETTLES, 2010).

A maneira com que os dados são modelados varia em função do problema, pode-se utilizar uma abordagem probabilística, ou utilizar algum método baseado em *eager learning*, entretanto, o principal diferencial do aprendizado ativo é o sistema de solicitação de rótulos. Por isso, é necessário que exista uma estratégia precisa para realizar as consultas.

Diversas abordagens já existem para criar essas estratégias e algumas das mais usadas são:

Consulta por incerteza: uma das estratégias mais simples e mais utilizadas, o classicador ativo solicita a consulta das instâncias que ele t m menos certeza de como classicar. Escolha muito utilizada em modelos probabilísticos, já que o processo de determinar a incerteza é direto;

Consulta por comit : a estratégia consiste em manter um "comit "composto por diferentes modelos que são treinados pelos mesmo conjunto de dados. Cada modelo vota no rótulo das instâncias de teste e as que gerarem maior diverg ncia de votos são selecionadas para consulta;

Expectativa de mudança no modelo: a estratégia tenta selecionar as instâncias que causariam a maior mudança no modelo caso seus rótulos fossem conhecidos;

Expectativa de redução de erro: a estratégia tenta selecionar as instâncias que tem a maior chance de diminuir a incerteza do modelo;

Redução da Variância: estratégia tenta selecionar as instâncias que reduziriam a variância no modelo.

Para uma compreensão mais detalhada dessas estratégias recomenda-se a leitura de (SETTLES, 2010).

2.7 Classicação em Fluxos Contínuos de Dados

Quando se analisa a tarefa de classicação em Fluxos Contínuos de Dados é importante ressaltar os problemas que surgem ao se trabalhar em um ambiente dinâmico. Além do problema da mudança de conceito (*Concept Drift*), diversos outros podem surgir dependendo do tipo de aplicação em questão. Alguns dos problemas a serem tratados, de acordo com (AGGARWAL, 2007), são:

Pode ser necessário a classicação de itens com uma baixa taxa de ocorr ncia no uxo de dados. Isso pode tornar a tarefa difícil, pois classicar exemplos "raros"pode aumentar consideravelmente o custo do algoritmo.

Alguns problemas requerem que o classicador trabalhe de maneira incremental ou decremental. Quando se observavam os algoritmos incrementais, eles devem apenas considerar os exemplos que chegam, enquanto os decrementais normalmente utilizam de um protocolo de janela deslizante, o que torna algumas classes de modelos inviáveis.

Em alguns casos, é possível que o Fluxo de Dados não esteja apenas em um sistema e, nesse caso, é necessário utilizar de algoritmos distribuídos.

A restrição do tempo de processamento e da quantidade de memória que pode ser gasto para cada exemplo.

Na maioria dos casos não é possível acessar o mesmo dado mais de uma vez, que é uma prática extremamente comum e amplamente utilizada por algoritmos de classicação em lote.

2.7.1 Mudança de conceito

A maioria dos métodos de aprendizado de máquina é concebida para operar em cenários que seguem uma distribuição estacionária de dados. Entretanto, na maioria dos sistemas reais, sejam sistemas de vigilância, telecomunicação, ou redes de sensores, os

dados são dinâmicos e, por isso, estão constantemente mudando (GAMA, 2010). Diante disso, é importante que exista uma medida para levar essa constante mudança em consideração, de forma que ela possa ser integrada aos métodos de aprendizado de máquina, permitindo uma operação mais precisa em cenários reais.

A mudança de conceito é a ideia de que o conceito dos dados pode mudar com o passar do tempo, situação em que dados de treinamento antigos podem perder significância, ou até mesmo se tornarem irrelevantes. Por isso, é importante que os métodos aplicados a cenários reais sejam capazes de atualizar seus modelos, de forma a poder contar sempre com informações atualizadas.

Os algoritmos que incorporam a mudança de conceito ao seu modelo tendem a ser mais resilientes a mudanças e até mesmo a ruídos no sistema. Um exemplo desses algoritmos é a *Very Fast Decision Tree*, que é abordado na Seção 2.7.2.1.

2.7.2 Algoritmos para classicação em uxos contínuos de dados

Nesta seção serão apresentadas as versões dinâmicas dos algoritmos vistos na Seção 2.6, com nfaze nas alterações que permitem que eles sejam aplicados a uxos contínuos de dados. Uma das principais características dos algoritmos dinâmicos é a capacidade de assimilar novas informações ao modelo, de forma que ele não que desatualizado, e a capacidade de esquecer informações antigas que se tornaram irrelevantes, já que manter muitas informações desatualizadas pode vir a impactar o desempenho de um algoritmo de maneira negativa.

2.7.2.1 Very Fast Decision Tree

A VFDT *Very Fast Decision Tree* segue o modelo de uma árvore de decisão tradicional, descrito na seção 2.6.1, entretanto, a sua construção é feita substituindo folhas por nós de maneira recursiva. Para que esse processo ocorra, cada folha armazena informações estatísticas necessárias para a aplicação de uma função heurística que determina quando uma folha deve ser divida. Sempre que um registro novo é apresentado à arvore, ele percorre um caminho da raiz até alguma folha que, quando o recebe, atualiza suas informações. Depois que um número N de exemplos chegam a um folha l, um teste de avaliação de divisão é aplicado a todos os atributos possíveis. Se um dos atributos tiver um resultado consideravelmente melhor que todos os outros, a folha l é transformada em um nó baseado naquele atributo.

Entrada: S: Fluxo de exemplos;

X: Conjunto de atributos nominais;

Y: $Y = (y_1, ..., y_k)$ Conjunto de rótulos das classes;

$H(.)$: função de *split*;

N_{min}: número mínimo de exemplos;

: um menos a probabilidade desejada de escolher o atributo correto em qualquer nó;

: Constante para desempate;

Saída: HT : é uma árvore de decisão.

Para HT $FolhaVazia(raíz)$

Para cada exemplo (x, y_k) S: Percorrer HT da raiz até uma folha l;
if y_k *é desconhecido* **then**
  Classica o exemplo com a classe majoritária na folha l;
else
  Atualiza as estatísticas da folha
  if *Número de exemplos em* $l > N_{min}$ **then**
    Computar $H_j(X_i)$ para todos os atributos;
    Para X_a sendo o atributo com maior H_l;
    Para X_b sendo o atributo com segundo maior maior H_l;
    Computa $= \sqrt{\frac{R^2 ln(2/\ )}{2n}}$ (Hoeding bound);
    if $H(X_a)$ $H(X_b) >$ **then**
      Substitui l por um teste de *split* baseado no atributo X_a;
      Adiciona uma nova folha vazia para cada ramo do *split*;
    else
      if $<$ **then**
        Substitui l por um teste de *split* baseado em um atributo X_a;
        Adiciona uma nova folha vazia para cada ramo do *split*;
      end
    end
  end
end

Algorithm 1: Algoritmo VFDT por (GAMA, 2010)

Percebe-se que a comparação dos atributos é baseada no limite de *Hoeding* (*Hoeding bound*), representado no Algoritmo 1 por . Essa etapa é feita ao se comparar o resultado de uma função de avaliação H aplicada aos possíveis atributos e, caso a diferença entre os dois melhores resultados for maior do que , a folha é substituída por um nó de decisão. No caso de dois atributos que possuem avaliação muito próxima, o algoritmo não irá decidir entre eles, por isso a VFDT utiliza de um parâmetro T, que funciona como um meio de "desempate". Considerando que o valor de diminui à medida que mais exemplos são observados, ou seja, a medida que N aumenta, a comparação $< T$ é realizada e, caso seja verdadeira, faz com que a folha em questão seja dividida.

2.7.2.2 Naive Bayes

Como visto na Seção 2.6.2, o *naive bayes* utiliza de um modelo probabilístico para fazer previsões sobre os dados. Como a dimensão da estrutura probabilística do modelo

é constante, uma vez que ela está associada ao número de atributos que representam os dados, não há limitação para a quantidade de elementos que o algoritmo pode processar em um cenário dinâmico, visto que, durante tal processamento, ele precisa apenas atualizar as probabilidades de seu modelo na medida em que novos dados são processados. Por isso, as modicações necessárias para que o algoritmo *naive bayes* seja utilizado em um cenário dinâmico, ao invés de um estático, são mínimas.

2.7.2.3 OzaBoost Adwin

O *OzaBoost* (OZA; RUSSELL, 2001) é uma adaptação do *AdaBoost* (Seção 2.6.3) (SCHAPIRE; FREUND, 2012), que faz parte dos algoritmos de *ensemble* para dados de lote. O *OzaBoost* propõe mudanças que permitem que a tarefa de *boosting* possa ser aplicada a cenários dinâmicos com nível de eci ncia similar ao apresentado nos estáticos.

O principal problema em adaptar esse algoritmo para um cenário dinâmico é que os modelos novos não podem ser testados para todos os dados de treino, uma vez que em um cenário dinâmico não é possível saber o tamanho total desse conjunto, uma vez que ele é teoricamente innito. Então, para tentar simular o algoritmo estático, o *Ozaboost* tenta prever a quantidade de vezes que cada exemplo da base de treino irá ser apresentada para cada modelo. De acordo com (OZA; RUSSELL, 2001), a quantidade de vezes que cada elemento do conjunto T é utilizado no treino de cada modelo pode ser descrita como:

$$P(K = k) = \quad \quad (1/N)^k \quad (1 \quad (1/N)^{N \ \ k}) \tag{2.14}$$

Onde k representa a quantidade de vezes que cada elemento de T é utilizado para treinar os modelos e N o tamanho do conjunto T. Dessa forma, quando K , a distribuição de K tende a uma distribuição de $Poisson(1)$, então sempre que um novo exemplo de treino é recebido, ele é escolhido K $Poisson(1)$ vezes para treinar os modelos.

Em seguida, sempre que um exemplo de treino é classicado incorretamente por algum dos modelos, o parâmetro da distribuição de *Poisson* associado a aquele exemplo é incrementado antes que ele seja passado para o próximo modelo e, sempre que um exemplo é corretamente classicado, seu parâmetro é decrementado, de forma que o peso total dos dados de treino sejam divididos da seguinte maneira: metade do valor é atribuída a exemplos que foram classicados incorretamente pelo modelo anterior e a outra metade para os exemplos classicados corretamente.

Em (OZA; RUSSELL, 2001), são realizados testes que demonstram que essa versão dinâmica do algoritmo possui desempenho comparável à versão estática correspondente.

2.7.2.4 K vizinhos mais próximos

O KNN, como visto na Seção 2.6.4, é um algoritmo *lazy*, ou seja, ele necessita que se mantenha uma base de dados de treino, uma vez que ele executa a tarefa de classicação por meio da comparação de novas instâncias com os elementos dessa base. Como os elementos de um uxo contínuo de dados chegam ao longo do tempo para o sistema, manter um modelo de treino estático poderia causar impactos negativos no desempenho do algoritmo, uma vez que os dados podem estar sujeitos à mudança de conceito, por exemplo. Por isso, em cenários dinâmicos, o KNN mantém uma janela deslizante como base de dados de treino, como é demonstrado na Figura 7. Dessa forma, se o algoritmo mantiver uma janela deslizante de tamanho 1000, por exemplo, sempre que uma nova instância a ser rotulada chegar pelo uxo de dados, o algoritmo irá compará-la com os últimos 1000 elementos que foram recebidos, seguindo o padrão do KNN tradicional para classicá-la.

