UNIVERSIDADE FEDERAL DO ACRE

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

UMA EXTENSÃO DA FERRAMENTA WEKA PARA AVALIAÇÃO DE TAREFAS PREDITIVAS

RIO BRANCO

2017

MAX WILLIAN SOARES LIMA

UMA EXTENSÃO DA FERRAMENTA WEKA PARA AVALIAÇÃO DE TAREFAS PREDITIVAS

Monografia apresentada como exigência final para obtenção do grau de bacharel em Sistemas de Informação da Universidade Federal do Acre.

Prof. Orientador: Manoel Limeira de Lima Júnior Almeida

RIO BRANCO

2017

TERMO DE APROVAÇÃO

MAX WILLIAN SOARES LIMA

UMA EXTENSÃO DA FERRAMENTA WEKA PARA AVALIAÇÃO DE TAREFAS PREDITIVAS

Esta monografia foi apresentada como trabalho de conclusão de Curso de Bacharelado em Sistemas de Informação da Universidade Federal do Acre, sendo aprovado pela banca constituída pelo professor orientador e membros abaixo mencionados.

Compuseram a banca:

Prof. Manoel Limeira de Lima Júnior Almeida, Me.

Curso de Bacharelado em Sistemas de Informação

Prof. Daricelio Moreira Soares, Me.

Curso de Bacharelado em Sistemas de Informação

Prof. Macilon Araújo Costa Neto, Dr.

Curso de Bacharelado em Sistemas de Informação

Rio Branco, 02 de maio de 2017.

Dedico este trabalho aos meus pais, meu alicerce.

AGRADECIMENTOS

Esse trabalho marca a conclusão de uma etapa. Mesmo sendo uma etapa repleta de dificuldades, com medos e angústias, essa conclusão é tida de forma satisfatória e cheia de agradecimentos a serem feitos.

Primeiramente ao meu pai e a minha mãe, sem eles os meus esforços não teriam sentido. Obrigado por serem sempre o meu maior apoio e motivação.

Agradeço também ao meu professor e orientador Manoel Limeira de Lima Júnior Almeida, pela dedicação, apoio e paciência no desenvolvimento deste trabalho. Agradeço ao professor Daniel Augusto Nunes da Silva pela atenção e auxílio para conseguir chegar aos resultados deste trabalho. Em caráter de formação, agradeço também a todo o corpo docente que teve, de forma direta ou indireta, participação nessa conquista, em especial ao professor Macilon Araújo Costa Neto, pelo seu esforço e comprometimento nessa realização.

Agradeço também aos meus colegas de curso e amigos para a vida. Obrigado Eleandro, Giuliano, Luisa, Wallison e Rodrigo.

Meu sentimento é de satisfação e gratidão, a todos vocês meu sincero obrigado!

"A ciência é uma perversão de si mesma, a menos que tenha como fim último melhorar a humanidade."

(Nikola Tesla)

RESUMO

O crescente armazenamento de dados abre espaço para o processo de descoberta de conhecimento em bases de dados (KDD). Analistas de dados lançam mão da mineração de dados, uma das etapas do KDD, em busca de extrair informações valiosas para várias áreas como, por exemplo, finanças, ciência, medicina e negócios. Ferramentas de mineração de dados, como a WEKA, fornecem recursos para auxiliar o trabalho dos analistas, como por exemplo, nas tarefas preditivas. Essas ferramentas disponibilizam algoritmos preditivos e métodos de avaliação para medir a capacidade de predizer determinadas variáveis de uma base de dados. Essa capacidade é avaliada pela ferramenta a partir de métodos tradicionais conhecidos na literatura. No entanto, esses métodos apresentam uma limitação ao avaliar bases de dados onde suas instâncias apresentam dependência cronológica. Nesse contexto, o presente trabalho desenvolveu, disponibilizou e avaliou uma extensão para a ferramenta WEKA que possibilita a avaliação de algoritmos preditivos nessas bases, apresentando resultados satisfatórios quanto a sua implementação e aceitação de usuários da WEKA.

Palavras-chave:

WEKA, KDD, mineração de dados, classificação, regressão, *plug-in*, extensão

ABSTRACT

The growing data storage opens up space for knowledge discovery process in databases (KDD). Data analysts make use of data mining, one of KDD's stages, in the pursuit of extracting valuable information for areas such as finance, science, medicine, and business. Data mining tools, such as WEKA, provide resources to assist the work of analysts, such as in predictive tasks. These tools provide predictive algorithms and evaluation methods to measure the ability to predict certain variables in a database. This capability is evaluated by the tool from traditional methods known in the literature. However, these methods have a limitation when evaluating databases where their instances are chronologically dependent. In this context, the present work developed, made available and evaluated an extension for the WEKA tool that allows the evaluation of predictive algorithms in these bases, having satisfactory results on its implementation and acceptance of WEKA users.

Key-words:

WEKA, KDD, data mining, classification, regression, plug-in, extension

LISTAS DE FIGURAS

LISTAS DE QUADROS

LISTAS DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O avanço das tecnologias de informação proporciona a geração e armazenamento de dados em proporção exponencial, esses dados são armazenados diariamente em diversos repositórios. Han, Kamber e Pei (2012) afirmam que já se chega na casa dos *terabytes* e *petabytes* passando todos os dias em redes de computadores e em variados dispositivos de armazenamento, provenientes de diversas áreas da sociedade moderna, como o comércio, a ciência, engenharia e a medicina. Nesse contexto, empresas e instituições precisam armazenar e lidar com dados transacionais, informações de estoque, descrição de produtos, dados de clientes e finanças. Cientistas e engenheiros coletam dados de experimentos científicos, performance de sistemas, avaliação de processos e observação de ambientes. De acordo com Witten, Frank e Hall (2011), a análise inteligente desses dados pode trazer conhecimentos valiosos.

A fim de obter tais conhecimentos, os analistas de dados, especialistas em realizar a análise de dados, lançam mão do processo de descoberta de conhecimento em bases de dados (*Knowledge-Discovery in Databases* – KDD) para que possam descobrir padrões, relacionamentos e detalhar o comportamento destes dados. Esse processo envolve algumas etapas, mas para Han, Kamber e Pei (2012), a principal delas é a de mineração de dados, que consiste no processamento dos dados para obtenção das novas informações.

Na mineração de dados existem duas tarefas de modelagem preditiva, classificação e regressão, ambas são utilizadas quando se busca predizer o valor de uma variável da base de dados, como por exemplo, a quantidade de um produto em estoque ou a performance de um computador.

Para isso, algoritmos são treinados com um conjunto de dados dessa base, gerando um modelo preditivo, e posteriormente testados em outro conjunto de dados,

com base no modelo gerado. O desempenho desses algoritmos pode ser definido por medidas de interesse, como a taxa de erros ou o coeficiente de correlação entre os valores reais e os previstos, calculadas a partir das instâncias do conjunto de dados usados nos testes (WITTEN; FRANK; HALL, 2011).

Considerando a complexidade destas tarefas, existem ferramentas computacionais para auxiliar na realização do processo de KDD, como por exemplo as ferramentas KMine, R, e Rattle. A ferramenta de análise de conhecimento da universidade de Waikato, comumente conhecida como WEKA (*Waikato Environment for Knowledge Analysis*), também se enquadra como uma dessas ferramentas, buscando auxiliar os seus usuários nesse processo através da integração de diversos recursos e funcionalidades da mineração de dados. Contudo, os métodos de avaliação de algoritmos preditivos implementados pela ferramenta WEKA possuem uma inconsistência quando se pretende predizer valores em uma base de dados onde as instâncias apresentam uma dependência cronológica. Os métodos podem fornecer instâncias do futuro para o conjunto de treino e instâncias do passado para o conjunto de teste, podendo gerar resultados incoerentes com a realidade.

Sabendo disso, este trabalho desenvolveu uma extensão da WEKA que possibilita a avaliação de algoritmos preditivos aplicados em bases com tal característica, realizando, após isso, a avaliação da extensão através de: testes em uma base de dados real e; verificação da aceitação dos usuários já familiarizados com a ferramenta WEKA.

1.1 PROBLEMA DA PESQUISA

No cenário de modelagem preditiva, o analista de dados conta com variados algoritmos disponibilizados pelas ferramentas para predizer o valor de uma variável. Os algoritmos geram um modelo preditivo treinando com um conjunto de instâncias

da base, e assim testam o modelo tentando predizer o valor da variável em instâncias de um conjunto separado para teste.

Considerando que esses algoritmos utilizam diferentes técnicas para gerar o seu modelo preditivo, suas previsões são avaliadas e os resultados são comparados para que se verifique qual algoritmo apresenta a melhor capacidade preditiva para o domínio em questão. A ferramenta WEKA implementa métodos tradicionais de avaliação, como o *holdout* e o *cross-validation*, que dividem a base de dados de acordo com parâmetros de entrada. Esses parâmetros definem como as instâncias da base formarão os conjuntos de treino e teste. O *holdout*, no entanto, determina o modelo preditivo treinando com apenas uma porção dos dados iniciais, podendo não ser representativo. Já o *cross-validation* realiza múltiplos testes até treinar com toda a base, fazendo desse um método amplamente utilizado (HAN; KAMBER; PEI, 2012).

Contudo, existem bases de dados em que suas instâncias são organizadas sequencialmente, de forma cronológica, onde os períodos temporais de registro dessas instâncias influenciam na predição do valor de uma determinada variável. Como por exemplo, a temperatura coletada em determinada época do ano que apresenta um valor, mas pode apresentar outro valor completamente diferente no ano seguinte. Para esse tipo de base, o método *cross-validation,* em determinado momento, utiliza instâncias futuras para treinar o algoritmo e depois testa o modelo em instâncias do passado. Como por exemplo, na tarefa de predizer o comportamento de *pull requests* em bases com dependência cronológica (DE LIMA JÚNIOR, et al., 2015), onde o problema da avaliação cruzada de classificadores é utilizar instâncias do futuro para predizer uma informação do passado.

A partir deste contexto, se estabelece o questionamento: como utilizar a ferramenta WEKA para avaliar os algoritmos preditivos em bases de dados que apresentam dependência cronológica entre suas instâncias?

1.2 OBJETIVOS DA PESQUISA

O objetivo geral a que se propõe essa pesquisa é desenvolver e avaliar uma extensão para a ferramenta WEKA que possibilite a avaliação de algoritmos preditivos de forma coerente para os casos onde as instâncias das bases possuem dependência cronológica.

Para atingir o objetivo geral, têm-se os seguintes objetivos específicos:

a) Realizar revisão bibliográfica acerca dos processos de KDD, em específico a mineração de dados e as tarefas preditivas;
b) Levantar requisitos e especificar a extensão a ser desenvolvida;
c) Implementar a extensão da ferramenta WEKA;
d) Avaliar a extensão com: experimentos em uma base real e; aplicação de questionário a usuários.
e) Disponibilizar a extensão por meio de um pacote instalável da ferramenta WEKA;

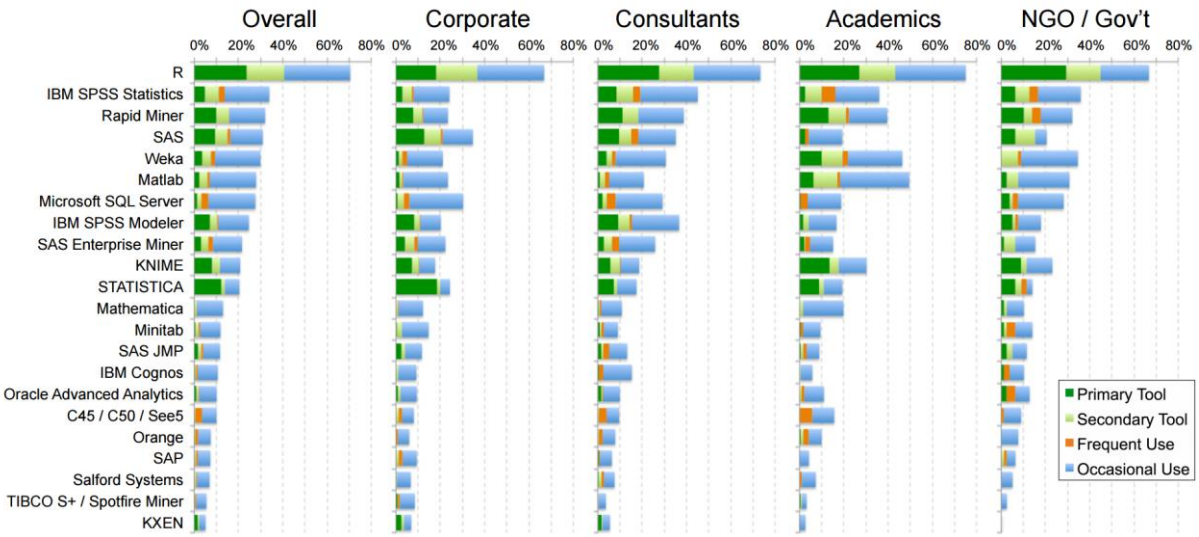
1.2 JUSTIFICATIVA DA PESQUISA

De acordo com Piatetsky-Shapiro (1996), o processo de descoberta de conhecimento tem um alto valor para a sociedade, possibilitando vantagens, eficiência e melhores serviços em diversas áreas de aplicação.

Tendo em vista o problema apresentado na Seção 1.1, este trabalho busca proporcionar resultados coerentes na avaliação de algoritmos preditivos na etapa de mineração de dados do processo de KDD, justificado pela importância dessa etapa e pela necessidade de uma alta precisão na realização de suas técnicas (HAN; KAMBER; PEI, 2012).

A escolha de solucionar o problema através do desenvolvimento de uma extensão para a ferramenta WEKA se deu pela sua grande utilização entre os analistas de dados, estando entre as opções de ferramentas mais escolhidas para a realização das tarefas no cenário de mineração de dados (REXER, 2013), destacando-se como uma solução de código aberto ao lado de softwares proprietários, como pode ser visto na Figura 1.

Figura 1 – Uso das ferramentas de mineração de dado.



Fonte: Rexer (2013)

Além disso, a ferramenta WEKA permite a integração das extensões desenvolvidas por meio da geração de pacotes instaláveis, que podem ser facilmente compartilhados e adotados por outros usuários da ferramenta (BOUCKAERT et al., 2014), justificando assim sua escolha para implementação, automação e disponibilização da solução proposta.