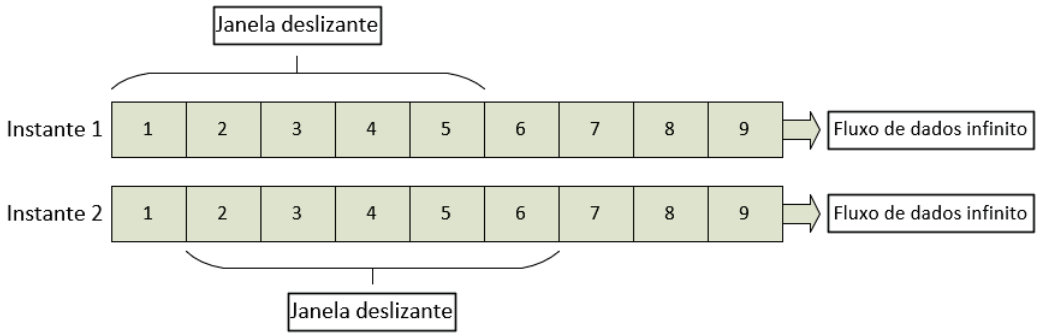


Figura 7 Janela deslizante do KNN

2.7.2.5 Classicador ativo

Embora as estratégias de solicitação de rótulos vistas na Seção 2.6.5 sejam ecientes em cenários estáticos, elas podem apresentar problemas ao serem aplicadas a cenários dinâmicos, uma vez que tendem a selecionar instâncias que estão concentradas perto da fronteira de decisão (*decision boundary*), que consiste na região do espaço do problema onde os rótulos determinados pela classicação começam a se tornar ambíguos. Esses problemas podem ocorrer devido a mudança de conceito, que pode causar com que mudanças ocorram longe dessa fronteira. Nesses cenários são utilizadas estratégias de *xed uncertainty*, alocação dinâmica da rotulação ao longo do tempo e a randomização do espaço de busca (BIFET et al., 2010).

3 Metodologia

3.1 Introdução

Conforme apresentado na seção 1.3, o principal objetivo desta monograa é realizar uma comparação de desempenho entre alguns dos principais algoritmos de classicação aplicados a diferentes bases de dados reais e articiais.

Neste capítulo, serão apresentadas as ferramentas utilizadas para a realização dos experimentos, assim como as diferentes bases de dados, metodologias e medidas de avaliação que foram selecionadas.

3.2 Algoritmos Selecionados

Como a proposta deste relatório é observar o desempenho de diferentes algoritmos em cenários dinâmicos, foram selecionados algoritmos que utilizam de diferentes estratégias classicação, com o objetivo de apresentar os aspectos positivos e negativos de diferentes tipos de aprendizagem. Nas subseções a seguir será justicada, brevemente, a escolha de cada algoritmo. Vale salientar que explicações detalhadas do funcionamento de todos os algoritmos utilizados estão presentes no Capítulo 2.

3.2.1 Naive Bayes

A estratégia Bayesiana de classicação estima a probabilidade de uma instância pertencer à alguma classe assumindo uma independ ncia condicional entre os atributos (TAN; STEINBACH; KUMAR, 2005). A motivação por trás da escolha desse algoritmo foi o fato dele utilizar de um modelo probabilístico para fazer previsões, além de ser um algoritmo muito estudado para o aprendizado de máquina.

3.2.2 K-Nearest Neighbors

O KNN faz parte dos classicadores conhecidos como *lazy learners* e foi selecionado por utilizar uma estratégia de aprendizado diferente dos demais algoritmos selecionados, uma vez que ele realiza previsões sem utilizar de um modelo de decisão pré construído.

3.2.3 Active Classier

Algoritmos de classicação, de maneira geral, tendem a atualizar seus modelos de decisão supondo que os rótulos corretos de todos os exemplos estarão disponíveis,

entretanto, para diversos cenários, essa suposição é irrealista. Por isso, optou-se por fazer o uso de um classicador ativo neste trabalho, o *active classier* .

O *active classier* é um *learner* que utiliza de uma estratégia de aprendizado semi-supervisionada, onde o algoritmo pode solicitar apenas alguns dos rótulos dos dados de treinamento. Como visto na seção 2.6.5, o *active classier* pode utilizar de diferentes estratégias tradicionais de solicitação de rótulos, mas essas estratégias podem vir a ser insucientes no tratamento de uxos contínuos de dados. Por isso foram utilizadas das estratégias disponibilizadas pelo MOA, que são: *Fixed uncertainty*, *VarUncertainty*, *RandVarUncertainty* e *SelSampling*. A seleção entre essas foi realizada de maneira aleatória, uma vez que a escolha entre elas não aparenta impactar drasticamente o resultado.

3.2.4 OzaBoost Adwin

O *OzaBoost* foi selecionado por ser uma adaptação de um algoritmo de *boosting* estático (*AdaBoost*), que utiliza da estratégia de *ensamble learning* para melhorar o desempenho de classicadores.

3.2.5 Hoeding Tree

A **Hoeding Tree** , ou **Very Fast Decision Tree**, segue a estratégia de uma **Árvore de Decisão** tradicional, entretanto, ela constrói seu modelo substituindo nós por folhas de maneira recursiva, de forma a se adaptar às diferentes mudanças que podem ocorrer em cenários dinâmicos. A motivação por trás da escolha desse algoritmo é que o aprendizado de árvores de decisão em Fluxos Contínuos de Dados é um dos problemas mais desaadores na mineração de dados (GAMA, 2010) e a VFDT é um dos algoritmos que consegue realizar essa tarefa de maneira eciente e precisa.

3.3 Massive Online Analysis Framework

A ferramenta escolhida para a realização dos experimentos foi o MOA (*Massive Online Analysis*), uma vez que ele é o *framework* aberto mais popular para realização de tarefas de classicação, regressão, clusterização e simulação da mudança de conceito (BIFET et al., 2010).

Esta monograa irá trabalhar com a tarefa de **Classicação** e com a simulação da mudança de conceito, além da geração de uxos de dados articias fornecidas pela ferramenta.

Embora o MOA possa ser executado por linha de comando, ele também oferece uma interface gráca que permite acesso a todas as ferramentas para a tarefa de classicação (Figuras 8 e 9). As principais funcionalidades que foram utilizadas são:

Selecionar o método de avaliação;

Selecionar um uxo de dados, real ou articial;

Selecionar um algoritmo;

Congurar o número de instâncias analisadas;

Observar os resultados;

Geração dos grácos com base nos resultados;

Geração de um arquivo .txt com os resultados.

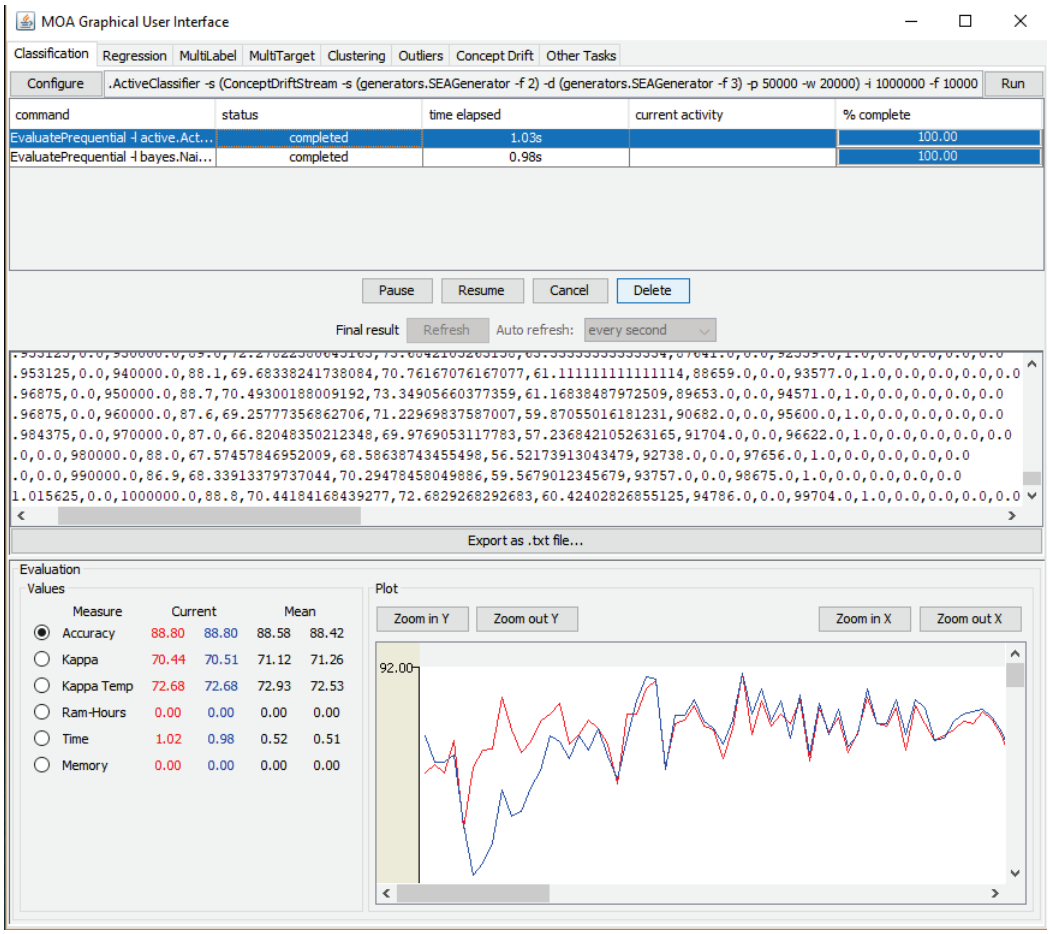


Figura 8 **MOA**: Interface do Classicador

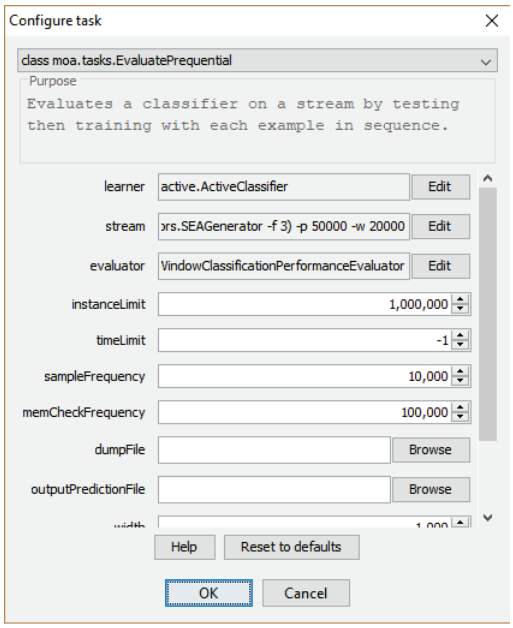


Figura 9 **MOA**: Conguração

3.3.1 Processamento dos dados

Os requisitos para um modelo de análise de um uxo contínuo de dados de acordo com (BIFET et al., 2010) são:

Processar um exemplo por vez;

Analisar cada exemplo apenas uma vez;

Usar uma quantidade limitada de memória

Estar pronto para realizar uma previsão a qualquer momento.

Diante desses requisitos, a maneira com que o processamento de dados ocorre pelo MOA é:

1. O algoritmo recebe um exemplo do uxo de dados;
2. O algoritmo processa o exemplo e atualiza seu modelo da maneira mais rápida possível e respeitando a quantidade delimitada de memória;
3. O algoritmo já pode aceitar outro exemplo e está pronto para rotular exemplos desconhecidos.

3.4 Bases de dados

Nesta seção serão apresentadas as bases selecionadas para a execução dos experimentos, tanto reais quanto articiais, suas características gerais e a maneira com que elas foram conguradas. O motivo da escolha das bases articiais, em adição as reais, foi que as bases articiais permitem o controle de como as instâncias são geradas e, por isso, tornam mais fácil a simulação de diferentes cenários de teste.

3.4.1 Bases Reais

O MOA permite que os dados sejam lidos de um arquivo *ARFF*. Dessa forma, para a execução dos experimentos foram selecionadas as seguintes base reais:

Forest CoverType: representa o tipo de árvores de um espaço de 30x30 metros obtido do Serviço Floresta Americano (USFS). Contém 7 tipos diferentes de árvores, 581.012 instâncias e 54 atributos (SCIENCES COLLEGE OF NATURAL RESOURCES,).

Electricity: são dados coletados do *Australian New South Wales Electricity Market*, onde o preço da eletricidade é atualizado de 5 em 5 minutos. Contém 45.312, 8 atributos e dene se o preço da eletricidade subiu ou desceu nas últimas 24 horas (GAMA, 2010).

KDD CUP 99: Conjunto de dados utilizado para a terceira competição *International Knowledge Discovery and Data Mining Tools*. A tarefa da competição era construir uma rede de detecção de intrusos, um modelo capaz de prever e distinguir conexões normais de intrusos ou ataques. Contém 42 atributos, 23 classes e 4.898.582 instâncias (INFORMATION; SCIENCE UNIVERSITY OF CALIFORNIA,).

3.4.2 Bases Articiais

O MOA permite a geração de uxos de dados aleatórios com base em diferentes estratégias. As que foram utilizadas para este relatório são:

RandomTreeGenerator: gera um uxo de dados baseado em uma árvore aleatória. Foi selecionado por permitir a seleção do número de atributos a serem utilizados, tanto numéricos quanto nominais, assim como o número total de classes. Todos os uxos de dados foram gerados a partir da mesma semente para garantir que os testes fossem realizados nas mesmas circunstâncias.

SeaGenerator: gera um uxo de dados baseado em uma de quatro funções disponíveis. Foi selecionado por permitir a introdução de ruído ao ambiente, o que permite simular melhor um cenário real.

ConceptDriftRealStream: simula a mudança de conceito. A simulação é feita selecionando dois geradores de uxos de dados, no caso deste trabalho foi utilizado o *SeaGenerator* com diferentes funções geradoras e diferentes níveis de ruído, e selecionando um ponto do uxo a partir do qual as instâncias selecionadas passam a serem retiradas do segundo gerador, ao invés do primeiro.

3.5 Estratégias de avaliação

Das diversas estratégias de teste e treino disponíveis pelo MOA, duas foram selecionadas para a realização desta monograa, de forma que todos os diferentes cenários de testes foram replicados para cada uma delas. Essas estratégias são:

Evaluate Prequential: cada exemplo é utilizado primeiramente para teste e, em seguida, para treino. Dessa maneira, o modelo está constantemente sendo treinado com exemplos que ainda não foram apresentados a ele.

Interleaved Chunks: avalia o classicador em um uxo de dados testando e depois o treinando com blocos seguidos de dados. É similar ao *Evaluate Prequential* só que utiliza de blocos de exemplos ao invés de exemplos individuais.

3.6 Medidas de avaliação

As medidas de avaliação oferecidas pelo MOA são precisão, *kappa*, tempo e memória. Para os experimentos realizados nesta monograa foram utilizados a precisão, *kappa* e o tempo. A precisão indica a porcentagem de acerto entra as previsões das classes pelo algoritmo e os rótulos reais das instâncias, enquanto o *kappa* indica o nível de concordância entre dois avaliadores para um mesmo conjunto de dados, levando em consideração a probabilidade de concordância por sorte.

3.7 Teste estatístico

Como esta monograa propõe um estudo em torno da comparação de diferentes algoritmos é necessário que haja algum método para determinar se os resultados encontrados possuem alguma signicância estatística. Para isso, foi selecionado o *Friedman test.*

3.7.1 Friedman test

O *Friedman test* é um teste estatístico não-paramétrico que tem como objetivo montar um *rank* de diferentes algoritmos para um determinado conjunto de dados. Para

isso, ele parte da hipótese de que todos os algoritmos terão desempenho e, consequentemente, *ranks* equivalentes. Em seguida, ele calcula o *rank* médio de todos os algoritmos e, baseado nos valores críticos, aceita ou rejeita a hipótese inicial. Dessa forma, é possível, com base nesses valores, determinar os classicadores que possuem um desempenho signicativamente superior aos demais (GARCIA et al., 2010).

Para a execução dos testes foi utilizada uma implementação em *Java* do *Friedman Test* aplicada aos resultados obtidos pelos experimentos realizados no MOA.

3.8 Considerações nais

Neste capítulo foram apresentadas as ferramentas e métodos que foram utilizados na realizações dos experimentos realizados neste trabalho. No Capítulo 4, os experimentos e seus resultados serão apresentados e analisados e, por m, será aplicado o teste estatístico.

4 Experimentos e resultados

4.1 Introdução

Neste capítulo serão apresentados os experimentos realizados nesta monograa, assim como os resultados obtidos. Todos os experimentos foram realizados no *framework* MOA (BIFET et al., 2010) e a motivação por trás deles é observar o desempenho de alguns algoritmos, que utilizam de diferentes estratégias de classicação, e determinar as vantagens e desvantagens do uso deles para a mineração de uxos contínuos de dados.

Bases reais e articiais foram utilizadas nos experimentos, sendo que as articiais foram geradas pelos geradores de dados do *framework* MOA.

Para o teste e treino foram utilizadas duas estratégias de avaliação, *evaluate prequential* e *interleaved chunks* e as medidas de avaliação precisão e *kappa*, sendo que todos estão implementados no *framework* MOA.

Todos os parâmetros necessários para a conguração dos experimentos foram mantidos no padrão *default* do MOA, com exceção dos que estão indicados nas tabelas de resultados.

4.2 Bases de Dados

Todos os experimentos foram realizados nas mesmas bases de dados que possuem quantidade de atributos, classes, ruído e mudança de conceito variada.

Tabela 10 Características das bases reais

	Atributos	Classes	Instâncias	Concept Drift G	Concept Drift A	Ruido	Base/Ger
Base 1	8	2	45311	não	não	não	Elecnorm
Base 2	54	7	581055	não	não	não	Covertype
Base 3	42	23	4898582	não	não	não	kddcup 99

As Tabelas 10 e 11 representam as características das bases reais e articiais utilizadas nos experimentos. Nessas tabelas, as colunas representam, respectivamente:

1. Quantidade de atributos;
2. Quantidade de classes;
3. Número de Instâncias;

Tabela 11 Características das bases articiais

	Atributos	Classes	Instâncias	Concept Drift G	Concept Drift A	Ruido	Base/Ger
Base 4	5	2	100000	não	não	não	RandTree
Base 5	10	5	100000	não	não	não	RandTree
Base 6	15	10	100000	não	não	não	RandTree
Base 7	30	20	100000	não	não	não	RandTree
Base 8	-	-	100000	não	não	não	SeaGen
Base 9	-	-	100000	não	não	Sim	SeaGen
Base 10	-	-	100000	Sim	não	não	SeaGen
Base 11	-	-	100000	não	Sim	não	SeaGen
Base 12	-	-	100000	não	Sim	Sim	SeaGen
Base 13	-	-	100000	Sim	não	Sim	SeaGen
Base 14	5	10	100000	não	não	não	RandTree
Base 15	5	30	100000	não	não	não	RandTree
Base 16	15	2	100000	não	não	não	RandTree
Base 17	15	30	100000	não	não	não	RandTree
Base 18	50	2	100000	não	não	não	RandTree
Base 19	50	10	100000	não	não	não	RandTree
Base 20	50	30	100000	não	não	não	RandTree

4. Se foi, ou não, simulada a mudança de conceito gradual;

5. Se foi, ou não, simulada a mudança de conceito abrupta;

6. Se a base possui, ou não, ruído (para todos as bases que possuíam ruído, a quantidade utilizada foi de 10%);

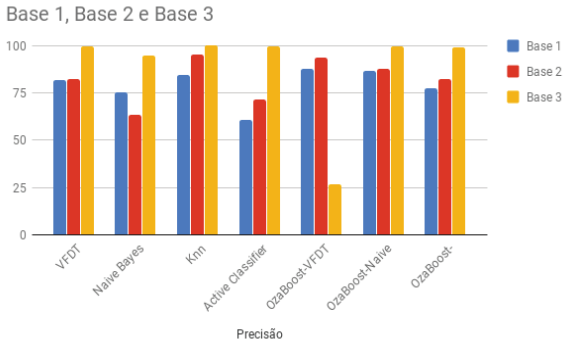
7. A base, ou gerador de dados, que foi utilizado.

4.3 Experimentos com bases reais

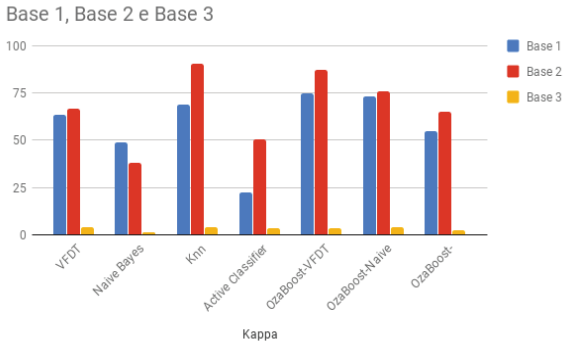
Nesta seção serão apresentados os resultados dos testes realizados em bases reais. Consta salientar que nos experimentos desta monograa não foi utilizado o algoritmo *OzaBoost* aplicado ao KNN, uma vez que o tempo de execução desses testes, mesmo para as menores bases de dados, era extremamente elevado.