1.3 METODOLOGIA

A pesquisa realizada neste trabalho pode ser classificada, de acordo com Wazlawick (2014), em relação a 3 diferentes critérios, sendo:

a) Quanto à sua natureza: é uma pesquisa original, por se tratar do desenvolvimento de uma nova extensão para a ferramenta WEKA;
b) Quanto ao seu objetivo: se trata de uma pesquisa exploratória, pois fornece uma alternativa aos métodos tradicionais de avaliação, através da implementação de uma extensão para a WEKA;
c) Quanto ao procedimento técnico: é classificada como uma pesquisa experimental, já que foi introduzido um novo método de avaliação de algoritmos através de uma extensão para a ferramenta WEKA, e observado experimentos com tarefas preditivas e aceitação de usuários da ferramenta.

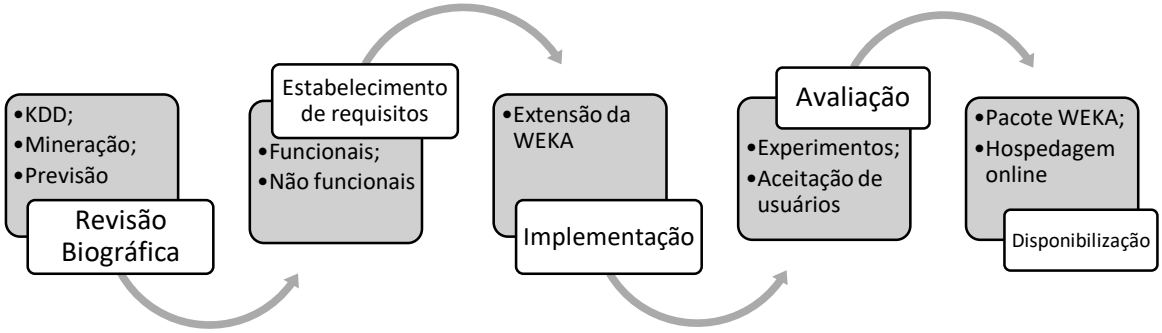
A realização desse trabalho foi dividida em 5 etapas:

a) Na primeira etapa: foram realizados o levantamento e o estudo da literatura disponível a respeito das áreas que envolvem essa pesquisa, como o processo de descoberta de conhecimento, mineração de dados e suas tarefas, algoritmos preditivos, medidas de interesse e métodos de avaliação. Esta etapa aconteceu de forma iterativa, possibilitando a sua atualização sempre que necessário durante o desenvolvimento do trabalho;
b) Na segunda etapa: deu-se início ao estabelecimento de requisitos da extensão, especificando as suas funcionalidades e limitações;
c) Na terceira etapa: a extensão foi implementada em linguagem de programação Java como uma extensão da ferramenta WEKA;
d) Na quarta etapa: a extensão desenvolvida foi estruturada em um pacote instalável da ferramenta WEKA, possibilitando a sua disponibilização através de uma plataforma online de compartilhamento de projetos;
e) Na quinta etapa: a extensão foi avaliada a partir da análise de suas funcionalidades com experimentos em uma base de dados do Instituto

Nacional de Meteorologia (INMET)[1]. Além disso, a aceitação da extensão foi avaliada a partir da aplicação de um questionário para usuários familiarizados com a WEKA.

A Figura 2 demonstra de forma simplificada as cinco etapas estabelecidas para a realização desse trabalho.

Figura 2 – Etapas para a realização do trabalho.



Fonte: Elaboração própria.

Para a implementação da extensão foram utilizadas as seguintes ferramentas e tecnologias:

a) Linguagem de programação Java[2];
b) O ambiente de desenvolvimento integrado (*Integrated Development Environment* – IDE) Eclipse[3].
c) API (*Application Programming Interface*) da WEKA[4].

[1] http://www.inmet.gov.br/
[2] https://www.oracle.com/java/
[3] https://eclipse.org/
[4] http://www.cs.waikato.ac.nz/ml/weka/

1.4 ORGANIZAÇÃO DO ESTUDO

Incluindo esse capítulo de introdução, a organização deste trabalho está distribuída em 5 capítulos.

No Capítulo 2, são apresentados conceitos básicos sobre o processo de descoberta de conhecimento, mineração de dados, tarefas de modelagem preditiva e alguns algoritmos, métodos de avaliação e medidas de interesse.

O Capítulo 3 é destinado à explicação teórica do método e sua implementação, a visão geral e integração com a ferramenta WEKA e disponibilização da extensão como um pacote instalável.

O Capítulo 4 detalha a avaliação da extensão, por meio da análise de suas funcionalidades e aceitação dos usuários.

No Capítulo 5, são feitas as considerações finais e recomendações para trabalhos futuros.

2 MINERAÇÃO DE DADOS

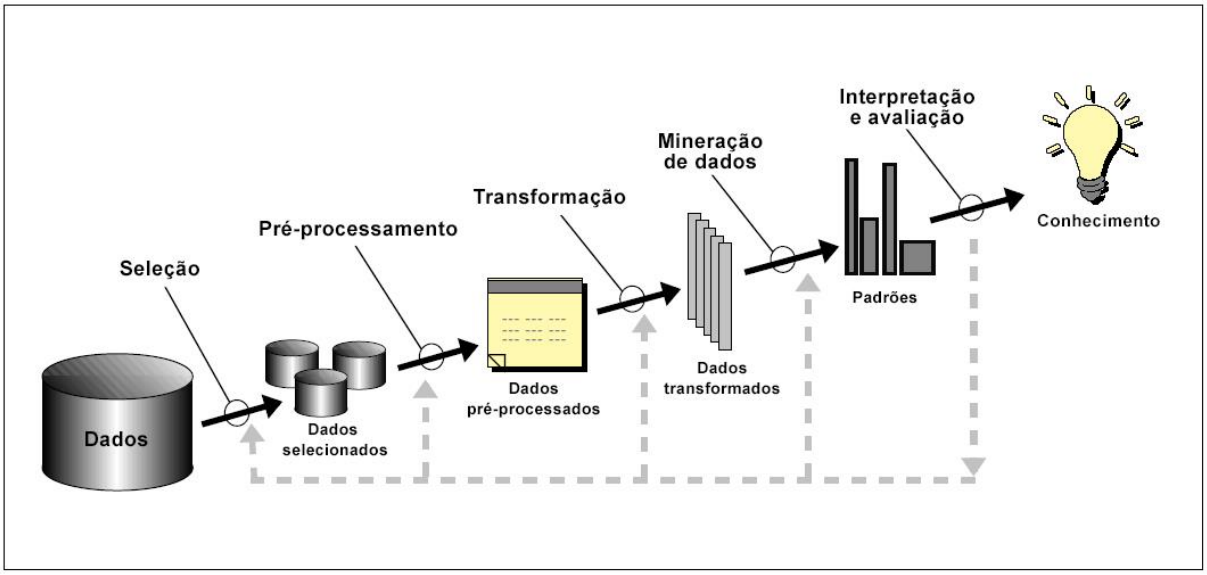
Descobrir conhecimento significa, para Wives (2002), identificar, receber informações que tenham relevância e então processá-las e agregá-las a um conhecimento prévio, possibilitando a resolução de uma determinada situação ou problema.

É notório que a quantidade de informações encontrada atualmente, seja em empresas, organizações ou redes sociais, é muito grande. Morais e Ambrósio (2007) afirmam que devido a isso é necessário a utilização de mecanismos automáticos, como softwares de computador, para o processamento dessas informações, o que deu origem a Descoberta de Conhecimento Apoiada por Computador, ou apenas Descoberta de Conhecimento, processo voltado a analisar e manipular dados/informações e fornecer conhecimento.

A descoberta de conhecimento é uma área dinâmica e evolutiva, envolvendo integrações com outras áreas de conhecimento como Estatística, Inteligência Artificial e Banco de Dados. Para Witten, Frank e Hall (2011), os padrões extraídos devem ser além de confiáveis, compreensíveis e úteis, podendo empregar o conhecimento com utilidade e tirar proveito de alguma vantagem, seja de forma científica ou comercial.

De acordo com Han, Kamber e Pei (2012), muitas pessoas tratam o processo de KDD e a mineração de dados como sinônimos, no entanto, a mineração de dados é contextualizada como apenas uma etapa dentro do processo de KDD como é possível ver na Figura 3, mas certamente pode ser vista como a etapa mais importante.

Figura 3 – Processos do KDD.



Fonte: Adaptado de Piatetsky-Shapiro (1996).

A Figura 3 demonstra todo o processo de KDD, descrito por Piatetsky-Shapiro (1996) este processo se dá pela sequência iterativa das seguintes etapas:

a) **Seleção**: onde os dados relevantes para a análise são coletados do conjunto de dados inicial;
b) **Pré-processamento**: etapa para limpar os dados selecionados, removendo inconsistências e, se necessário, integrando múltiplos dados;
c) **Transformação**: onde os dados são transformados e consolidados com o objetivo de enriquecê-los, através de tarefas como agrupamento de dados, transformação de tipos, normalização utilizando intervalos definidos;
d) **Mineração de dados**: processo que se baseia na aplicação de algoritmos com o objetivo descobrir padrões;
e) **Interpretação e avaliação**: etapa final do processo onde o especialista de domínio verifica se os resultados contribuem para a solução do problema, identificando a qualidade dos padrões descobertos, com o auxílio das métricas adequadas a cada caso.

De acordo com Piatetsky-Shapiro (1996), o processo de interpretação e análise de dados, de maneira manual, pode ser considerada uma forma tradicional de transformar dados em conhecimento. No entanto, quanto maior a quantidade de dados, maior será a complexidade para realizar esta análise, tornando a prática inviável em determinados domínios onde os dados crescem de maneira exponencial.

Han, Kamber e Pei (2012) afirmam que esse crescimento de dados é o resultado da computação, do rápido desenvolvimento de poderosas coleções de dados e também das ferramentas de armazenamento. Logo, a mineração de dados pode ser vista como resultado da evolução dos dados e das tecnologias utilizadas para analisá-los e interpretá-los, já que, segundo Witten, Frank e Hall (2011), o avanço dessas tecnologias possibilitou que os dados fossem armazenados de forma eletrônica e que suas buscas pudessem ocorrer de maneira automatizada, ou, pelo menos assistida, por computador, sendo essa uma característica marcante da mineração de dados.

Na sua essência, a mineração de dados é a aplicação de algoritmos específicos em um determinado conjunto de dados, com a finalidade de verificar padrões de comportamento (PIATETSKY-SHAPIRO, 1996).

2.1 TAREFAS DE MINERAÇÃO DE DADOS

A mineração de dados pode ser aplicada em diversos domínios, como a medicina ou a astronomia, e com diversos objetivos, como predizer ou agrupar instâncias. Para isso, dispõe de diferentes conjuntos de métodos e técnicas específicas para cada objetivo, as tarefas (CHEN; HAN; YU, 1996). Apesar da existência de diferentes tarefas, Hand, Mannila e Smyth (2001) afirmam que é possível categorizá-las de cinco maneiras:

a) **Análise exploratória de dados:** basicamente consiste em explorar os dados sem nenhuma ideia clara do que se procura. Tipicamente, as tarefas desta categoria são interativas e visuais, fazendo o uso de gráficos para mostrar pequenas dimensões dos dados;
b) **Modelagem descritiva:** tem objetivo de descrever os dados e apresentá-los de forma compreensível, demonstrando a distribuição e a relação entre as variáveis;
c) **Modelagem preditiva:** neste caso, o objetivo é a construção de modelos que permitam a predição do valor de determinada variável baseando-se nos valores já conhecidos de outras variáveis;
d) **Extração de regras e padrões:** esta categoria não está preocupada com a construção de modelos, e sim com a detecção de padrões, como a combinação de itens que ocorrem frequentemente em bases de dados de compras;
e) **Recuperação por conteúdo:** nesta categoria o usuário apresenta um padrão de interesses, a fim de encontrar padrões similares no conjunto de dados. As técnicas são geralmente usadas em bases de dados com imagens e/ou textos, largamente empregadas em motores de busca na web, por exemplo.

A escolha da técnica de mineração de dados, que deve ser utilizada no processo de KDD, depende do domínio da aplicação e também dos objetivos que se pretende alcançar. O estudo de caso desse trabalho se concentra na tarefa de modelagem preditiva. A básica distinção entre modelagem preditiva e modelagem descritiva, é que na preditiva há apenas uma variável como objetivo, enquanto nos problemas descritivos não há uma variável central no modelo (HAN; KAMBER; PEI, 2012).

A tarefa de previsão é aplicada quando o analista de dados deseja conhecer qual será o comportamento de determinada variável em novas instâncias de dados. Para a variável cujo valor se pretende predizer é dado o nome de variável dependente, enquanto que as variáveis usadas para fazer a predição são chamados de variáveis independentes (TAN; STEINBACH; KUMAR, 2006).

Há duas tarefas básicas no cenário de previsão: classificação e regressão. Basicamente, quando a variável dependente é categórica se faz o uso da classificação, já quando esta variável apresenta valores numéricos, é utilizado a tarefa de regressão (HAND; MANNILA; SMYTH, 2001). As próximas seções são destinadas a essas duas tarefas, assim como seus métodos de validação, alguns algoritmos e medidas.

2.2 CLASSIFICAÇÃO

A classificação é aplicada quando a variável dependente assume valores discretos. Estes valores são as classes nas quais a variável poderá ser classificada. Como por exemplo, classificar uma pessoa como boa ou má pagadora de empréstimos, ou o tipo de hábito alimentar de um animal como carnívoro, herbívoro ou onívoro.

A tarefa de classificação é um processo de dois passos, onde o primeiro passo é destinado ao aprendizado, ou seja, para a construção do modelo preditivo, e o segundo é o passo da classificação, onde o modelo é usado para predizer qual classe se encaixa a determinada variável em análise (HAN; KAMBER; PEI, 2012). Os algoritmos de classificação, chamados de classificadores, são os responsáveis pela realização destes dois passos.

No primeiro passo, o classificador receberá os dados para **treino**, onde ele deverá aprender as associações e quais os padrões que levam a variável dependente ter esse determinado valor. Em outras palavras, descobrir porque ela pertence a tal classe. Gerando assim o modelo de classificação, que pode ser entendido como uma função $y = f(\boldsymbol{x})$ que pode predizer qual é a classe y de acordo com uma dada instância $\boldsymbol{x}$ (HAN; KAMBER; PEI, 2012).

No segundo passo, o modelo construído será utilizado para a classificação. Os dados agora serão utilizados para **teste**, para que assim se possa avaliar o desempenho do classificador nessa determinada base de dados (HAN; KAMBER; PEI, 2012).

De acordo com Witten, Frank e Hall (2011), nos problemas de classificação é natural mensurar a performance do classificador de acordo com a sua taxa de erros, o que é feito simplesmente analisando a qual classe a variável pertence e a qual classe o classificador previu que esta pertencia, se o classificador acertou, é um sucesso, se ele errou, é contado como um erro.