4.3.1 Evaluate Prequential

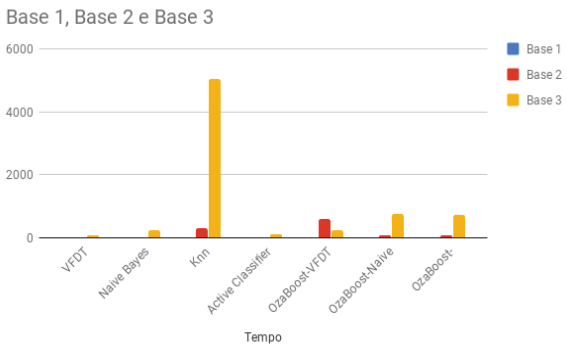
As Tabelas 12, 13 e 14 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases reais utilizadas. A Figura 10 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados reais, utilizando o método *evaluate prequential*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados reais, utilizando o método *evaluate prequential*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados reais, utilizando o método *evaluate prequential*

Figura 10 Grácos dos resultados do *Evaluate Prequential* para bases reais.

Tabela 12 Tabela com os resultados da medida de avaliação **precisão** nas bases reais, usando o método *evaluate prequential*. Resultados em %

Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 1	81,6	75,3	84,5	60,6	87,6	86,6	77,6
Base 2	82,25	63,3	95,4	71,75	93,52	87,9	82,5
Base 3	99,94	95,06	99,97	99,53	26,47	99,83	99

Tabela 13 Tabela com os resultados da medida de avaliação **kappa** nas bases reais, usando o método *evaluate prequential*. Resultados em %

Kappa	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 1	63,19	48,86	68,72	22,5	75,03	73,17	54,62
Base 2	66,81	38,29	90,68	50,26	87,23	75,9	65,22
Base 3	3,72	1,27	3,79	3,32	3,48	3,79	2,46

Tabela 14 Tabela com os resultados da medida de avaliação tempo nas bases reais, usando o método *evaluate prequential*. Resultados em segundos

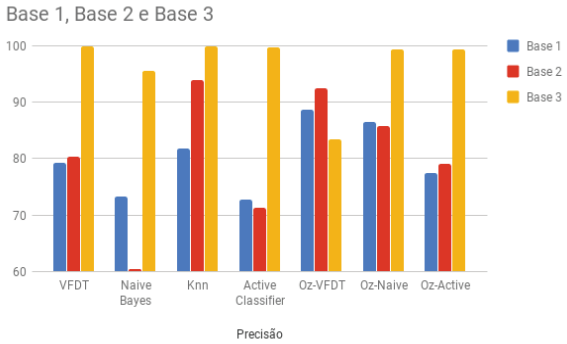
Tempo	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 1	0,22	0,14	5,73	0,12	9,03	1,22	1,3
Base 2	10,59	9,72	297,95	9,48	606,41	66,72	62,95
Base 3	67	238,19	5047,83	95	238,81	743,36	725,73

4.3.2 Interleaved Chunks

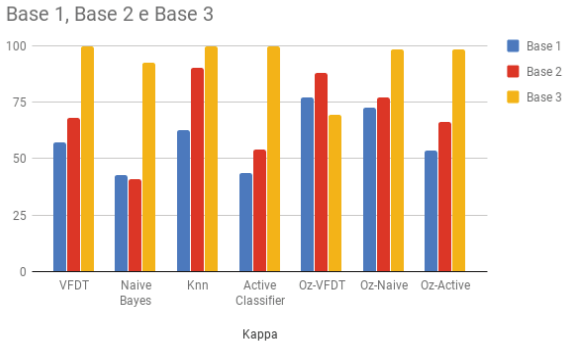
As Tabelas 15, 16 e 17 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases reais utilizadas. A Figura 11 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 15 Tabela com os resultados da medida de avaliação **precisão** nas bases reais, usando o método *interleaved chunks*. Resultados em %

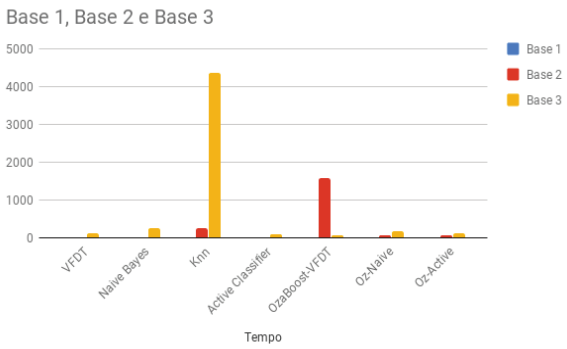
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 1	79,2	73,36	81,87	72,7	88,76	86,59	77,54
Base 2	80,31	60,52	93,99	71,23	92,48	85,78	79,04
Base 3	99,9	95,6	99,98	99,78	83,35	99,39	99,42



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados reais, utilizando o método *interleaved chunks*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados reais, utilizando o método *interleaved chunks*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados reais, utilizando o método *interleaved chunks*

Figura 11 Grácos dos resultados do *Interleaved Chunks* para bases reais.

Tabela 16 Tabela com os resultados da medida de avaliação *kappa* nas bases reais, usando o método *interleaved chunks*. Resultados em %

Kappa	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 1	57,23	42,54	62,77	43,83	76,9	72,57	53,54
Base 2	68,25	40,91	90,33	53,98	87,88	77,09	66,17
Base 3	99,83	92,59	99,96	99,62	69,56	98,17	98,31

Tabela 17 Tabela com os resultados da medida de avaliação tempo nas bases reais, usando o método *interleaved chunks*. Resultados em %

Tempo	VFDT	Naive Bayes	Knn	Active Classier	OzaBoost-VFDT	Oz-Naive	Oz-Active
Base 1	0,22	0,12	5,47	0,14	9,23	1,23	1,34
Base 2	12,41	9,66	241,88	9,42	1588,29	65,48	62,64
Base 3	124	240	4366	96	52,33	177	124

4.3.3 Discussão dos resultados

De maneira geral, a utilização da estratégia de treino e teste *evaluate prequential* ou *interleaved chunks* não apresentou diferenças muito signicativas nos resultados dos experimentos realizados em bases reais. A única exceção foi o tempo, que aumentou consideravelmente com o uso da estratégia *evaluate prequential*, uma vez que ela testa e treina o modelo para cada uma das instâncias da base, enquanto o *interleaved chunks* separa pedaços diferentes da base para teste e treino.

Percebe-se os quatro primeiro algoritmos mantiveram um padrão com relação a precisão, de forma que o Knn teve melhor resultado para todas as bases, seguido da VFDT e, o *active classier* e o *naive bayes* alternavam em terceiro e quarto lugar. O uso do *OzaBoost* causou uma melhora no desempenho dos algoritmos na maioria dos casos, permitindo que os resultados da VFDT e do *naive bayes* ultrapassassem os do Knn em certas situações. Além disso, a precisão do *active classier* foi extremamente interessante para a **base 3**, uma vez que ele, mesmo sendo um algoritmo que não tem acesso aos rótulos de todos os exemplos, conseguiu manter um resultado muito bom.

No caso do *Kappa*, o padrão foi similar ao da precisão para a **Base 1**, entretanto, para a **base 2**, o Knn apresentou resultados muito superiores aos demais, superando-os mesmo com o uso do *OzaBoost*. Já para a **base 3**, que é a base com a maior quantidade de instâncias, todos os algoritmos tiveram resultados muito ruins.

Com relação ao tempo de execução dos algoritmos para as bases reais, o *naive bayes* e o *active classier* são mais rápidos para bases menores, como a **base 1** e a **base 2**, entretanto, para bases de tamanho maior, como a **base 3**, a VFDT é consideravelmente mais rápida. Já o Knn apresentou um desempenho signicativamente pior que todos os demais, obtendo um tempo melhor apenas que o *Ozaboost* aplicado à VFDT na **base 2**.

Isso ocorre porque o KNN tem que encontrar os k vizinhos mais proxímos, considerando uma janela de dados, para cada exemplo de teste. Percebe-se também que a utilização do *OzaBoost* causa um grande aumento no tempo de execução de todos os algoritmos, uma vez que, de forma geral, os algoritmos que usam *ensemble* gastam mais tempo, já que a classicação é baseada em um comit de classicadores e todos esses classicadores devem ser treinados separadamente.

4.4 Experimentos com bases articiais

Nesta seção serão apresentados os resultados dos testes realizados em bases de dados articiais. Os testes serão divididos pela quantidade de atributos, assim como pela presença, ou não, de ruído e mudança de conceito nas bases.

4.4.1 Experimentos com bases articiais com poucos atributos

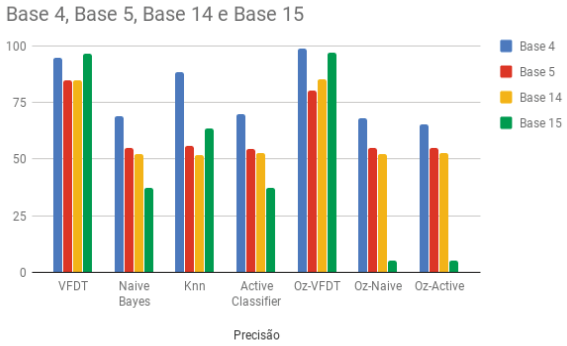
Para as bases com poucos atributos, foram selecionadas as bases 4, 5, 14 e 15, todas descritas detalhadamente na Tabela 11. Consta ressaltar que a quantidade de classes nessas bases varia entre 2, 5, 10 e 30. Essas bases não possuem mudança de conceito e ruído. Nesses experimentos tem-se como objetivo vericar se o desempenho dos classicadores é afetado pelo aumento do número de classes.

4.4.1.1 Evaluate Prequential

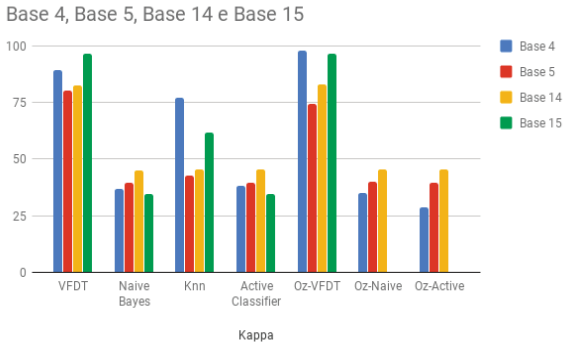
As Tabelas 18, 19 e 20 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases articiais, com poucos atributos, utilizadas. A Figura 12 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 18 Tabela com os resultados da medida de avaliação precisão nas bases articiais com poucos atributos, usando o método *evaluate prequential*. Resultados em %

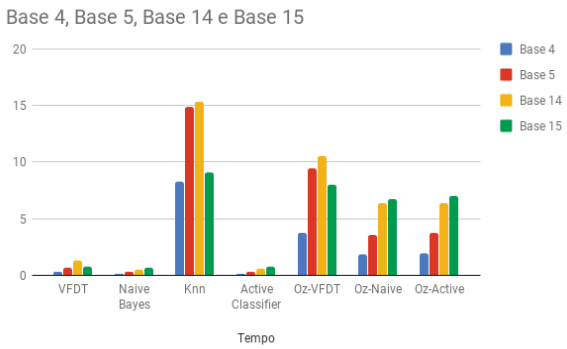
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 4	94,8	68,8	88,6	69,7	98,7	68,1	65,3
Base 5	84,7	55	55,9	54,6	80,4	55,1	54,8
Base 14	84,8	52,2	52	52,5	85,1	52,3	52,7
Base 15	96,5	37,3	63,5	37,4	96,8	5,4	5,4



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados articiais com poucos atributos, utilizando o método *evaluate prequential*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados articiais com poucos atributos, utilizando o método *evaluate prequential*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados articiais com poucos atributos, utilizando o método *evaluate prequential*

Figura 12 Grácos dos resultados do *evaluate prequential* para bases articiais com poucos atributos.

Tabela 19 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com poucos atributos, usando o método *evaluate prequential*. Resultados em %

Kappa	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 4	89,52	36,68	77,02	38,28	97,98	35,07	28,8
Base 5	80,09	39,77	42,85	39,47	74,57	39,88	39,63
Base 14	82,62	45,16	45,23	45,54	82,98	45,23	45,58
Base 15	96,35	34,52	61,92	34,59	96,66	0	0

Tabela 20 Tabela com os resultados da medida de avaliação tempo nas bases articiais com poucos atributos, usando o método *evaluate prequential*. Resultados em segundos

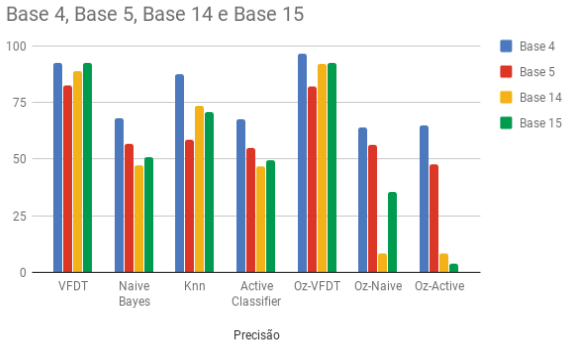
Tempo	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 4	0,33	0,14	8,31	0,14	3,75	1,84	1,91
Base 5	0,67	0,31	14,88	0,33	9,48	3,53	3,77
Base 14	1,31	0,53	15,3	0,55	10,53	6,36	6,36
Base 15	0,8	0,67	9,05	0,75	7,97	6,72	6,98

4.4.1.2 Interleaved Chunks

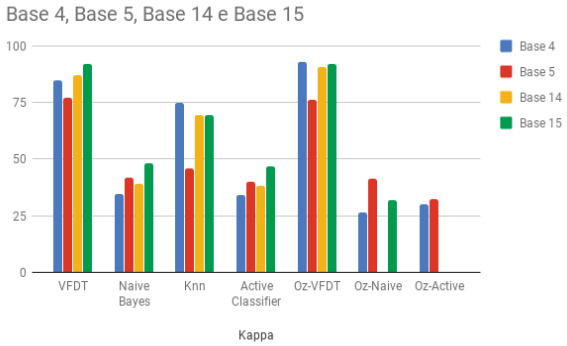
As Tabelas 18, 19 e 20 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases articiais, com poucos atributos, utilizadas. A Figura 12 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 21 Tabela com os resultados da medida de avaliação precisão nas bases articiais com poucos atributos, usando o método *interleaved chunks*. Resultados em %

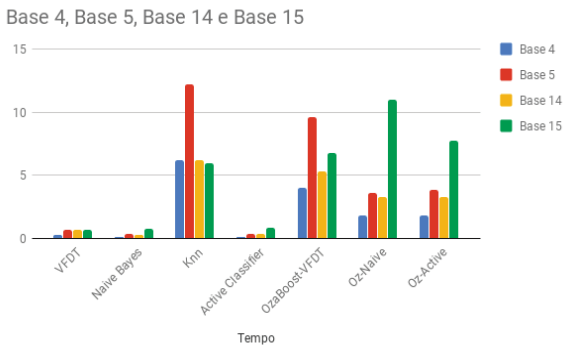
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 4	92,49	67,96	87,71	67,72	96,5	63,87	64,69
Base 5	82,49	56,55	58,45	55,15	81,92	56,46	47,91
Base 14	88,91	47,16	73,46	46,71	91,95	8,32	8,32
Base 15	92,35	50,92	70,84	49,54	92,57	35,61	3,85



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados articiais com poucos atributos, utilizando o método *interleaved chunks*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados articiais com poucos atributos, utilizando o método *interleaved chunks*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados articiais com poucos atributos, utilizando o método *interleaved chunks*

Figura 13 Grácos dos resultados do *Interleaved Chunks* para bases articiais com poucos atributos.

Tabela 22 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com poucos atributos, usando o método *interleaved chunks*. Resultados em %

Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 4	84,76	34,58	75,03	33,93	92,91	26,48	29,96
Base 5	77,08	41,63	45,75	40,06	76,36	41,47	32,42
Base 14	87,29	38,92	69,58	38,4	90,79	0	0
Base 15	91,92	48,14	69,2	46,62	92,16	31,8	0

Tabela 23 Tabela com os resultados da medida de avaliação tempo nas bases articiais com poucos atributos, usando o método *interleaved chunks*. Resultados em segundos

Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 4	0,3	0,14	6,19	0,12	3,97	1,83	1,81
Base 5	0,7	0,33	12,22	0,33	9,61	3,56	3,8
Base 14	0,67	0,3	6,23	0,33	5,3	3,27	3,27
Base 15	0,69	0,77	5,95	0,81	6,77	10,95	7,72

4.4.1.3 Discussão dos resultados

Para as bases geradas articialmente com poucos atributos, percebe-se que o algoritmo que obteve melhores resultados, tanto na precisão, quanto no *kappa*, foi a VFDT. A árvore obteve valores superiores para todas as bases, tendo seu resultado melhorado nas bases 5, 14 e 15 com o uso do *OzaBoost*. Assim, é possível perceber que a VFDT não aparenta ser muito sensível à mudança no número de classes, entretanto, o *naive bayes* e o *active classier* tiveram uma perda notável de desempenho a medida que o número de classes aumentou, sugerindo que eles são signicantemente afetados pela mudança.

Também foi possível observar que a quantidade de classes tem uma relação com o tempo de execução, uma vez que a todos os algoritmos foram mais rápidos para **base 4**, que possui o menor número de classes. Entretanto, os algoritmos também foram mais lentos para a **base 14**, que possui 10 classes, do que para a **base 15**, que possui 30 classes. Dessa forma, a relação aparenta ser mais complicada do que simplesmente: quanto mais classes, mais tempo necessário para a execução dos algoritmos. O mesmo comportamento ocorreu com relação aos resultados de precisão e *kappa* do KNN, uma vez seus resultados na **base 14** foram melhores, para estes parâmetros, do que na **base 15**. Dessa forma, seriam necessários testes mais detalhados para deduzir algo a mais sobre essa relação.

4.4.2 Experimentos com bases articiais com muitos atributos

Para as bases com muitos atributos foram selecionadas as bases 6, 7, 16, 17, 18, 19, 20, todas descritas detalhadamente na Tabela 11. A quantidade de de classes nessas

bases continua variando entre 2, 5, 10 e 30. Essas bases não possuem mudança de conceito e ruído. Nesses experimentos tem-se como objetivo vericar se o desempenho dos classicadores é afetado pelo aumento do número de atributos, que varia entre 15, 30 e 50.

4.4.2.1 Evaluate Prequential

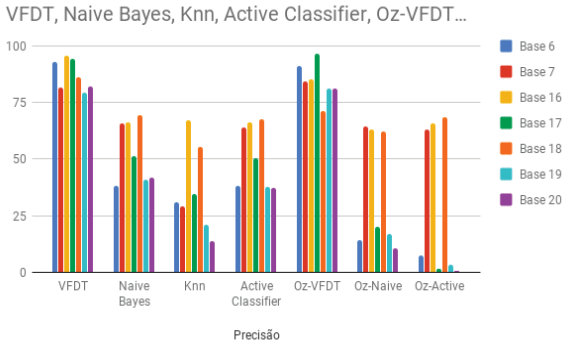
As Tabelas 24, 25 e 26 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases articiais, com muitos atributos, utilizadas. A Figura 14 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 24 Tabela com os resultados da medida de avaliação precisão nas bases articiais com muitos atributos, usando o método *evaluate prequential*. Resultados em %

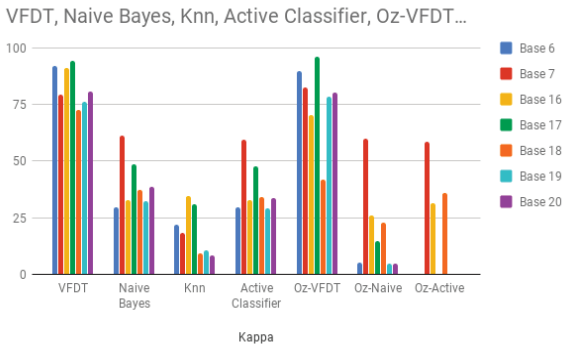
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 6	92,8	38,1	30,8	38	91,1	14,4	7,5
Base 7	81,7	65,7	29,1	64,2	84,3	64,4	63,3
Base 16	95,6	66,4	67,3	66,4	85,2	62,9	65,7
Base 17	94,4	51,2	34,5	50,5	96,4	19,9	1,4
Base 18	86,3	69,4	55,3	67,8	71,2	62,1	68,7
Base 19	79,2	40,8	21	37,9	81	17	3,3
Base 20	81,9	41,9	13,7	37,1	81,2	10,7	0,6

Tabela 25 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com muitos atributos, usando o método *evaluate prequential*. Resultados em %

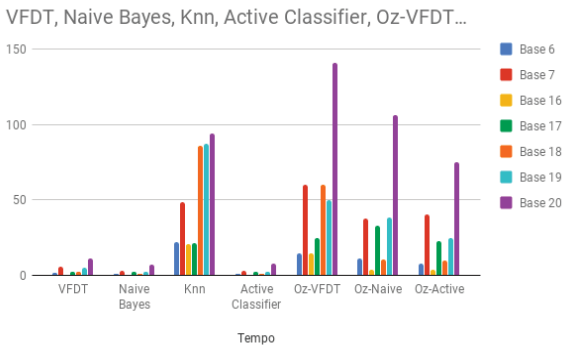
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 6	91,89	29,81	21,82	29,78	89,96	5,14	0
Base 7	79,45	61,25	18,44	59,51	82,37	59,79	58,44
Base 16	91,19	32,75	34,62	32,71	70,38	25,84	31,27
Base 17	94,1	48,41	30,94	47,72	96,21	14,5	0
Base 18	72,37	37,48	9,13	34,27	41,9	22,65	36,14
Base 19	76,41	32,33	10,57	29,05	78,43	4,57	0
Base 20	80,9	38,54	8,52	33,51	80,15	4,61	0



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados articiais com muitos atributos, utilizando o método *evaluate prequential*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados articiais com muitos atributos, utilizando o método *evaluate prequential*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados articiais com muitos atributos, utilizando o método *evaluate prequential*

Figura 14 Grácos dos resultados do *Evaluate Prequential* para bases articiais com muitos atributos.