2.3 ALGORITMOS DE CLASSIFICAÇÃO

Baseado no teorema de Bayes, o algoritmo **NaiveBayes** é um algoritmo simples utilizado na construção de modelos preditivos na tarefa de classificação, e é chamado assim[5] porque ele se apoia em duas suposições simples. Este algoritmo assume que as variáveis preditivas são condicionalmente independentes da classe, e nenhuma variável oculta ou latente influencie na previsão (GEORGE; PAT, 1995).

O NaiveBayes calcula a probabilidade de determinada instância ser classificada para cada possibilidade. George e Pat (1995) dizem que para isso tem-se C como uma variável aleatória representando a classe da instância a ser testada, e X é o vetor das variáveis aleatórias que representa as variáveis associadas à C. Assim, para calcular a probabilidade de cada classe C condicionada a X , utiliza-se a Equação 1:

$$probabilidade(C \mid X) = \frac{p(C \mid X)p(X)}{p(C)} \quad (1)$$

[5] *Naive*, do inglês, significa algo ingênuo.

Implementado na WEKA como **IBk**, esse é um tipo de algoritmo de vizinho mais próximo (*k-Nearest Neighbors*, ou simplesmente k-NN). Esse algoritmo normaliza a distância entre as variáveis com relação às suas distribuições de frequência para diminuir sua sensibilidade e distribuições distorcidas, e tem uma política simples para tolerar valores vazios, onde estes são assumidos como sendo maximamente diferentes do valor presente, e em caso de ambos estarem vazios, $f(x_i, y_i)$ resultará em 1. A implementação é baseada na similaridade das instâncias, e é dada pela Equação 2:

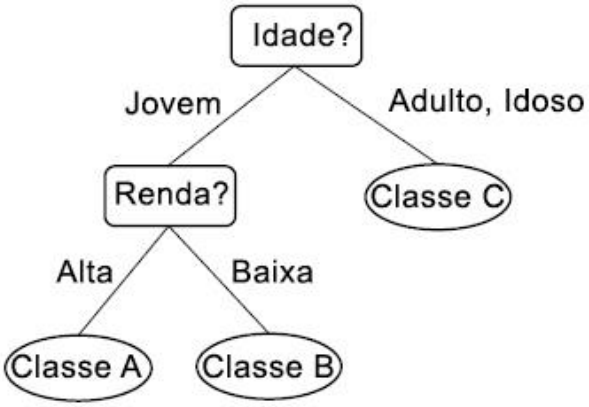
$$similaridade = \sqrt{\sum_{i=1}^{n} f(x_i, y_i)} \ , onde\ n \text{ é } a\ quantidade\ de\ variáveis\ da\ instância \tag{2}$$

É definido $f(x_i, y_i) = f(x_i - y_i)^2$ quando a variável assume valores numéricos, e $f(x_i, y_i) = f(x_i \neq y_i)$ para valores categóricos, onde x representa o valor previsto e y o valor real da variável (AHA; KIBLER; ALBERT, 1991). Dessa forma, o IBk pode ser utilizado tanto para problemas de classificação quanto para problemas de regressão.

De acordo com Breiman (2001), melhorias significantes no desempenho de classificações são resultados da utilização de estruturas de árvores[7]. Chamadas de árvores de decisão, essas estruturas são organizadas em um fluxograma, onde cada nó denota um teste em um valor da variável, cada ramo representa um resultado do teste e as folhas da árvore representam as classes (HAN; KAMBER; PEI, 2012). A Figura 4 demonstra um exemplo de árvore de decisão utilizada para classificar determinada variável entre as classes A, B ou C.

[7] Uma árvore é uma forma de estrutura dados definido por um conjunto finito de elementos denominados nós, onde há um nó especial chamado raiz, os demais são conjuntos subárvores do nó raiz, cada qual por sua vez uma árvore (SZWARCFITER; MARKENZON, 1994).

Figura 4 – Exemplo de árvore de decisão.



Fonte: Adaptado de Han, Kamber e Pei (2012).

A WEKA dispõe do algoritmo **J48** que é uma implementação do algoritmo C4.5 desenvolvido por Quinlan (1993). Dessa forma, o J48 utiliza a técnica de árvores de decisão para a construção do seu modelo preditivo, assim a árvore é aplicada para cada instância no momento do treino, classificando a variável dependente. Ao construir uma árvore, o J48 ignora os valores ausentes, para isso, calcula o valor para determinado item com o que pode ser previsto utilizando o conhecimento sobre os valores das demais variáveis das instâncias utilizadas no treino (PATIL; SHEREKAR, 2013).

2.4 REGRESSÃO

Quando a variável dependente assume valores numéricos, ou seja, não categóricos como no caso da classificação, a tarefa de previsão utilizada é chamada de regressão. Para Piatetsky-Shapiro(1996), regressão é aprender uma função que mapeia uma instância de dados de acordo com o real valor da variável dependente.

A tarefa de regressão tem basicamente o mesmo princípio de funcionamento da tarefa de classificação, divide-se nos passos de treinar com os dados da base, construindo o modelo para previsão e, depois, realizar o teste para predizer o valor da variável dependente. A diferença é que esse valor na regressão é numérico, o que difere como os algoritmos serão aplicados e como as medidas de interesse serão calculadas.

Com foco na previsão de variáveis com valores numéricos, a tarefa de regressão tem grande aplicabilidade, como por exemplo, predizer a quantidade de biomassa presente em uma floresta pelas medidas capturadas por micro-ondas, ou estimar a probabilidade de sobrevivência de um paciente de acordo com os testes nos dados de diagnóstico, ou predizer a demanda de consumidores de determinado produto de acordo com os dados das suas despesas publicitárias (PIATETSKY-SHAPIRO, 1996).

2.5 ALGORITMOS DE REGRESSÃO

Segundo Witten, Frank e Hall (2011), quando a variável que se pretende classificar é do tipo numérica, regressão linear é a técnica tradicional a se considerar. Implementado pela WEKA através do algoritmo **LinearRegression**, essa técnica consiste em, de acordo com Witten, Frank e Hall (2011), expressar o valor numérico da variável dependente com uma combinação linear das demais variáveis, com seus respectivos pesos: $x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + + w_k a_k$, onde x, a e e representam, respectivamente, a classe, o valor da variável e os pesos. Com os cálculos feitos, se tem a função que determinará o valor de x de acordo com os demais valores da instância, sendo esse o modelo preditivo gerado.

A técnica de regressão linear, juntamente a de árvores de decisão, servem como base para outra técnica utilizada na tarefa de regressão, chamada M5. A ferramenta WEKA faz a implementação dessa técnica através do algoritmo chamado

M5P. De acordo com Quinlan et. al. (1992), a técnica constrói uma árvore baseada em modelos, porém, a árvore construída pode ter múltiplos modelos lineares em suas folhas. Dessa forma, o algoritmo aprende de maneira eficiente e pode lidar com tarefas de dimensões altas, até centenas de variáveis por instância.

Além desses, o algoritmo **IBk**, apresentado na Seção 2.3, também pode ser utilizado para a construção de modelos preditivos com base em uma variável alvo de valor numérico.

2.6 MÉTODOS DE AVALIAÇÃO PARA ALGORITMOS PREDITIVOS

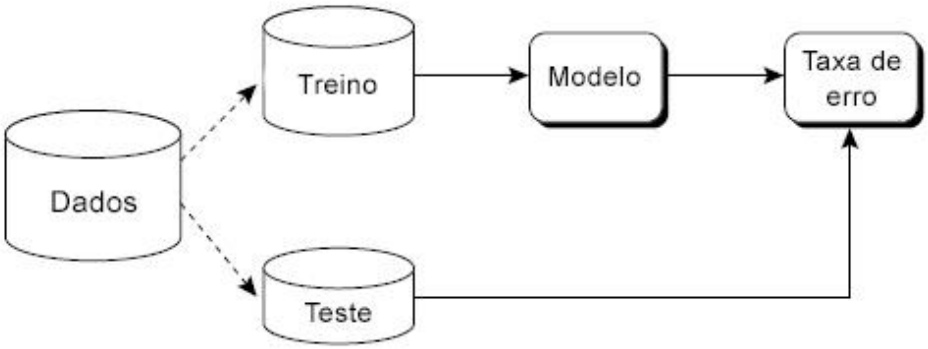
Os algoritmos utilizados no cenário de previsão, independentemente da tarefa de classificação ou regressão, têm o objetivo de gerar o melhor modelo preditivo e assim minimizar os erros nas previsões. Os algoritmos utilizam paradigmas diferentes com a finalidade de previsão e para medir a capacidade preditiva, é necessário a utilização de algum método padronizado para avaliar o desempenho dos algoritmos. Isso permite a comparação de desempenho e a definição de qual algoritmo é melhor utilizar em determinado domínio.

Para Hand, Mannila e Smyth (2001), nas tarefas de previsão, as instâncias de treino vêm de um determinado valor alvo $\boldsymbol{x}$, que é o valor da variável dependente, podendo este ser um valor quantitativo, para regressão, ou um valor categórico, para classificação. Durante os passos de previsão, as instâncias analisados ficam organizados em pares $D = \{(\boldsymbol{x}(1), y(1)), \ldots (\boldsymbol{x}(n), y(n))\}$, de forma que para cada $\boldsymbol{x}$ haverá uma instância y com os demais valores da instância. Logo, $f(x(a), b)$ será a previsão gerada pelo modelo individual de a, usando os parâmetros b. Assim, o desempenho de um algoritmo pode ser medido de forma bem direta, simplesmente analisando a diferença entre o valor previsto $f(x(a), b)$ e o valor real $y(a)$, onde essa diferença é, por motivos claros, a medida chamada de **erro**.

Ambas tarefas de previsão fazem o uso dos passos de **treino** e depois **teste**, onde obviamente se pretende predizer os valores de novas instâncias, e não de instâncias antigas. Desta forma, testar o desempenho dos algoritmos com as mesmas instâncias utilizadas para construir os modelos preditivos não produz bons indicativos de desempenho. Testar algo com os mesmos parâmetros utilizados na aprendizagem é um tanto otimista e por isso não é uma boa forma de avaliar o seu desempenho (WITTEN; FRANK; HALL, 2011). Assim, é necessário um grupo de instâncias para o treino e criação do modelo e um segundo grupo de instâncias para o teste, dando origem aos independentes grupo de treino e grupo de teste.

Chamado de **holdout**, este método aloca, de forma aleatória, tipicamente dois terços da base de dados para o conjunto de treino e o restante, um terço, é utilizado para o teste, como é visto na Figura 5. No entanto, esse método é pessimista porque o modelo preditivo é determinado a partir de apenas uma porção das instâncias iniciais, que pode não ser representativo (HAN; KAMBER; PEI, 2012).

Figura 5 – Método *holdout*.

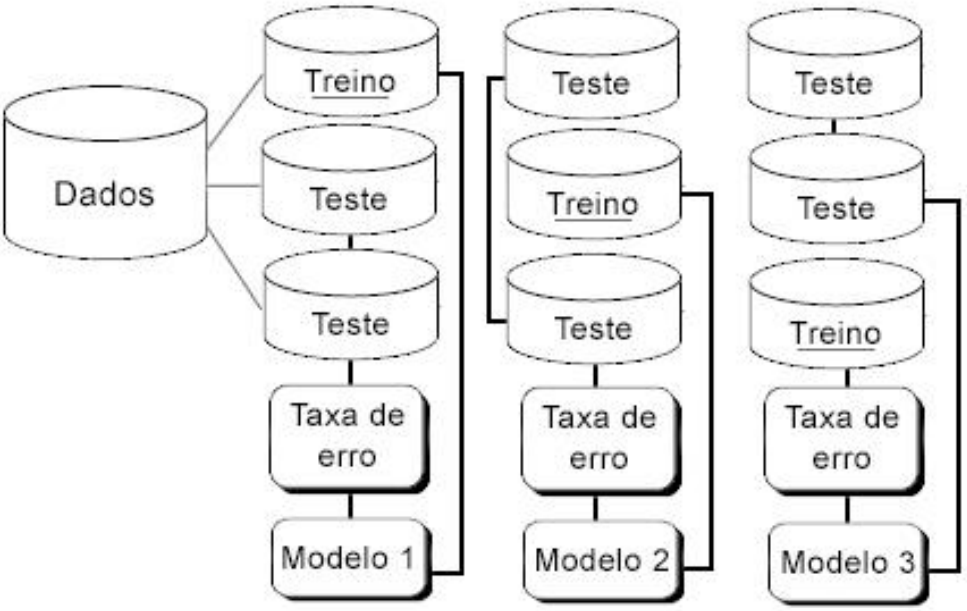


Fonte: Adaptado de (HAN; KAMBER; PEI, 2012).

De acordo com Witten, Frank e Hall (2011), para mitigar qualquer viés causado pelo treino com apenas uma porção de instâncias no método *holdout*, há uma simples variância que forma uma importante técnica de validação chamada **cross-validation**. Neste método, é decidido um número fixo de partições das instâncias. Supondo a escolha de x partições, as instâncias serão separados em x partes aproximadamente

iguais, onde cada parte será utilizada para teste e o restante será utilizado para treino, depois o processo será repetido até que todas as instâncias tenham sido usadas uma vez para o treino, gerando um modelo diferente para cada repetição. Após isso, a taxa de erro é calculada tirando a média da taxa de erro de cada modelo gerado. A Figura 6 mostra um exemplo com 3 partições.

Figura 6 – Método *cross-validation* com 3 partições.



Fonte: Adaptado de (HAN; KAMBER; PEI, 2012).

Leave-one-out é um caso especial de *cross-validation* onde o número de partições é igual ao número de instâncias na base de dados. Desta forma apenas uma instância é deixada de fora para teste, e o restante é usada para treino (HAN; KAMBER; PEI, 2012).

2.7 MEDIDAS DE CLASSIFICAÇÃO

Segundo Han, Kamber e Pei (2012), antes de falar das medidas, é preciso estabelecer quatro terminologias necessárias para computar as demais medidas, são elas:

a) True positives (TP): são as instâncias positivas que foram corretamente classificadas pelo classificador;
b) True negatives (TN): são as instâncias negativas que foram corretamente classificadas pelo classificador;
c) False positives (FP): são as instâncias negativas que foram incorretamente classificadas pelo classificador;
d) False negatives (FN): são as instâncias positivas que foram incorretamente classificadas pelo classificador.

Dado a quantidade de classes como m, uma **matriz de confusão** é uma tabela de tamanho m por m, assim uma entrada CM_{ij} nas primeiras m linhas e m colunas indica o número de instâncias da classe i que foram classificadas como classe j. Assim, um classificador com bom desempenho deve ter a maioria das suas instâncias representadas na diagonal principal da matriz, com o restante sendo zero ou o mais próximo possível de zero. Ou seja, FP e FN pertos de zero (HAN; KAMBER; PEI, 2012).

A **acurácia** de um classificador em um determinado teste é a porcentagem de instâncias do conjunto de teste que foram classificadas corretamente pelo classificador e definido pela Equação 3.

$$acuracia = \frac{TP + TN}{total\ de\ instâncias} \quad (3)$$

Em contra partida, a **taxa de erro**, também dada em porcentagem, de um classificador M pode ser medida pelo resultado de $1 - acuracia(M)$, que também pode ser calculado pela Equação 4:

$$taxa\ de\ erro = \frac{FP + FN}{total\ de\ instâncias} \tag{4}$$

Han, Kamber e Pei (2012) afirmam que há outras duas medidas amplamente utilizadas na classificação: **precisão** e ***recall***. A primeira utilizada como uma medida de exatidão, calculada pela Equação 5, já a medida *recall* significa integridade, calculada pela Equação 6:

$$precisao = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

Outra medida de classificação é a **estatística kappa**, utilizada para avaliar o nível de concordância entre os valores reais e os que foram previstos, segundo Vieira (2005) é calculada pela Equação 7.

$$kappa = \frac{Pa - Pe}{1 - Pe} \tag{7}$$

Onde Pa representa a quantidade de vezes que o valor previsto condiz com o real, e Pe representa a quantidade total de vezes em que deveria haver concordância. Neste caso um resultado próximo de zero significa baixa concordância e 1 significa total concordância.

2.8 MEDIDAS DE REGRESSÃO

Hand, Mannila e Smyth (2001) afirmam que, as funções para medir o **erro** de um algoritmo de regressão fazem a real diferença entre o valor previsto e o valor real, por se tratarem de valores numéricos. Então realizam a soma do quadrado dos erros para cada modelo gerado, desta forma com modelos de `a` até `N` têm-se a Equação 8:

$$erro = \sum_{a=1}^{N} (f(x(a); b) - (y(a)))^2 \tag{8}$$

Assim, na tarefa de regressão a medida pode representar ou não algum **erro**, porém quando apresenta, este pode vir de diferentes tamanhos, então são estabelecidas novas medidas para uma análise mais profunda do erro.

De acordo com Witten, Frank e Hall (2011), há medidas que podem ser usadas para avaliar o sucesso de uma previsão numérica. Sabendo que os valores previstos nas instâncias de teste são $p_1, p_2, \dots, p_n$, e que os reais valores são $a_1, a_2, \dots, a_n$, é possível calcular as medidas a seguir.

Root mean-squared error (RMSE): é a principal e mais comum medida, utiliza a raiz para manter a proporção do valor previsto. É dada pela Equação 9:

$$RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \cdots + (p_n - a_n)^2}{n}} \tag{9}$$

Mean absolute error (MAE): aqui apenas é tirado a média dos erros sem levar em consideração os sinais, tratando todos os tamanhos de erro de acordo com a sua magnitude. É dada pela Equação 10:

$$MAE = \frac{|p_1 - a_1| + \cdots + |p_n - a_n|}{n} \tag{10}$$

Coeficiente de correlação: também conhecida como coeficiente de Pearson, esta busca medir a correlação estatística entre os valores reais e os previstos. Indo de +1, para uma total correlação, até -1 para uma total correlação inversa, 0 significa que não há nenhuma correlação. Sabendo que $\bar{a}$ e $\bar{p}$ significam, respectivamente, o valor médio do conjunto de teste e valor médio do conjunto previsto, a equação do coeficiente de correlação é dada pela Equação 11:

$$CC = \frac{S_{PA}}{\sqrt{S_P S_A}}, onde\ S_{PA} = \frac{\sum_i (p_i - \bar{p})\,(a - \bar{a})}{n-1}, S_P = \frac{\sum_i (p_i - \bar{p})^2}{n-1}, S_A = \frac{\sum_i (a_i - \bar{a})^2}{n-1} \quad (11)$$

Após a avaliação de determinados algoritmos, o que tiver melhor desempenho será o que mostrará os menores valores para as medidas de erro e o maior valor de correlação (WITTEN; FRANK; HALL 2011).

2.9 CONCLUSÃO

A mineração de dados se destaca entre as etapas do processo de descoberta de conhecimentos. Necessitando de técnicas precisas, essa etapa do KDD resultará em um conjunto de informações prontas para serem analisadas e transformadas em conhecimento. Este capítulo apresentou uma revisão bibliográfica realizada sobre as diferentes categorias de aplicação utilizadas na mineração de dados, com ênfase na modelagem preditiva e em suas tarefas, detalhando alguns algoritmos, medidas e métodos de avaliação desse cenário.

No próximo capítulo será abordado o desenvolvimento da solução proposta, desde a sua explicação teórica até a sua disponibilização como uma extensão da ferramenta WEKA

3 UMA EXTENSÃO PARA AVALIAÇÃO DE CLASSIFICADORES EM BASES TEMPORAIS

Desenvolvida e mantida pela Universidade de Waikato, na Nova Zelândia, a ferramenta WEKA consiste em um software de uso livre e de código aberto, o que possibilita aos seus usuários desenvolverem funcionalidades extras, sob a forma de extensões, fazendo o reuso das implementações já desenvolvidas. Além disso, permite também a integração destas extensões à própria ferramenta por meio da geração de pacotes instaláveis, que podem ser facilmente compartilhados e adotados por outros usuários da WEKA (BOUCKAERT et al., 2014).

Os softwares de alta qualidade apresentam como característica importante componentes reutilizáveis, e o reúso destes componentes pode proporcionar benefícios (PRESSMAN, 2011), como é o caso da WEKA e sua possibilidade de extensão. Neste contexto, Sommerville (2011) conceitua o desenvolvimento de software baseado em reúso como uma estratégia da engenharia de software que reutiliza um software já existente e seus componentes para chegar ao resultado requerido, sendo este o modelo utilizado para o desenvolvimento da solução proposta.

Este capítulo aborda, na Seção 3.1, a especificação teórica do método de avaliação implementado. Na Seção 3.2, descreve a implementação da extensão para a ferramenta WEKA. A Seção 3.3 mostra uma visão geral do que foi implementado e

da integração da extensão, seguida pelo detalhamento, na Seção 3.4, de como se dá a disponibilização desta extensão através da criação de um pacote para a WEKA. Por fim, a Seção 4.4 apresenta uma conclusão para o capítulo.

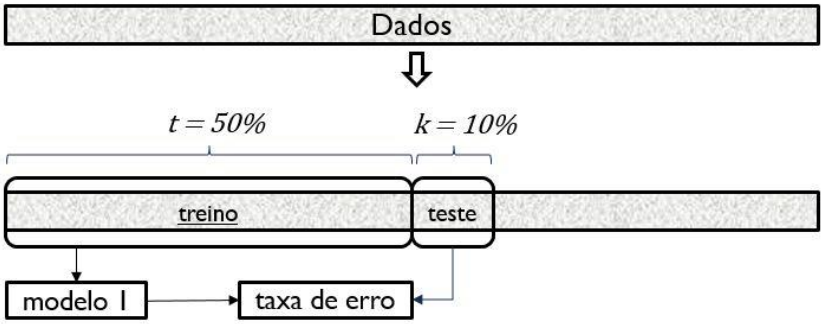
3.1 O MÉTODO DE AVALIAÇÃO IMPLEMENTADO COMO SOLUÇÃO

Os algoritmos do cenário de previsão, independentemente da tarefa, passam por dois passos básicos: a) criar o modelo preditivo baseado nos dados aprendidos; b) testar a acurácia do modelo através de uma avaliação que compara os resultados previstos e os valores reais. De acordo com Han, Kamber e Pei (2012), o ideal é a utilização de um determinado grupo de dados para o passo a) e depois um outro grupo, não utilizado para criação do modelo, para realização do passo b), e para isso há diferentes técnicas amplamente utilizadas, como a *houldout*, *cross-validation* e *leave-on-out*. No entanto, como já foi explicado, esses métodos não são adequados para previsão de variáveis em bases de dados com dependência cronológica.

O método implementado busca mitigar as possíveis inconsistências. Dessa forma, inicialmente são estabelecidos dois parâmetros de entrada: t e k, o primeiro significa o tamanho, em porcentagem, do conjunto de instâncias que será utilizado para geração do modelo preditivo, e o segundo, também em porcentagem, representa o tamanho do conjunto de instâncias que será utilizado para testar o modelo. Essas porcentagens são utilizadas para selecionar o início dos conjuntos de treino e teste, por isso a base precisa estar previamente organizada com as instâncias em ordem cronológica.

A Figura 7 mostra o Exemplo 1 da geração de um modelo preditivo do método implementado e de como ele é utilizado para calcular a taxa de erro do algoritmo preditivo, a partir dos conjuntos de teste. Para o Exemplo 1, foram utilizados 50% da base para treino e 10% para teste.

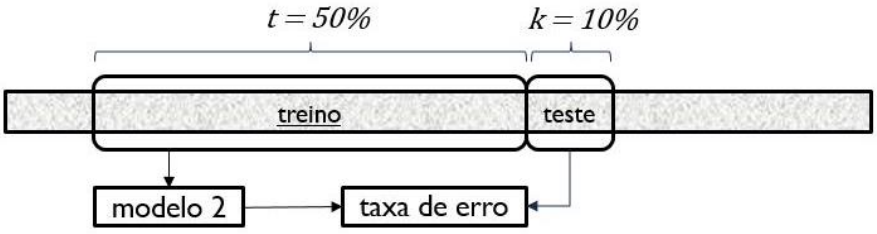
Figura 7 – Geração do modelo 1 com o método implementado de acordo com o Exemplo 1.



Fonte: Elaboração própria.

Após a geração do primeiro modelo, os conjuntos de treino e de teste serão deslocados, ou seja, das instâncias mais antigas para as mais atuais, selecionando a próxima k por cento das instâncias da base, conforme a Figura 8, gerando um segundo modelo e uma nova taxa de erro.

Figura 8 – Geração do modelo 2 com o método implementado de acordo com o Exemplo 1.



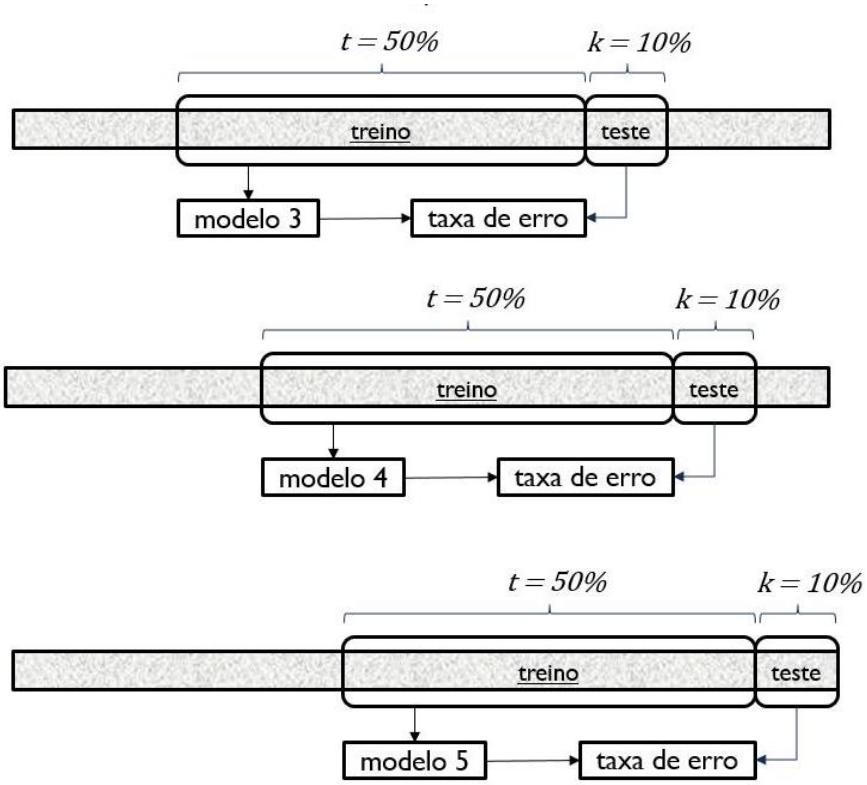
Fonte: Elaboração própria.

Esse processo será repetido até que o conjunto de teste chegue ao fim da base ou até que não reste instâncias suficientes para formar o tamanho do conjunto de teste, nesse caso as instâncias restantes não serão testadas. A Figura 9 demonstra a geração dos demais modelos, continuando com o Exemplo 1 de $t = 50$ e $k = 10$.

Semelhante aos demais métodos, a média da taxa de erro é calculada de acordo com a quantidade de modelos gerados. A quantidade de modelos gerados pode ser definido pela Equação 12:

$$quantidade\ de\ modelos = \frac{100 - t}{k}, onde\ quantidade\ de\ modelos\ \acute{e}\ um\ inteiro \quad (12)$$

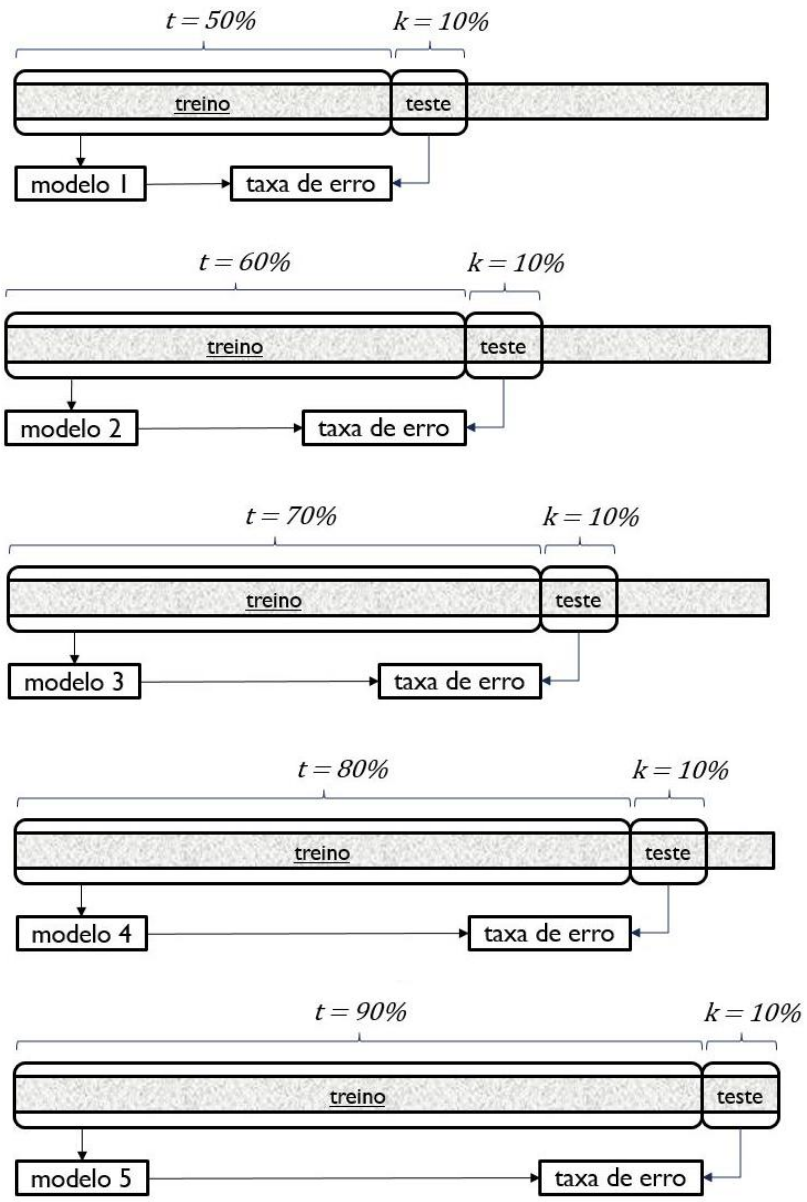
Figura 9 – Geração do modelo 3, 4 e 5 com o método implementado utilizando o Exemplo 1.