Tabela 26 Tabela com os resultados da medida de avaliação tempo nas bases articiais com muitos atributos, usando o método *evaluate prequential*. Resultados em %

Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 6	1,36	0,8	21,94	0,86	14,39	10,95	8,08
Base 7	5,7	2,91	48,75	3,12	60	37,73	40,02
Base 16	0,64	0,25	20,67	0,27	14,67	3,75	3,75
Base 17	2,16	2,16	21,16	2,38	24,78	33,16	22,41
Base 18	2,19	0,77	86	0,78	59,94	10,47	10,05
Base 19	5,33	2,44	87	2,67	49,55	38,44	24,42
Base 20	11,3	6,97	94	7,66	141	106	75

4.4.2.2 Interleaved Chunks

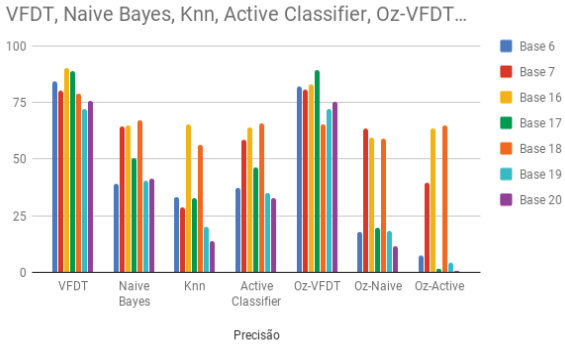
As Tabelas 27, 28 e 29 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases articiais, com muitos atributos, utilizadas. A Figura 15 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 27 Tabela com os resultados da medida de avaliação precisão nas bases articiais com muitos atributos, usando o método *interleaved chunks*. Resultados em %

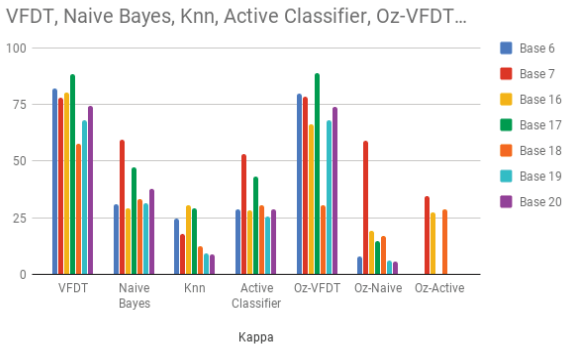
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 6	84,24	39,19	33,15	37,19	82,02	17,91	7,23
Base 7	80,4	64,24	28,71	58,36	80,71	63,7	39,61
Base 16	90,13	64,69	65,4	64,08	83,19	59,61	63,56
Base 17	88,84	50,27	32,77	46,14	89,29	19,68	1,36
Base 18	78,93	66,94	56,46	65,6	65,45	58,94	64,78
Base 19	72,1	40,4	20,02	35,23	72,06	18,44	4,1
Base 20	75,63	41,27	13,66	32,64	75,2	11,34	0,74

Tabela 28 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com muitos atributos, usando o método *interleaved chunks*. Resultados em %

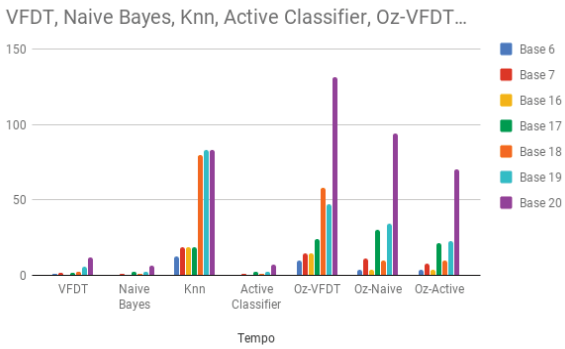
Kappa	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 6	82,21	30,85	24,44	28,78	79,73	7,8	0
Base 7	78	59,62	17,89	52,98	78,32	59,01	34,48
Base 16	80,25	29,28	30,7	28,11	66,39	19,07	27,17
Base 17	88,25	47,46	29,22	43,13	88,74	14,48	0
Base 18	57,71	33,3	12,37	30,61	30,58	17,12	28,71
Base 19	68,26	31,48	9,31	25,71	68,28	6,29	0
Base 20	74,32	37,9	8,71	28,78	73,84	5,58	0



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados articiais com muitos atributos, utilizando o método *interleaved chunks*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados articiais com muitos atributos, utilizando o método *interleaved chunks*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados articiais com muitos atributos, utilizando o método *interleaved chunks*

Figura 15 Grácos dos resultados do *Interleaved Chunks* para bases articiais com muitos atributos.

Tabela 29 Tabela com os resultados da medida de avaliação tempo nas bases articiais com muitos atributos, usando o método *interleaved chunks*. Resultados em segundos

Tempo	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 6	0,7	0,33	12,22	0,33	9,61	3,56	3,8
Base 7	1,47	0,75	18,91	0,83	14,27	11,08	7,88
Base 16	0,64	0,27	18,67	0,23	14,31	3,69	3,7
Base 17	1,94	2,03	18,45	2,25	24,2	30,12	21,25
Base 18	2,38	0,8	80	0,75	58,12	10,03	9,86
Base 19	5,41	2,3	83	2,56	47,3	34,27	22,8
Base 20	11,84	6,38	83	7,03	131	94	70

4.4.2.3 Discussão dos resultados

O objetivo desta análise das bases articiais com muitos atributos é observar como o desempenho dos algoritmos varia em função da quantidade de atributos. Dessa forma, foi possível perceber que ambos a VFDT e o KNN sofreram uma clara perda de desempenho, com relação aos parâmetros *kappa* e precisão, a medida em que o número de atributos aumentou de 15 para 30. O mais interessante é que o *naive bayes* obteve uma melhora em em relação a esses parâmetros com o aumento do número de atributos. Isso pode ser devido ao fato de que ele utiliza de um modelo probabilístico baseados em atributos para realizar previsões. Dessa forma, quanto mais atributos os exemplos que o algoritmo receber possuírem, mais informações são disponibilizadas para o modelo, que se torna mais preciso como consequ ncia.

Percebe-se também que o número de atributos está diretamente relacionado ao tempo, uma vez que, a medida que ele subia, o tempo de execução também aumentava para todos os algoritmos testados. Isso ocorre porque todos eles fazem previsões por meio de cálculos associados ao número de atributos, por isso, é intuitivo esperar que a medida que esse número aumente, o custo de execução dos algoritmos também suba.

4.4.3 Experimentos com bases articiais com ruído e mudança de conceito

Nesta seção serão apresentados os resultados dos experimentos realizados nas bases 8, 9, 10, 11, 12 e 13, descritas na tabela 11, que representam conjuntos de dados sujeitos a mudança de conceito e ruído. Não foi possível encontrar o número de atributos e classes utilizado pelo *SeaGenerator*, dessa forma, foi considerado apenas que eles são valores constantes. As bases de dados utilizadas nesta seção são as que mais se aproximam de um cenário real, uma vez que elas sofrem de mudanças de conceito abruptas, ou graduais, e apresentam uma taxa de ruído entre seus dados.

4.4.3.1 Evaluate Prequential

As Tabelas 30, 31 e 32 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases articiais, com mudança de conceito e ruído, utilizadas. A Figura 16 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 30 Tabela com os resultados da medida de avaliação precisão nas bases articiais com mudança de conceito e/ou ruído, usando o método *evaluate prequential*. Resultados em %

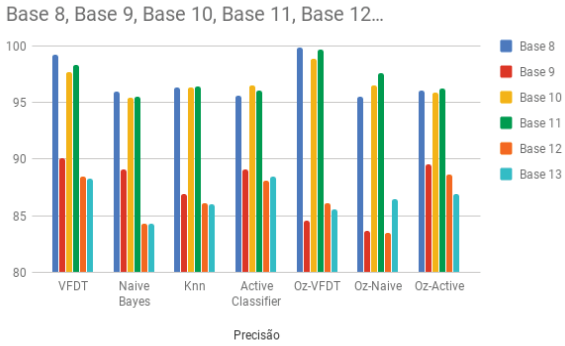
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 8	99,2	96	96,3	95,6	99,9	95,5	96,1
Base 9	90,1	89,1	86,9	89,1	84,6	83,7	89,5
Base 10	97,7	95,4	96,3	96,5	98,9	96,5	95,9
Base 11	98,3	95,5	96,4	96,1	99,7	97,6	96,2
Base 12	88,5	84,3	86,1	88,1	86,1	83,5	88,6
Base 13	88,3	84,3	86	88,5	85,6	86,5	86,9

Tabela 31 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com mudança de conceito e/ou ruído, usando o método *evaluate prequential*. Resultados em %

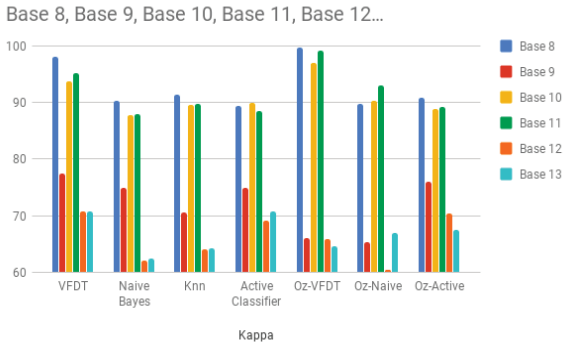
Kappa	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 8	98,14	90,38	91,42	89,38	99,77	89,74	90,81
Base 9	77,51	74,91	70,54	74,87	66,07	65,38	75,94
Base 10	93,74	87,79	89,51	89,92	96,95	90,25	88,94
Base 11	95,2	87,9	89,68	88,48	99,15	93,03	89,3
Base 12	70,79	62,15	64,11	69,14	65,89	60,53	70,47
Base 13	70,74	62,52	64,25	70,73	64,59	66,96	67,44

Tabela 32 Tabela com os resultados da medida de avaliação tempo nas bases articiais com mudança de conceito e/ou ruído, usando o método *evaluate prequential*. Resultados em segundos

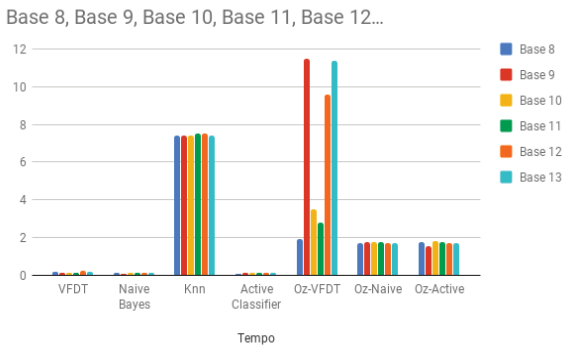
Tempo	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 8	0,2	0,11	7,41	0,09	1,94	1,73	1,78
Base 9	0,16	0,09	7,41	0,11	11,48	1,75	1,53
Base 10	0,16	0,11	7,41	0,12	3,48	1,75	1,8
Base 11	0,16	0,12	7,52	0,12	2,8	1,77	1,77
Base 12	0,22	0,12	7,52	0,12	9,58	1,7	1,72
Base 13	0,19	0,12	7,42	0,11	11,39	1,73	1,72



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados articiais com mudança de conceito e/ou ruído, utilizando o método *evaluate prequential*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados articiais com mudança de conceito e/ou ruído, utilizando o método *evaluate prequential*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados articiais com mudança de conceito e/ou ruído, utilizando o método *evaluate prequential*

Figura 16 Grácos dos resultados do *Evaluate Prequential* para bases articiais com ruído e/ou mudança de conceito.

4.4.3.2 Interleaved Chunks

As Tabelas 33, 34 e 35 mostram, respectivamente, a precisão, a concordância *kappa* e o tempo que cada algoritmo obteve para cada uma das bases articiais, com mudança de conceito e ruído, utilizadas. A Figura 17 apresenta os resultados para a precisão e para o *kappa* em formato de grácos, onde o eixo x representa o algoritmo utilizado, o eixo y representa o resultado e a cor utilizada representa a base correspondente ao resultado.

Tabela 33 Tabela com os resultados da medida de avaliação precisão nas bases articiais com mudança de conceito e/ou ruído, usando o método *interleaved chunks*. Resultados em %

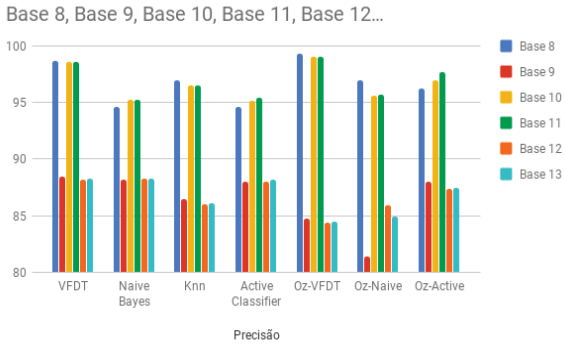
Precisão	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 8	98,69	94,57	96,99	94,58	99,3	96,98	96,26
Base 9	88,42	88,17	86,48	87,96	84,76	81,41	87,97
Base 10	98,6	95,26	96,55	95,14	99,05	95,61	96,96
Base 11	98,59	95,26	96,48	95,44	99,02	95,68	97,71
Base 12	88,19	88,23	86,03	88,02	84,37	85,88	87,33
Base 13	88,27	88,23	86,09	88,16	84,47	84,91	87,46

Tabela 34 Tabela com os resultados da medida de avaliação *kappa* nas bases articiais com mudança de conceito e/ou ruído, usando o método *interleaved chunks*. Resultados em %

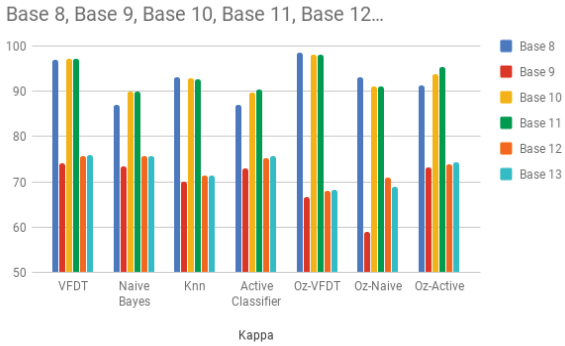
Kappa	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 8	97	86,97	93,07	87,04	98,4	93,08	91,31
Base 9	74,17	73,32	69,9	72,95	66,53	58,87	73,11
Base 10	97,09	90	92,83	89,74	98,02	90,97	93,7
Base 11	97,07	89,99	92,68	90,39	97,97	91,14	95,24
Base 12	75,69	75,67	71,26	75,28	67,95	70,94	73,86
Base 13	75,86	75,66	71,37	75,56	68,15	68,94	74,22

Tabela 35 Tabela com os resultados da medida de avaliação tempo nas bases articiais com mudança de conceito e/ou ruído, usando o método *interleaved chunks*. Resultados em segundos

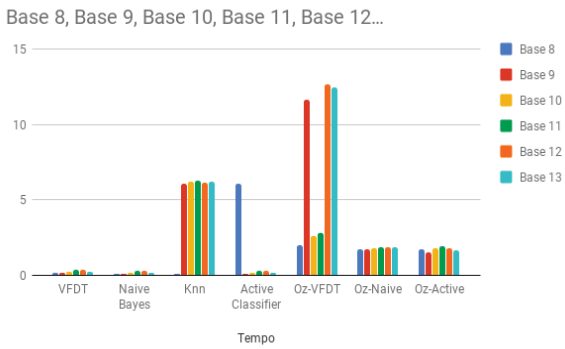
Tempo	VFDT	Naive Bayes	Knn	Active Classier	Oz-VFDT	Oz-Naive	Oz-Active
Base 8	0,2	0,11	0,12	6,09	2,03	1,73	1,73
Base 9	0,19	0,11	6,08	0,11	11,64	1,75	1,5
Base 10	0,25	0,16	6,19	0,17	2,62	1,78	1,77
Base 11	0,34	0,3	6,25	0,28	2,81	1,89	1,91
Base 12	0,36	0,28	6,17	0,3	12,62	1,88	1,81
Base 13	0,23	0,16	6,19	0,14	12,42	1,84	1,66



(a) Gráco ilustrando a medida de avaliação precisão nas bases de dados articiais com mudança de conceito e/ou ruído, utilizando o método *interleaved chunks*



(b) Gráco ilustrando a medida de avaliação *kappa* nas bases de dados articiais com mudança de conceito e/ou ruído, utilizando o método *interleaved chunks*



(c) Gráco ilustrando a medida de avaliação tempo nas bases de dados articiais com mudança de conceito e/ou ruído, utilizando o método *interleaved chunks*

Figura 17 Grácos dos resultados do *Interleaved Chunks* para bases articiais com ruído e/ou mudança de conceito.

4.4.4 Discussão dos resultados

Pode-se perceber que o ruído teve um grande impacto no desempenho de todos os algoritmos. Já a mudança de conceito causou um impacto mínimo no desempenho, isso ocorreu, provavelmente, porque embora o KNN, o *naive bayes* e o **classicador ativo** sejam algoritmos tradicionalmente desenvolvidos para cenários de lote, suas implementações no MOA são voltadas para cenários dinâmicos, de forma que elas permitem a adaptação a novos conceitos.

Outro fator relevante é que a VFDT apresentou resultados extremamente resilientes, não só a mudança de conceito e ruído, mas a todas as mudanças que foram aplicadas as bases com que se trabalhou neste capítulo.

4.5 Teste estatístico

Nesta seção serão apresentados e discutidos os resultados obtidos a partir do uso do teste de *Friedman* (GARCIA et al., 2010) nos algoritmos utilizados nesta monograa.

Nas tabelas de signicância estatística, o símbolo "+"indica que o algoritmo da coluna obteve resultados melhores que o da linha com signicância de 90%, enquanto o "*"representa uma signicância de 95% ou mais.

Os testes estatísticos referentes à medida *kappa* apresentam todas as diferenças estatísticas apontadas pelos testes realizados com a precisão, além deles também apresentarem informações adicionais em alguns casos. Dessa forma, serão apresentados e discutidos nesta seção apenas os resultados dos testes estatísticos referentes ao *kappa*.

4.6 Bases Reais

Serão apresentados e analisados nesta seção os resultados do teste de *Friedman* para as bases de dados reais. As Tabelas 36 e 37 representam, respectivamente, o *rank* e o resultado do teste de signicância estatística de cada um dos algoritmos relativo aos experimentos realizados nas bases de dados reais.

Tabela 36 *Rank* dos algoritmos para bases reais em relação ao valor *kappa*

Ranking:	
Knn	1
Oz-Naive	2
Oz-VFDT	3
VFDT	4
Oz-Active	5
Active	6
Naive	7

Tabela 37 Resultados do teste de signicância estatística dos algoritmos para bases reais em relação ao valor *kappa*

	VFDT	Naive	Knn	Active	Oz-VFDT	Oz-Naive	Oz-Active
VFDT							
Naive			+				
Knn							
Active							
Oz-VFDT							
Oz-Naive							
Oz-Active							

Pode-se observar que existe apenas uma diferença estatística nos resultados entre os resultados do KNN e do *naive bayes*. Consta salientar que o **knn** obteve a melhor colocação com relação às bases reais, mesmo se tratando de um algoritmo desenvolvido primariamente para cenários estáticos. Um dos motivos pelo qual ele obteve um desempenho melhor do que a VFDT, que é um algoritmo desenvolvimento para cenários dinâmicos, é o fato de que, embora as bases sejam simuladas de maneira dinâmica no MOA, esses dados reais provavelmente não sofrem de diculdades tradicionalmente associadas a uxos contínuos de dados. Por exemplo, como essas são bases reais, não se sabe ao certo os momentos em que acontece mudança de conceito.

4.7 Bases articiais

A seguir serão apresentados e discutidos os *ranks* e os resultados do teste de significância estatística para as bases de dados articiais utilizadas neste trabalho, separados em bases com poucos atributos, bases com muitos atributos e bases com mudança de conceito e ruído.

4.7.1 Bases articiais com pouco atributos

Serão apresentados nesta seção os resultados do teste de *Friedman* para as bases de dados com poucos atributos geradas articialmente. As Tabelas 38 e 39 representam, respectivamente, o *rank* e o resultado do teste de signicância estatística de cada um dos algoritmos relativo aos experimentos realizados nas bases de dados articiais com poucos atributos.

Tabela 38 *Rank* dos algoritmos em relação ao valor *kappa*

Ranking:	
Oz-VFDT	1
VFDT	2
Knn	3
Active	4
Oz-Naive	5
Naive	6
Oz-Active	7

Tabela 39 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa*

	VFDT	Naive	Knn	Active	Oz-VFDT	Oz-Naive	Oz-Active
VFDT							
Naive					+		
Knn							
Active							
Oz-VFDT							
Oz-Naive					+		
Oz-Active					+		

Nas bases de dados articiais com pouco atributos percebe-se que o algoritmo que obteve a melhor posição no *rank* foi a VFDT com o uso do *OzaBoost* e que ela foi signicamente superior ao *naive bayes*. Como foi discutido na Seção 4.4.2, o desempenho do *naive bayes* aparenta ser fortemente atrelado à quantidade de atributos e, por meio dos resultados do teste estatístico, foi possível comprovar essa relação.