Fonte: Elaboração própria.

Há uma variância do método para os casos onde se deseja treinar com todas as instâncias anteriores ao grupo de teste. Nesse caso, na medida em que o conjunto de teste é realocado para as instâncias mais recentes, o grupo de treino engloba as instâncias previamente testadas para a construção do modelo, em outras palavras, o treino acontece de forma acumulada, fazendo a soma de $t + k$ em cada novo modelo gerado após o primeiro. A Figura 10 demonstra o mesmo Exemplo 1 para os valores das variáveis t e k, porém agora acumulando o conjunto de treino, onde o valor de t vai de 50 até 90.

Figura 10 – Geração dos modelos com o método implementado de acordo com o Exemplo 1, acumulando o grupo de treino.



Fonte: Elaboração própria.

O cálculo para a quantidade de modelos gerados permanece o mesmo para as duas variâncias do método, tendo em vista que o acúmulo, no treino, das instâncias já testadas não influencia na quantidade de instâncias a serem testadas até o fim da base.

Além das possíveis instâncias restantes que podem não serem suficientes para formar um novo grupo de testes de acordo com o tamanho de k. Nesse método, as instâncias utilizadas para a geração do primeiro modelo também serão testadas, já que o princípio do funcionamento do método é predizer os valores futuros de acordo com o conhecimento que se tem dos valores já conhecidos.

3.2 IMPLEMENTAÇÃO

A implementação da extensão seguiu o modelo de desenvolvimento baseado em reúso de software, foi necessário o uso da linguagem de programação Java, já que está é a linguagem utilizada no projeto da WEKA.

Uma das aplicações que a WEKA oferece é o *Explorer*, nesta estão presentes algumas das tarefas de mineração de dados e estão divididas em guias, de forma que cada uma apresenta seu próprio painel de configuração e utilização. As tarefas de classificação e regressão estão agrupadas em uma só, tendo em vista que utilizam os mesmos métodos de avaliação, diferenciadas apenas pela escolha dos algoritmos e pelas medidas resultantes. Apesar de estar inserida no cenário de previsão, a extensão implementou um método de avaliação diferente dos disponibilizados na WEKA.

A WEKA permite a extensão do *Explorer,* o que possibilita a adição de novas guias. Sabendo disso, foi decidido então que as funcionalidades e configurações da solução proposta seriam apresentadas em uma guia separada no *Explorer* da WEKA. Para isso, é necessário que a classe responsável pela nova guia seja uma subclasse de `javax.swing.Jpanel`, no intuito de garantir o seu comportamento como um painel da biblioteca *swing*, que é a biblioteca utilizada para a interface gráfica da WEKA, mantendo o padrão visual da ferramenta.

Além disso, a classe da nova guia também precisa implementar a interface `weka.gui.explorer.Explorer.ExplorerPanel`, que também serve para garantir que a classe faça a implementação dos métodos necessários para troca de mensagens com os outros componentes do *Explorer*.

Na Figura 11, pode ser visto a assinatura dos cinco métodos provenientes da interface *ExplorerPanel*, onde a WEKA faz chamada durante a sua execução para definir parâmetros como: o nome da guia e estabelecer/recuperar o valor do objeto referenciado ao próprio *Explorer* na classe desenvolvida. Dentre esses, o método mais significativo para o desenvolvimento da extensão é o *setInstances()*, responsável por informar e fornecer à classe o conjunto de instâncias carregadas pelo usuário na guia inicial do *Explorer*, para que se possa manipulá-la e realizar as tarefas necessárias.

Figura 11 – Assinatura dos métodos da interface *Explorer*.

```
/** Sets the Explorer to use as parent frame */
public void setExplorer(Explorer parent) { ..

/** returns the parent Explorer frame */
public Explorer getExplorer() { ..

/** Returns the title for the tab in the Explorer */
public String getTabTitle() { ..

/** Returns the tooltip for the tab in the Explorer */
public String getTabTitleToolTip() { ..

 * Tells the panel to use a new set of instances.
public void setInstances(Instances inst) {..
```

Fonte: Adaptado de WEKA (2017).

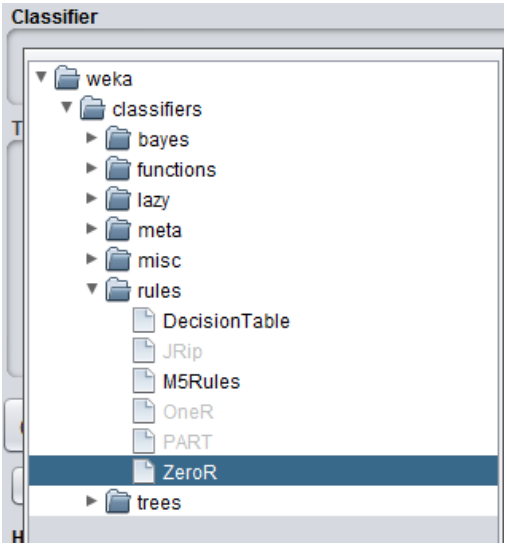
A interface `weka.gui.explorer.Explorer.LogHandler`, também foi implementada, oferecendo a possibilidade de gerenciar as mensagens do painel de *log* do *Explorer*, com a intenção de mostrar para o usuário quais as ações foras realizadas e informar o andamento ações. Essas informações permitem que o analista de dados acompanhe o funcionamento das demais funcionalidades nativas da ferramenta, melhorando a usabilidade da extensão.

Foi instanciado um objeto da classe *GenericObjectEditor* da WEKA, que possibilita a configuração dos algoritmos de previsão e que pode ser enviado como parâmetro à instância do objeto da classe *PropertyPanel*, um elemento gráfico, que

ao ser adicionado na interface gráfica, possibilita ao usuário a fácil escolha do algoritmo a ser avaliado (aqui chamado de *classifier*), por meio de uma estrutura de pastas como visto na Figura 12, junto com as opções únicas de configuração e de informação de cada algoritmo. Ao selecionar um algoritmo no componente gráfico, este é convertido para seu objeto referente da classe *Classifier*, que já fornece as funcionalidades referentes a um algoritmo de previsão, como construção do modelo preditivo, por exemplo.

Através do objeto da classe *Instances*, já visto na Figura 11, é possível ter acesso as classes da base de dados selecionada pelo usuário, ou seja, a variável que se pretende predizer o valor. É então preenchido uma lista em tempo de execução com as variáveis disponíveis para que o usuário selecione qual irá utilizar na previsão, junto com um *listener* para sempre atualizar a lista de algoritmos disponíveis no *PropertyPanel* de acordo com o tipo de variável selecionada pelo usuário, se estarão disponíveis algoritmos de classificação ou regressão.

Figura 12 – Componente gráfico fornecido pela WEKA para seleção do algoritmo.



Fonte: Adaptado de WEKA (2017).

No painel da extensão há três *inputs* para o usuário fornecer o valor das variáveis exclusivas e necessárias para o método implementado, sendo elas o tamanho do conjunto de treino, o tamanho do conjunto de teste e qual opção do método será utilizada: com ou sem acúmulo do conjunto de treino.

Após calcular a quantidade de modelos que serão gerados, a classe entra em um laço de repetição para cada modelo, onde em cada iteração as instâncias são repartidas nos grupos de teste e treino de acordo com o parâmetro estabelecido pelo usuário, e então através do método *buildClassifier()* da classe *Classifier* o algoritmo é treinado com o conjunto de treino, e é criado uma instância da classe *Evaluation*, responsável por realizar a avaliação do modelo preditivo por meio do método *evaluateModel()*. O método *evaluateModel()* recebe a instância do algoritmo já treinado e o grupo de teste para realizar a avaliação do modelo, como pode ser visto no trecho de código da Figura 13.

Figura 13 – Trecho do código do laço de repetição do método implementado.

```
SlideEvaluation slideEval = new SlideEvaluation(inst);

classifier.buildClassifier(trainInstances);
Evaluation eval = new Evaluation(testInstances);
eval.evaluateModel(classifier, testInstances);

if (isClassification){
    slideEval.accumlateClassifyMetrics(eval.confusionMatrix());
}else{
    slideEval.accumulateRegressionMetrics(testInstances, classifier, eval);
}
```

Fonte: Elaboração própria.

Dessa forma, o objeto da classe *Evaluation*, identificado na Figura 13 como *eval*, guarda as informações da taxa de erro do algoritmo em questão, para este modelo, e mais algumas das medidas calculadas no teste. Foi desenvolvida uma segunda classe para trabalhar com cada modelo gerado, identificada como *SlideEvaluation* ela é responsável por receber os modelos de cada iteração e acumular os seus resultados, assim como calcular as demais medidas e acumulá-las também. A classa *SlideEvaluation* trabalha de duas formas básicas, a primeira é para o cálculo das medidas de classificação, que é feito a partir da matriz de confusão fornecida pelo objeto *eval*, e a segunda é para o cálculo das medidas de regressão que, além do objeto *eval,* necessita também das instâncias testadas e do algoritmo utilizado. Após o cálculo de todas as medidas de cada modelo gerado, esta classe

fornece as suas respectivas médias, de acordo com as medidas referentes à tarefa em execução.

Além do resultado das medidas, a saída do método também apresenta um sumário especificando a quantidade de modelos de treino e teste, os parâmetros selecionados, a quantidade de instâncias totais testadas e treinadas, entre outras informações da avaliação. Essas saídas são salvas em um histórico acessível em tempo de execução.

3.3 VISÃO GERAL E INTEGRAÇÃO COM A WEKA

Ao utilizar essa extensão, o usuário da ferramenta WEKA continua tendo acesso a todas as outras funcionalidades. Os usuários já acostumados com a ferramenta apenas terão uma nova opção pra avaliação de algoritmos no cenário de previsão, indicada para uso em bases de dados com dependência cronológica entre as instâncias.

Além de fornecer essa alternativa, a extensão também preocupou-se com a integração com a WEKA e a usabilidade oferecida aos usuários, a fim de fornecer uma boa experiência tanto aos novos usuários da ferramenta para utilização desta extensão, quanto aos usuários que já fazem o seu uso.

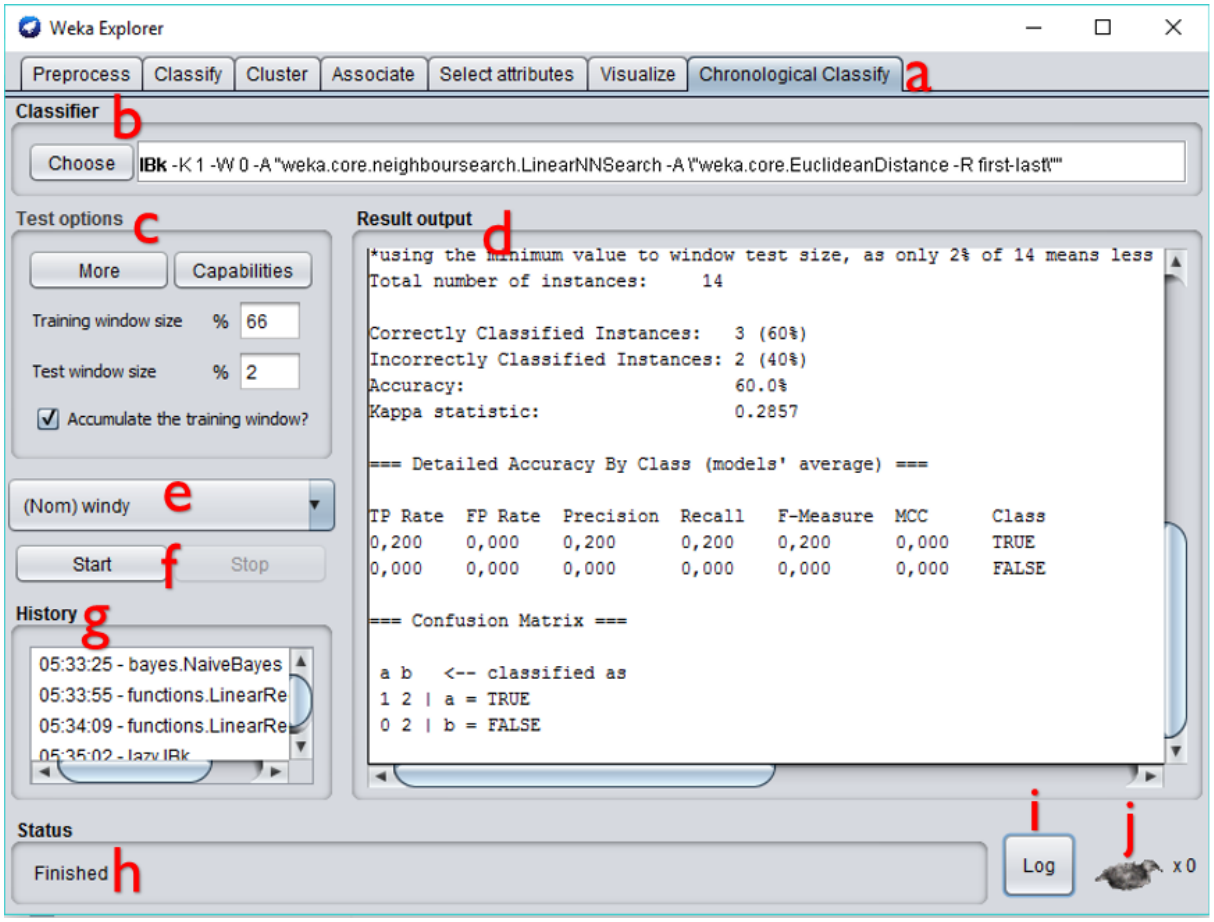
Para isso, buscou-se aproximar as interações feitas na ferramenta com a forma de utilização dos recursos nativos da WEKA. O painel da extensão foi inspirado no painel desenvolvido para as tarefas de previsão já existente na ferramenta, já que a solução proposta apresenta similaridades no seu funcionamento e na forma como o usuário deve interagir.

Na Figura 14, é apresentada a visão geral da extensão desenvolvida, onde pode ser visto o painel principal com as suas respectivas funcionalidades:

a) Aba de seleção do painel específico da extensão;

b) Opção para escolher, através de uma estrutura de pastas, o classificador a ser avaliado;
c) Opções do teste, para fornecimento dos parâmetros do método. Conta também com os botões *More* para mais informações sobre o método, e *Capabilities* para as capacidades e limitações do método.
d) Saída do sumário da avaliação, com informações dos números totais de instâncias, dos parâmetros utilizados e do classificador. Há também a saída dos resultados das medidas calculadas, de acordo com a tarefa em execução;
e) Lista de variáveis disponíveis na base para seleção da variável dependente, ou seja, para seleção da variável a ser prevista;
f) Botões para dar início ou interromper uma avaliação;
g) Lista com o histórico das avaliações anteriores, ocorridas na sessão em execução, identificadas pela hora de início da avaliação seguida pelo nome do classificador avaliado;
h) Barra de *status* para *feedback* sobre o andamento do processo em execução, além de informações sobre a quantidade de memória em uso pela WEKA;
i) Botão para abrir a janela dos *logs* universais da WEKA, incluindo histórico em tempo de execução das ações iniciadas neste e/ou em outros painéis;
j) Ícone para representar se a ferramenta está ou não executando algum processo em segundo plano, quando algum processo está em execução o ícone é animado. Há também um número ao lado direito do ícone que identifica a quantidade de processos solicitados pelo usuário que estão em andamento.

Figura 14 – Visão geral do painel da extensão desenvolvida.



Fonte: Elaboração própria.

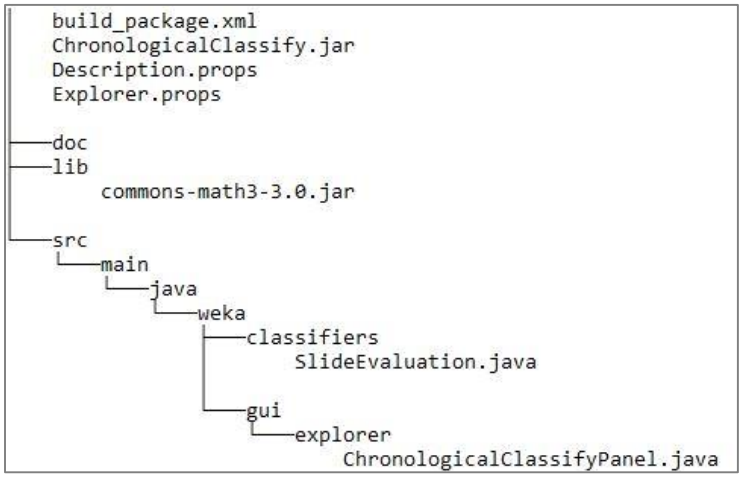
No exemplo utilizado na Figura 14, vemos o resultado da avaliação do algoritmo IBk ao predizer a variável *windy*, utilizando 66% das instâncias para treino e 2% para teste, e permitindo o acúmulo do conjunto de treino. No caso em questão, pela base analisada ter apenas 14 instâncias, foi utilizado o valor mínimo para o conjunto de teste (uma instância), conforme informado ao usuário no *output*. A variável dependente *windy* tem duas classes, *TRUE* e *FALSE*, assim foi disponibilizado ao usuário a matriz de confusão com as respectivas classes e uma análise detalhada das demais medidas por classe, além da média da acurácia e da estatística *kappa*.

3.4 DISPONIBILIZAÇÃO DA EXTENSÃO

Além da possibilidade de realizar extensões para a ferramenta WEKA, desde a versão 3.7.2 essas extensões podem ser compartilhadas e adicionadas a partir de um pacote. Na WEKA, um pacote é um conjunto de funcionalidades adicionais que podem ser disponibilizados pelos próprios usuários ou mantedores da ferramenta. Um pacote pode ser um arquivo comprimido no formato ZIP, que reúne todos os recursos necessários para seu funcionamento, tais como executáveis Java, documentação, arquivos de configuração, o código-fonte e outros.

A Figura 15 mostra como os arquivos da extensão desenvolvida nesse trabalho estão organizados de acordo com a estrutura definida de um pacote. Alguns subdiretórios e arquivos foram omitidos para proporcionar uma melhor compreensão da estrutura principal de um pacote.

Figura 15 – Estrutura do pacote da extensão.



Fonte: Elaboração própria.

As duas classes desenvolvidas na execução desse trabalho devem constar seus códigos fontes na pasta `src` do pacote, no mesmo endereço correspondente ao pacote Java no qual a classe foi implementada na WEKA, desta forma a classe `SlideEvaluation.java` está no pacote Java de endereço `weka.classifiers` do projeto da WEKA, e a classe desenvolvida para o painel principal da solução

`ChronologicalClassifyPanel.java` está no pacote Java correspondente ao endereço `weka.gui.explorer` do projeto da WEKA.

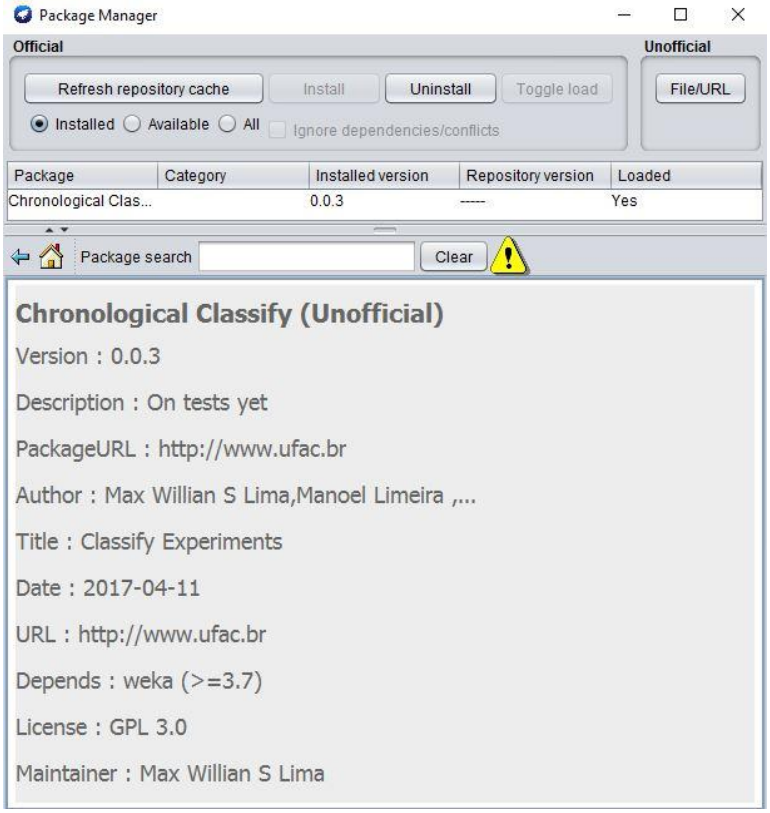
No diretório `lib` estão as bibliotecas de terceiros que foram utilizadas no desenvolvimento da extensão e que são disponibilizadas livremente por seus desenvolvedores. No caso extensão desenvolvida é possível ver nesta pasta o arquivo `commons-math3-3-0.jar` da biblioteca Java que foi utilizada na implementação para realizar o cálculo do coeficiente de correlação.

Opcionalmente, o desenvolvedor da extensão pode usar o diretório doc para disponibilizar a documentação do projeto. Apesar de opcional, representa uma boa prática, ainda mais por se tratar de um software de código aberto e que este código pode vir a ser utilizado por terceiros.

Outro arquivo importante é o `Explorer.props`, que pode ser visto na raiz da estrutura do pacote na Figura 15. Nesse arquivo deve ser definido a referência para a classe que implementa a nova guia criada, no caso a classe *ChronologicalClassifyPanel*. Caso essa referência não seja definida, a guia desenvolvida não estará disponível no *Explorer* da WEKA.

Para descrever as propriedades do pacote, como o nome da extensão, o nome do autor, as dependências, a versão da extensão e da sua licença e outras informações básicas, se faz necessário o uso do arquivo `Description.props`. Este arquivo é utilizado pelo gerenciador de pacotes da WEKA, mostrando as informações contidas nele para quem deseja instalá-lo, além de verificar a compatibilidade da versão descrita no arquivo com a versão do usuário que deseja instalar o pacote, informando-o em caso de incompatibilidade. A Figura 16 ilustra o gerenciador de pacotes da WEKA, onde foi selecionado o pacote da extensão proposta para visualizar as informações provenientes do arquivo `Description.props`.

Figura 16 – Gerenciador de pacotes da WEKA.



Fonte: Elaboração própria.

Para se tornar um pacote oficial da ferramenta e aparecer para os usuários a partir da pesquisa no gerenciador de pacotes da WEKA, o pacote deve ser enviado aos mantedores do projeto onde passará por testes de antivírus e *malware*, depois será analisado as suas funcionalidades, comportamento e contribuição.

Como alternativa, o pacote pode ser instalado diretamente dos arquivos do computador simplesmente fornecendo ao gerenciador da WEKA o arquivo ZIP do pacote. Chamado de Chronological Classify, o pacote foi disponibilizado, em sua versão 1.0, através da plataforma de compartilhamento de projetos de código aberto SourceForge[9].

[9] https://sourceforge.net/projects/chronological-classify-1-0/

3.5 CONCLUSÃO

Neste capítulo foi detalhado o desenvolvimento da extensão. Especificou-se o funcionamento do método implementado, assim como uma explicação mais técnica referente a sua implementação na linguagem Java fazendo o reúso de componentes do projeto WEKA. A extensão desenvolvida seguiu as especificações estabelecidas para realizar a integração, como heranças de classes Java e implementação de interfaces da WEKA. Também foi demonstrado uma visão geral da extensão e de suas funcionalidades. Ao fim, a extensão desenvolvida foi estruturada em um pacote WEKA, com código fonte e demais arquivos necessários, e disponibilizada em uma plataforma online para compartilhamento de projetos de código aberto.

O próximo capítulo destina-se a relatar a avaliação da extensão, a partir da descrição da base de dados de domínio específico utilizada, a análise das funcionalidades da extensão e da aceitação dos usuários.

4 AVALIAÇÃO DA EXTENSÃO

De acordo com Sommerville (2011), os requisitos de um software consistem em um conjunto de funcionalidades que deve ser fornecido pelo software. Esse conjunto pode ser dividido em: requisitos funcionais, que representam atividades que podem ser realizadas com o software, e os requisitos não funcionais, que estão relacionados com as propriedades do software, como por exemplo, a usabilidade. Ambos devem ser estabelecidos e registrados no documento de requisitos do software[10]. Está disponível, no Apêndice A, o documento de requisitos da extensão desenvolvida.

A fim de avaliar a implementação desses requisitos e a execução das funcionalidades na extensão, foram realizados experimentos utilizando uma base de dados meteorológicos da cidade de Rio Branco, no Acre. Isso possibilitou a análise dos requisitos funcionais da extensão na aplicação de um caso real do domínio específico da solução.

Além disso, foi realizado uma avaliação da extensão com usuários da ferramenta WEKA, onde estes usuários realizaram ambas tarefas de previsão com o auxílio da extensão e responderam um questionário sobre as suas experiências nesta utilização.

---

[10] Chamado de especificação de requisitos ou de documento de requisitos de software, este é um documento oficial que descreve tudo que deve ser implementado pelos desenvolvedores no desenvolvimento de determinado software (SOMMERVILLE, 2011).

Este capítulo está dividido em quatro seções, consistindo na apresentação da base de dados utilizada na Seção 4.1, nas duas etapas da avaliação nas Seções 4.2 e 4.3 e, por fim, uma conclusão referente aos resultados da avaliação da extensão na Seção 4.4.

4.1 BASE DE DADOS

Para a realização das avaliações, foi definida uma base de dados real que refletisse o cenário de aplicação da extensão proposta. Como a principal finalidade do método de avaliação desenvolvido é referente a bases de dados com dependência cronológica, procurou-se bases de dados onde as instâncias apresentassem essas características, além de um atributo específico que demarcasse o avanço cronológico.

Optou-se pela utilização de um contexto que apresentasse um simples nível de compreensão e demonstração sobre a previsão de atributos que denotam dependência cronológica, selecionando então o contexto de previsão do clima, mais precisamente da temperatura em relação ao passar das horas, dias ou meses de um ano.

O Instituto Nacional de Meteorologia (INMET) tem estações meteorológicas automáticas de observação de superfície em várias cidades do Brasil, segundo o INMET (2017), essas estações são compostas de uma unidade de memória central ligada a vários sensores de parâmetros meteorológicos, integrando estes valores minuto a minuto e os disponibilizando de forma automática a cada hora. Dentre os parâmetros observados estão: pressão atmosférica, temperatura e umidade relativa do ar, precipitação, radiação solar, direção e velocidade do vento, etc.

Foi escolhido um grupo de dados meteorológicos que contemplasse a cidade onde ocorreu o desenvolvimento deste trabalho, Rio Branco no Acre. O INMET permite, a partir de seu portal na internet, a recuperação dos dados coletados por

estas estações nos últimos 365 dias, porém, por uma limitação da data em que a estação da cidade de Rio Branco entrou em funcionamento, os dados disponibilizados pela mesma têm seu início em 26 de setembro de 2016, representando um período de 190 dias até a data de criação desta base de dados, 4 de abril de 2017.

A base apresenta os valores dos parâmetros coletados de acordo com o atributo hora, existindo 24 instâncias para cada dia coletado, com exceção de alguns dias que não apresentam o registro da coleta de alguma dessas horas. Além das horas ausentes, também foram identificadas 23 instâncias com inconsistências, que não apresentavam valores para os atributos meteorológicos. Então, na etapa de pré-processamento, essas instâncias foram removidas do conjunto de dados.

Na Tabela 1, pode ser visto um exemplo de algumas instâncias desta base e de alguns dos principais atributos referentes à elas. Os atributos Data e Hora são os responsáveis pela demarcação do avanço cronológico da base. Para alguns atributos, como a Temperatura, Umidade e Pressão, as instâncias apresentam o valor máximo e mínimo da hora coletada, e também o exato valor do momento do registro dessa instância.