4.7.2 Bases articiais com muitos atributos

Serão apresentados nesta seção os resultados do teste de *Friedman* para as bases de dados com muitos atributos geradas articialmente. As Tabelas 40 e 41 representam, respectivamente, o *rank* e o resultado do teste de signicância estatística de cada um dos algoritmos relativo aos experimentos realizados nas bases de dados articiais com muitos atributos.

Tabela 40 *Rank* dos algoritmos em relação ao valor *kappa*

Ranking:	
VFDT	1
Oz-VFDT	2
Naive	3
Active	4
Knn	5
Oz-Naive	6
Oz-Active	7

Tabela 41 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa*

	VFDT	Naive	Knn	Active	Oz-VFDT	Oz-Naive	Oz-Active
VFDT							
Naive							
Knn	*				*		
Active							
Oz-VFDT							
Oz-Naive	*				*		
Oz-Active	*	+			*		

Foi visto nos experimentos realizados nas bases com muitos atributos que todos os algoritmos, com exceção do *naive bayes*, são afetados negativamente pelo aumento do número de atributos. Entretanto, foi possível perceber, também, que a VFDT era o algoritmo mais resiliente ás mudanças que impactavam negativamente seu desempenho. Essa resist ncia pode ser observada no fato de que a VFDT obteve resultados superiores a tr s outros algoritmos no teste de signicância.

Consta apontar novamente a relação positiva que o *naive bayes* tem com o aumento da quantidade de atributos, uma vez que esta relação permitiu que ele obtivesse um resultado superior ao *Ozaboost* aplicado ao *active classier* no teste de signicância para bases de dados com muitos atributos, enquanto ele obteve os alguns dos piores resultados para as bases com poucos atributos.

4.7.3 Bases com mudança de conceito e/ou ruído

Serão apresentados nesta seção os resultados do teste de *Friedman* para as bases de dados geradas articialmente com mudança de conceito e ruído. As Tabelas 42 e 43 representam, respectivamente, o *rank* e o resultado do teste de signicância estatística de cada um dos algoritmos relativo aos experimentos realizados nas bases de dados articiais com mudança de conceito e ruído.

Tabela 42 *Rank* dos algoritmos em relação ao valor *kappa*

Ranking:	
VFDT	1
Oz-VFDT	2
Oz-Active	3
Active	4
Knn	5
Oz-Naive	6
Naive	7

Tabela 43 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa*

	VFDT	Naive	Knn	Active	Oz-VFDT	Oz-Naive	Oz-Active
VFDT							
Naive	*						
Knn							
Active							
Oz-VFDT							
Oz-Naive	+						
Oz-Active							

Para as bases com ruído e mudança de conceito geradas articialmente, pode-se perceber que a VFDT continuou obtendo os melhores resultados e que ela foi superior ao *naive bayes* no teste de signicância estatística.

4.7.4 Todas as bases

Por m, serão apresentados nesta seção os resultados do teste de *Friedman* para todas as bases de dados, tanto reais, quanto articiais, utilizadas nesta monograa. As Tabelas 44 e 45 representam, respectivamente, o *rank* e o resultado do teste de signicância estatística de cada um dos algoritmos relativo aos experimentos realizados em todas as bases de dados.

Tabela 44 *Rank* dos algoritmos em relação ao valor *kappa*

Ranking:	
VFDT	1
Oz-VFDT	2
Knn	3
Active	4
Naive	5
Oz-Naive	6
Oz-Active	7

Tabela 45 Resultados do teste de signicância estatística dos algoritmos em relação ao valor *kappa*

	VFDT	Naive	Knn	Active	Oz-VFDT	Oz-Naive	Oz-Active
VFDT							
Naive	*				*		
Knn	*				*		
Active	*				*		
Oz-VFDT							
Oz-Naive	*				*		
Oz-Active	*				*		

Quando executado para todas as bases de dados, o teste de *Friedman* indicou, claramente, que a VFDT, com e sem o uso do *OzaBoost* são superiores no tratamento de uxos contínuos de dados, uma vez que o algoritmo superou todos os outros no teste.

4.8 Considerações nais do capítulo

Neste capítulo foram apresentados e discutidos os resultados dos experimentos propostos nesta monograa. Para isso, tanto os testes gerais dos algoritmos, quanto os testes estatísticos, foram divididos em função das bases de dados reais, articiais com poucos atributos, articiais com muitos atributos e articiais com mudança de conceito e ruído. Ao m do capítulo foi aplicado o teste de *Friedman* a todos os experimentos, com o objetivo de descobrir se algum dos algoritmos pudesse ser considerado "melhor"que os outros para os diferentes uxos de dados que foram simulados. No capítulo seguinte serão apresentadas as conclusões retiradas desses experimentos.

5 Conclusão

A proposta desta monograa consistiu na realização de um estudo comparativo entre algoritmos de classicação para uxos contínuos de dados. A motivação desse estudo era comparar o desempenho de cada uma das estratégias de classicação selecionadas para uxos contínuos, em diferentes bases de dados reais e articiais. Essa comparação foi feita de forma a determinar, por meio do uso de um teste estatístico, se algum dos algoritmos era superior aos demais para os cenários trabalhados.

Outro intuito deste projeto de pesquisa foi o de demonstrar a importância do estudo de cenários dinâmicos, assim como suas diferenças em relação a cenários de lote, uma vez que, como foi discutido na seção 1.1, atualmente é crescente a quantidade de processos que recebem informação por meio de uxos contínuos de dados.

5.1 Contribuições

A principal contribuição oferecida por este trabalho é uma análise experimental que utiliza de um conjunto de 5 diferentes algoritmos de classicação, um conjunto de 17 bases de dados articiais e 3 reais, com medidas e metodologias adequadas para uxos contínuos de dados. O trabalho também traz como contribuição um estudo comparativo entre esses algoritmos, validado por meio de um teste estatístico.

5.2 Conclusões obtidas

Por meio dos experimentos realizados no capítulo 4, foi possível observar que, no contexto geral das bases utilizadas, a VFDT teve um desempenho superior a todos os outros algoritmos selecionados, apresentando resultados de precisão e *kappa* signicativamente melhores, assim como um tempo de execução que, embora não tão bom quanto o **naive bayes** e o classicador ativo, ainda foi relativamente baixo. Esse resultado demonstra que a incorporação de estratégias de atualização do modelo em função de novos dados relevantes e de dados antigos que perderam signicância, causam uma real diferença para o tratamento de uxos contínuos de dados, uma vez que tornam os algoritmos mais resiliente às diculdades existentes na mineração de dados em cenários dinâmicos.

5.3 Trabalhos futuros

Baseado nesta monograa, propõe-se, para possíveis trabalhos futuros, a incorporação de novos algoritmos aos experimentos, bem como a diversicação das bases de dados

articiais e reais utilizadas, de forma a ser possível justicar melhor o comportamento de cada algoritmo em cada uma delas.

Os resultados do estudo comparativo forneceram indícios do que poderia ser melhorado em cada algoritmo e, um fator comum que impactou drasticamente o desempenho de todos eles foi o ruído. Dessa forma, seria interessante trabalhar no desenvolvimento de algoritmos que consigam lidar melhor com a presença de ruídos nos dados.

Refer ncias

AGGARWAL, C. C. *Data streams: models and algorithms*. [S.l.]: Springer Science & Business Media, 2007. v. 31. Citado 2 vezes nas páginas 18 e 31.

ANYANWU, M. N.; SHIVA, S. G. Comparative analysis of serial decision tree classication algorithms. *International Journal of Computer Science and Security*, v. 3, n. 3, p. 230240, 2009. Citado na página 21.

ARARIBóIA, R. *Inteligencia Articial: Um Curso Prático* . [S.l.]: Instituto de Lógica e Tecnologia da Ciencia, 1988. ISBN 8521605919, 9788521605911. Citado na página 21.

BIFET, A. et al. Moa: Massive online analysis. *J. Mach. Learn. Res.*, JMLR.org, v. 11, p. 16011604, ago. 2010. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1756006.1859903>. Citado 4 vezes nas páginas 35, 37, 39 e 43.

FIDELIS, M.; LOPES, H. S.; FREITAS, A. A. Discovering comprehensible classication rules with a genetic algorithm. In: *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*. La Jolla, CA, USA: IEEE, 2000. v. 1, p. 805810. Disponível em: <http://kar.kent.ac.uk/22013/>. Citado 2 vezes nas páginas 13 e 19.

GAMA, J. *Knowledge Discovery from Data Streams*. 1st. ed. [S.l.]: Chapman & Hall/CRC, 2010. ISBN 1439826110, 9781439826119. Citado 9 vezes nas páginas 12, 14, 17, 18, 19, 32, 33, 37 e 40.

GARCIA, S. et al. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 180, n. 10, maio 2010. ISSN 0020-0255. Disponível em: <https://doi.org/10.1016/j.ins.2009.12.010>. Citado 2 vezes nas páginas 42 e 63.

GOLAB, L.; ÖZSU, M. T. Issues in data stream management. *SIGMOD Rec.*, ACM, New York, NY, USA, v. 32, n. 2, p. 514, jun. 2003. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/776985.776986>. Citado 2 vezes nas páginas 12 e 13.

INFORMATION; SCIENCE UNIVERSITY OF CALIFORNIA, I. C. *KDD cup 99*. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>. Citado na página 40.

OZA, N.; RUSSELL, S. Online bagging and boosting. 01 2001. Citado 2 vezes nas páginas 28 e 34.

RISH, I. An empirical study of the naive bayes classier. In: IBM. *IJCAI 2001 workshop on empirical methods in articial intelligence* . [S.l.], 2001. v. 3, n. 22, p. 4146. Citado na página 25.

RUSSELL, S.; NORVIG, P. *Articial Intelligence: A Modern Approach* . 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594. Citado na página 17.

SCHAPIRE, R. E.; FREUND, Y. *Boosting: Foundations and Algorithms*. [S.l.]: The MIT Press, 2012. ISBN 0262017180, 9780262017183. Citado 2 vezes nas páginas 28 e 34.

SCIENCES COLLEGE OF NATURAL RESOURCES, C. S. U. Department of F. *Forest Covertype*. <https://archive.ics.uci.edu/ml/datasets/Covertype>. Citado na página 40.

SETTLES, B. Active learning literature survey. *Computer Sciences Technical Report 1648*, v. 1, n. 1, 2010. Citado 2 vezes nas páginas 30 e 31.

STEFANOWSKI, J.; BRZEZINSKI, D. Stream classication. In: _____. *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer US, 2017. p. 11911199. ISBN 978-1-4899-7687-1. Disponível em: <http://dx.doi.org/10.1007/978-1-4899-7687-1_908>. Citado na página 13.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367. Citado 8 vezes nas páginas 5, 12, 13, 17, 20, 21, 29 e 36.