Tabela 1 - Exemplo da base de dados com algumas instâncias e principais atributos.

DATA	HORA	TEMPERATURA (°C)			UMIDADE (%)			PRESSÃO (HPA)			RADIAÇÃO	CHUVA
	UTC	Inst	Máx	Mín	Inst	Máx	Mín	Inst	Máx	Mín	(kJ/m²)	(mm)
25/09/2016	15	23,6	23,7	23,4	85	89	57	991,1	991,1	991,0	624,9	0,0
25/09/2016	16	24,6	24,8	23,6	81	86	81	990,1	991,1	990,1	973,7	2,0
25/09/2016	17	25,4	25,7	24,6	77	83	75	989,2	990,2	989,2	1555,	0,0
25/09/2016	18	26,5	26,5	25,4	73	78	72	988,2	989,2	988,2	1564,	0,0
25/09/2016	19	28,3	28,4	26,5	63	73	63	987,0	988,2	987,0	2110,	0,0

Fonte: INMET (2017).

Na etapa de **transformação,** o atributo Data foi discretizado em quatro valores, ano, mês, nome do mês e dia referente àquela data. Também foi realizada a discretização do atributo Hora, gerando um novo atributo a fim de demarcar o período do dia referente àquela hora. Para isso, dividiu-se o valor total da quantidade de horas

de um dia em quatro períodos, a começar da hora 00:00, madrugada, manhã, tarde e noite.

O resultado final, após o pré-processamento e a transformação, foi uma base de dados com 4299 instâncias e 25 atributos, referentes as datas e os parâmetros meteorológicos, possibilitando agora o início dos procedimentos de mineração de dados para avaliação da solução proposta neste trabalho. Nos experimentos apenas os atributos meteorológicos e o *complete_data* foram utilizados como preditivos (variável dependente ou independente), o Quadro 1 descreve o nome e a descrição dos 25 atributos.

Quadro 1 – Descrição dos 25 atributos da base de dados.

Nome	**Preditivo**	**Descrição**
complete_data	Sim	Identificador da instância, representado pela concatenação da data e da hora do registro (DD/MM/AAAA HH:mm).
data	Não	Data do registro da instância (DD/MM/AAAA).
ano	Não	Ano do registro da instância.
mes_valor	Não	Valor do mês do registro da instância (0-12).
mes	Não	Nome do mês do registro da instância.
dia	Não	Valor do dia do registro da instância.
hora	Não	Hora do registro da instância (UTC).
periodo_dia	Sim	Período do dia do registro da instância (madrugada, manhã, tarde ou noite).
temp_inst	Sim	Temperatura no instante do registro da instância (C°).
temp_max	Sim	Temperatura máxima da hora de registro da instância (C°).
temp_min	Sim	Temperatura mínima da hora de registro da instância (C°).
umid_inst	Sim	Umidade no instante de registro da instância (%).
umid_max	Sim	Umidade máxima da hora de registro da instância (%).
umid_min	Sim	Umidade mínima da hora de registro da instância (%).
pto_orvalho_inst	Sim	Ponto de orvalho no instante do registro da instância (C°).
pto_orvalho_max	Sim	Ponto de orvalho máximo na hora do registro da instância (C°).
pto_orvalho_min	Sim	Ponto de orvalho mínimo na hora do registro da instância (C°).
pressão	Sim	Pressão no instante de registro da instância (hPa).
pressao_max	Sim	Pressão máxima na hora do registro da instância (hPa).
pressao_min	Sim	Pressão mínima na hora do registro da instância (hPa).
vento_direcao	Sim	Direção do vento (0°-360°).
vento_vel	Sim	Velocidade do vento (m/s).
vento_rajada	Sim	Rajada do vento (m/s).
radiação	Sim	Radiação $(kj/m)^2$.

Fonte: Adaptado de INMET (2017).

4.2 ANÁLISE DAS FUNCIONALIDADES

A avaliação das funcionalidades da solução proposta foi realizada seguindo os passos típicos de um usuário que pretende extrair conhecimentos a partir da modelagem preditiva dos dados.

A partir do painel de processamento do *Explorer* foi possível carregar a base de dados na WEKA, dessa forma já se pôde ter acesso à aba para utilização do painel principal da extensão. Como escolha para os experimentos, as tarefas de previsão serão realizadas com a variável, ou classe, *temp_inst* que representa, em graus célsius, a temperatura no instante da coleta dos dados.

A intenção foi predizer a temperatura de acordo com os últimos sete dias, para isso foi optado pelo atributo *complete_data* como marcador cronológico, pois este representa cada instância única em cada hora coletada. Para isso, ainda na aba de processamento da WEKA, foram removidos da base os demais atributos temporais, como *data* e *ano*.

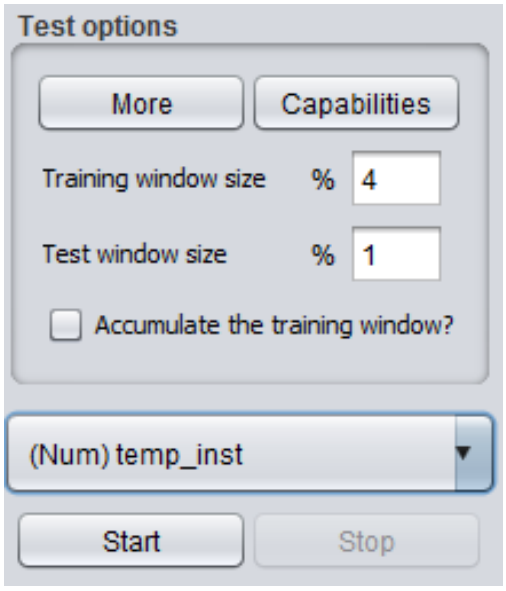
Deu-se início as configurações da avaliação na aba da extensão. Como padrão, a variável selecionada para previsão é sempre o último atributo da base de dados, então foi necessário selecionar manualmente a variável *temp_inst* para iniciar a avaliação, que é mostrada com seu nome precedido pelo indicador *(num)* no momento da seleção como pode ser visto na Figura 17, afirmando que esta é uma variável do tipo numérico. Com a variável selecionada, deve-se então escolher dentre os algoritmos disponíveis qual será avaliado, isso pôde ser feito a partir do botão de escolha do *classifier* que, ao ser acionado, abre uma estrutura de pastas onde os algoritmos estão organizados.

Devido a variável selecionada ser numérica os algoritmos exclusivos para classificação aparecem, na tela de seleção, como inativos, já que a tarefa de classificação não é capaz de trabalhar com variáveis que não sejam categóricas/nominais. Caso algum algoritmo de classificação seja selecionado ou seja

feita a troca da variável após a seleção do algoritmo, para uma em que o algoritmo não seja utilizável, o botão para iniciar a avaliação ficará desabilitado.

Para os parâmetros, decidiu-se treinar com 4% dos dados, o que representa aproximadamente uma semana na base de dados, e testar com 1% das instâncias seguintes, deixando a opção de acumular o conjunto de treino desativada. A Figura 17 demonstra os parâmetros fornecidos e a variável selecionada no painel da extensão.

Figura 17 – Parâmetro para realização dos testes da avaliação.



Fonte: Elaboração própria.

O teste ocorreu para dois algoritmos preditivos implementados na WEKA: M5P e IBk. O sumário das duas avaliações utilizando estes parâmetros pode ser visto no Quadro 2. Foram gerados 98 modelos preditivos, e o resultado do teste de cada modelo foi acumulado até que chegasse ao fim do teste onde foi calculado a média das medidas referentes a cada modelo. Das 4299 instâncias, a extensão não testou apenas as 171 primeiras, por serem utilizadas para o primeiro conjunto de treino, e as últimas 12 instâncias, por não serem o suficiente para formarem os 1% necessários para formar o conjunto de teste.

Quadro 2 – Sumário da avaliação.

Modelos	98
Acumular treino	Não
Tamanho do grupo de treino	171 (4%)
Total de instâncias usadas para treino	4245
Tamanho do grupo de teste	42 (1%)
Total de instâncias usadas para teste	4116
Total de instâncias não testadas	183 (Primeiras 171 e últimas 12)
Total de instâncias	4299

Fonte: Elaboração própria.

Todos os algoritmos foram configurados com suas opções padrões de acordo com a WEKA. A Tabela 2 apresenta os resultados das principais medidas de regressão de acordo com a avaliação de tais algoritmos.

Tabela 2 – Resultado da avaliação para algoritmos de regressão.

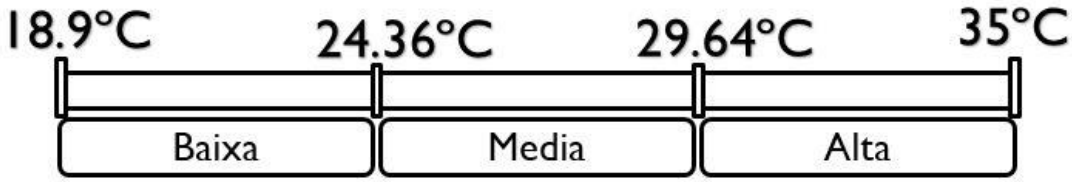
MEDIDAS	**ALGORITMOS**	
	M5P	**IBk**
Correlação	0,9544	0,3854
RMSE	0,85	2,88
NRMSE	0,66	0,723
MAE	0,6349	2,4202

Fonte: Elaboração própria.

Apesar de não ser o objetivo desta análise, é interessante observar o desempenho do algoritmo M5P através dos seus resultados, onde este teve uma maior correlação e menores medidas de erro quando comparado ao IBk.

Para realizar a tarefa de classificação a base precisou passar novamente pela etapa de transformação a fim de discretizar os valores da variável *temp_inst*, tendo em vista que esta tarefa não é capaz de lidar com variáveis numéricas como variável dependente. Decidiu-se discretizar a temperatura nas categorias Baixa, Média e Alta, para isso foi identificado o maior e o menor valor da temperatura na base, então o intervalo entre esses dois valores foi dividido em 3 frações. Cada fração correspondeu as respectivas categorias em que se pretendeu discretizar a temperatura: Baixa, Média e Alta. A Figura 18 ilustra como ficou a discretização da variável dependente.

Figura 18 – Discretização da variável *temp_inst*.



Fonte: Elaboração própria.

Os valores antigos da variável *temp_isnt* foram então substituídos por suas respectivas classes, deixando a base preparada para a tarefa de classificação. Da mesma forma como a tarefa de regressão, a base foi carregada na WEKA e, então, já se pôde selecionar a classe *temp_isnt,* que agora aparece precedida do indicador *(num)*, e selecionar os algoritmos de classificação disponíveis.

O teste foi realizado com os algoritmos NaiveBayes, J48 e o IBk, já que esse último pode ser usado em ambas tarefas. Utilizando os mesmos parâmetros da avaliação anterior, o sumário se deu da mesma forma como visto no Quadro 2. A Tabela 3 apresenta os resultados da acurácia e da estatística *kappa* dos algoritmos avaliados.

Tabela 3 – Resultado da avaliação com algoritmos de classificação.

MEDIDAS	**ALGORITMOS**		
	NaiveBayes	**J48**	**IBk**
Acurácia	90.11%	92,03%	92,97%
Estatística *kappa*	0,8247	0,858	0,8747

Fonte: Elaboração própria.

Os três algoritmos selecionados para essa tarefa utilizaram as configurações padrões de acordo com a ferramenta WEKA. Além das medidas geradas, a tarefa de classificação também tem como resultado a matriz de confusão, usada para detalhar quantas instâncias o algoritmo identificou corretamente como sendo de determinada classe e a qual classe este determinou de forma incorreta as demais instâncias.

Na Figura 19, estão as matrizes de confusão geradas pela extensão como resultado de cada teste. É perceptível que o algoritmo IBk, que obteve maior acurácia entre os três, concentrou suas predições na diagonal principal da matriz, ou seja, classificou o maior número de instâncias de acordo com sua real classe. Outro fator interessante é que os erros do IBk e do NaiveBayes foram de no máximo uma classe de distância da classe real. Por exemplo, o BaiveBayes classificou 88 instâncias de classe Alta (*c*) como sendo Média (*b*), porém nenhuma como Baixa (*a*), da mesma forma que classificou 63 instâncias de classe *a* como classe *b*, mas nenhuma de classe *a* como *c*.

Figura 19 – Matrizes de confusão da classificação.

NaiveBayes			
a	b	c	classificado como
2071	153	0	a = Baixa
63	1384	88	b = Media
0	103	254	c = Alta

J48			
a	b	c	classificado como
2141	82	1	a = Baixa
77	1376	82	b = Media
10	76	271	c = Alta

Ibk			
a	b	c	classificado como
2148	76	0	a = Baixa
60	1411	64	b = Media
0	89	268	c = Alta

Fonte: Elaboração própria.

Para a tarefa de classificação o IBk destacou-se com uma acurácia de quase 93% de acerto médio ao tentar predizer a classe de 4116 instâncias, sendo que as 289 instâncias incorretas foram classificadas como apenas uma classe acima ou abaixo da classe real, como observado na sua matriz de confusão. Essas medidas talvez possam ser melhoradas com um estudo mais detalhado sobre a organização da base para escolha dos melhores parâmetros para a avaliação, identificando os padrões temporais e os utilizando para formar os grupos de treino e testes.

Aos usuários que pretendem avaliar algoritmos para uso em bases com dependência cronológica, a extensão implementou todas as funcionalidades necessárias para a realização das tarefas de previsão, fornecendo ainda a disponibilização das medidas resultantes para análise, a customização dos parâmetros de entrada e a escolha do algoritmo.

4.3 ACEITAÇÃO DOS USUÁRIOS

A fim de verificar a usabilidade requerida para a solução, foi acordado com o professor da disciplina de Banco de Dados II do segundo semestre letivo de 2016, do curso de Sistemas de Informação da Universidade Federal do Acre, a utilização de uma aula para avaliar junto a turma a solução proposta neste trabalho.

Os 20 alunos que participaram da avaliação já haviam tido contato com a WEKA na disciplina e vinham utilizando a ferramenta apenas com seus recursos nativos para realização dos trabalhos acadêmicos, os caracterizando assim como potenciais avaliadores da extensão. A avaliação ocorreu no laboratório de informática, supervisionada pelo professor da disciplina, onde foi ministrado uma apresentação sobre o contexto da modelagem preditiva. Os avaliadores receberam o pacote da extensão desenvolvida e a base utilizada na avaliação da tarefa de regressão e a sua versão transformada para a tarefa de classificação, com a variável alvo discretizada.

Primeiramente foram repetidos os mesmos experimentos realizados na análise das funcionalidades (na Seção 4.2), depois os usuários puderam tomar suas ações livremente e explorar a extensão por um determinado tempo. Ao fim dos experimentos, foi aplicado um questionário aos 20 avaliadores.

O questionário era composto de 10 afirmações, referentes à solução, onde os avaliadores deveriam informar o seu nível de concordância com estas afirmações, seguindo a escala *Likert*[12], de acordo com as suas experiências na utilização da extensão. A Tabela 4 descreve as afirmações do questionário aplicado e o resultado em porcentagem da avaliação proveniente dos 20 avaliadores, onde foram agrupadas como positivas as respostas concordo e concordo totalmente, e agrupadas como negativas as respostas discordo e discordo totalmente.

[12] A escala *Likert* é caracterizada pela utilização de expressões verbais que vão desde a máxima discordância até a máxima concordância com determinada afirmação (BECKER, 2015).

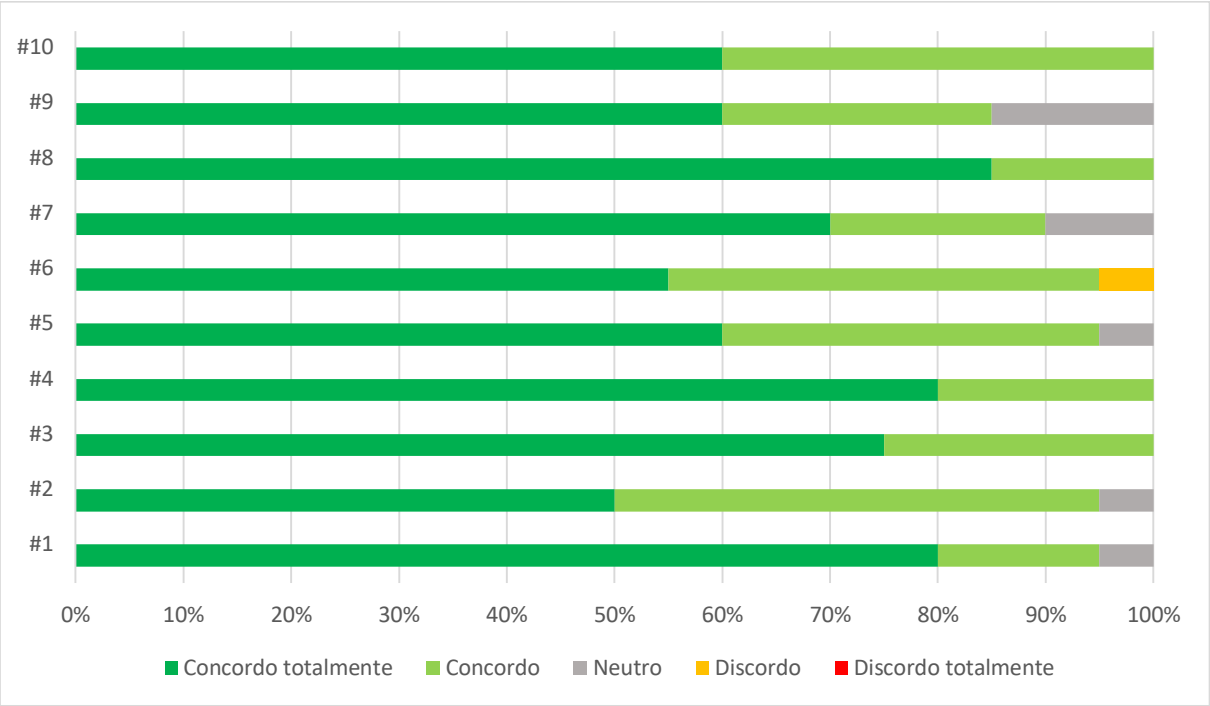
Tabela 4 – Resultado detalhado da avaliação dos 20 participantes.

#	AFIRMAÇÃO	AVALIAÇÃO (%)		
		Positiva	**Neutra**	**Negativa**
1	A integração da extensão com a ferramenta WEKA é satisfatória.	95	5	0
2	A solução fornece informações suficientes para avaliação de tarefas de previsão (regressão e classificação).	95	5	0
3	Os dados são apresentados em um formato útil que permite fácil compreensão e localização das informações na tela.	100	0	0
4	Os resultados são apresentados com exatidão.	100	0	0
5	Os nomes que identificam as opções na interface facilitam a compreensão do que cada ação executará.	95	5	0
6	A interface fornece feedback dos comandos acionados e processos em andamento.	95	0	5
7	O tempo de reposta dos comandos é satisfatório.	90	10	0
8	A terminologia utilizada está de acordo com as demais funcionalidades da WEKA e com as tarefas de classificação/regressão.	100	0	0
9	O propósito das variáveis de teste (tamanho do treino, teste e acumulação ou não do treino) é de fácil compreensão.	85	15	0
10	De maneira geral, a metodologia de avaliação proposta soluciona bem a limitação com bases de dependência cronológica.	100	0	0
	MÉDIA	**95,5**	**4**	**0,5**

Fonte: Elaboração própria.

A média de respostas positivas foi de 95,5%, onde 4 das 10 afirmações receberam 100% de concordância, e apenas uma afirmação obteve positividade abaixo de 90%. Com exceção da afirmação 6, nenhuma mais obteve avaliação negativa. A Figura 20 mostra um resumo comparativo da opinião dos avaliadores em relação as 10 afirmações. Os gráficos com as informações detalhadas da concordância dos usuários sobre as 10 afirmações então disponíveis no Apêndice B.

Figura 20 – Resumo comparativo da opinião dos avaliadores.



Fonte: Elaboração própria.

As quatro afirmações que receberam unanimidade positiva se referem a como os dados são apresentados ao usuário, sobre a familiaridade da extensão com a WEKA e sobre o propósito geral da extensão em oferecer uma solução ao problema apresentado. Já a afirmação com menor concordância, sendo 85%, se refere ao entendimento de maneira fácil dos usuários sobre o papel dos parâmetros utilizados para a avaliação, porém esta não recebeu nenhuma avaliação negativa, sendo os 15% restantes representados como neutralidade.

A única afirmação que recebeu algum tipo de discordância foi sobre o fornecimento de *feedback* sobre os comandos acionados e processos em andamento na extensão, sendo essa discordância partindo de apenas um avaliador, os outros 19 concordaram em dizer que a extensão fornece esse *feedback*.

Além das 10 afirmações para os usuários avaliarem se estão de acordo ou não, o questionário também continha uma questão opcional: *Você tem alguma opinião, sugestão ou crítica a respeito da sua experiência na utilização da extensão?* Dos 20

avaliadores, quatro deixaram suas respostas, elas estão descritas no Quadro 3. As recomendações foram aceitas e implementadas para a versão final da extensão.

Quadro 3 – Opiniões livres e opcionais deixadas pelos avaliadores da solução.

#Questionário	Opinião
12	A ferramenta parece seguir bem o padrão weka de apresentação da informação.
17	Achei que ficou muito bom. Faltou apenas algumas *tooltips* nas opções de parâmetro.
19	Na opção "Accumulate the training window?" Eu tiraria o ponto de interrogação.
20	Só tenho elogios. Gostei muito.

Fonte: Elaboração própria.

Em geral, as opiniões deixadas caracterizam uma boa experiência por parte dos avaliadores. Com menos de 1% na média de avaliações negativas e somente 4% de neutralidade, é possível considerar que a solução proposta teve um grande aceite por parte dos usuários da ferramenta WEKA.

4.4 CONCLUSÃO

Este capítulo teve como objetivo relatar as atividades realizadas para avaliação da solução proposta, tendo início pela demonstração e demonstração da base de dados, e sua passagem pelas etapas nas etapas do processo de descoberta de conhecimento, com foco na mineração de dados realizada para analisar as funcionalidades da extensão desenvolvida. Também foi avaliado o nível de aceitação da extensão por parte de usuários da WEKA.

A extensão cumpriu os requisitos estabelecidos, funcionais e não-funcionais, possibilitando a realização de todas as funcionalidades requeridas referentes as tarefas de previsão, de forma integrada à WEKA e seguindo o padrão de apresentação das funcionalidades nativas da ferramenta. Na avaliação, obteve-se uma média de 95,5% de resultados positivos entre os 20 usuários entrevistados, que no papel de

analista de dados informaram se concordavam ou não com dez afirmações. As afirmações vão desde especificações técnicas do propósito da extensão até a forma como a interface fornece os resultados e o *feedback* das ações. Com o cumprimento do propósito da extensão em ambos aspectos funcionais e não funcionais da implementação, e com o excelente resultado de aceitação de usuários já familiarizados com a WEKA, a extensão desenvolvida teve seus objetivos alcançados.

A seguir, o último capítulo reservou-se a falar das considerações finais em relação a solução proposta e fazer as recomendações de possíveis trabalhos futuros, a fim de complementar as contribuições feitas neste trabalho.

5 CONSIDERAÇÕES FINAIS E RECOMENDAÇÕES

Esse trabalho de pesquisa relatou o desenvolvimento de uma extensão para a ferramenta WEKA a fim de solucionar um problema específico da área de mineração de dados. Como quinto e último, esse capítulo conclui a pesquisa com as suas considerações finais na Seção 5.1 e, na Seção 5.2, uma lista de recomendações para trabalhos futuros.

5.1 CONSIDERAÇÕES FINAIS

O problema em avaliar algoritmos preditivos aplicando-os em bases com dependência cronológica decorre devido ao funcionamento dos métodos tradicionais. Desta forma, analistas de dados que utilizam a ferramenta WEKA, que implementa os métodos tradicionais, não podem avaliar tais algoritmos de forma coerente simplesmente utilizando tais métodos.

Esse trabalho de conclusão de curso realizou o desenvolvimento de uma extensão que possibilita a avaliação de algoritmos preditivos em bases de dados onde suas instâncias são apresentadas de maneira cronologicamente dependente. A implementação levou em conta o aspecto das bases de dados e possibilitou uma avaliação coerente, onde os algoritmos são sempre treinados com instâncias do

passado e testadas com instâncias futuras. A extensão possibilita a avaliação com todos os algoritmos preditivos implementados na WEKA, e fornece ao analista de dados as principais medidas provenientes das avaliações.

Para a implementação fez-se necessário um estudo bibliográfico referente aos principais conceitos de mineração de dados, assim como a elaboração do documento de requisitos. A implementação também se deu de acordo com um desenvolvimento baseado em reúso de software, utilizando as implementações do WEKA como base para solução e integração da extensão. Ao fim do desenvolvimento foi gerado um pacote instalável para a ferramenta WEKA, avaliado e então disponibilizado.

O trabalho de pesquisa cumpriu o seu objetivo geral, através da realização de todos os objetivos específicos. Houve dificuldade para encontrar um grupo grande de usuários experientes com as funcionalidades da ferramenta voltadas ao cenário de modelagem preditiva, e que estivessem disponíveis para realizar a avaliação. Porém, quase ao final da pesquisa foi possível realizar a avaliação com um grupo de 20 alunos, que, apesar da baixa experiência, já haviam utilizado a ferramenta e foram suficientes para avaliar os quesitos de integração e usabilidade da extensão desenvolvida, resultando em 95,5% de aceitação referente ao questionário aplicado.

Desta forma, os resultados alcançados foram satisfatórios e possibilitaram a solução do problema desta pesquisa, fornecendo uma extensão a ferramenta WEKA que possibilite os seus usuários avaliarem, de forma coesa, algoritmos preditivos em bases de dados com dependência cronológica.

5.2 RECOMENDAÇÕES

Apesar desse trabalho ter seus objetivos alcançados e ser concluído de forma satisfatória, é possível que sejam implementados algumas recomendações em

trabalhos futuros, a partir do código fonte da extensão disponibilizado junto ao pacote. Para tal fim, é recomendado:

a) Identificar na literatura demais medidas de regressão e classificação e implementá-las na extensão;
b) Desenvolver uma documentação detalhada e precisa sobre a implementação da extensão e melhorar a sua disponibilização.
c) Implementar o método de avaliação para algoritmos preditivos também para o módulo *Experimenter* da WEKA;
d) Implementar à extensão outros métodos de avaliação de algoritmos preditivos.

REFERÊNCIAS

AHA, David W.; KIBLER, Dennis; ALBERT, Marc K. **Instance-based learning algorithms**. Machine learning, v. 6, n. 1, p. 37-66, 1991.

BECKER, J. L. **Estatística Básica:** Transformando Dados em Informação. Porto Alegre: Bookman Editora, 2015.

BREIMAN, Leo. **Random forests**. Machine learning, v. 45, n. 1, p. 5-32, 2001.

BOUCKAERT, R. R.; FRANK, E.; HALL, M.; KIRKBY, R.; REUTEMANN, P.; SEEWALD, A.; SCUSE, D. **WEKA Manual for Version 3-7-12**. Hamilton: University of Waikato, 2014.

CHEN, Ming-Syan; HAN, Jiawei; YU, Philip S. **Data mining:** an overview from a database perspective. IEEE Transactions on Knowledge and data Engineering, v. 8, n. 6, p. 866-883, 1996.

DE LIMA JÚNIOR, Manoel Limeira et al. Developers assignment for analyzing pull requests. In: **Proceedings of the 30th Annual ACM Symposium on Applied Computing**. ACM, 2015. p. 1567-1572.

Piatetsky-Shapiro, Gregory. **Advances in knowledge discovery and data mining**. Eds. Usama M. Fayyad, Padhraic Smyth, and Ramasamy Uthurusamy. Vol. 21. Menlo Park: AAAI press, 1996.

GEORGE, H. John; PAT, Langley. **Estimating Continuous Distributions in Bayesian Classifiers**. In: Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, 338-345, 1995.

HAN, Jiawei; KAMBER, Micheline; PEI, Jian. **Data mining:** concepts and techniques. Elsevier, 2012.

HAND, David J.; MANNILA, Heikki; SMYTH, Padhraic. **Principles of data mining**. MIT press, 2001.

INMET. **Estações Automáticas**. Disponível em: <http://www.inmet.gov.br/portal/index.php?r=estacoes/estacoesAutomaticas>. Acesso em: 4 Abr. 2017.

MORAIS, Edison Andrade Martins; AMBRÓSIO, Ana Paula L. **Mineração de textos**. Relatório Técnico–Instituto de Informática (UFG), 2007.

PATIL, Tina R.; SHEREKAR, S. S. **Performance analysis of Naive Bayes and J48 classification algorithm for data classification**. International Journal of Computer Science and Applications, v. 6, n. 2, p. 256-261, 2013.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 7. ed. Porto Alegre: McGraw-Hill, 2011.

QUINLAN, J. R. **C4. 5**: Programs for Empirical Learning Morgan Kaufmann. San Francisco, CA, 1993.

QUINLAN, John R. et al. **Learning with continuous classes**. In: 5th Australian joint conference on artificial intelligence. 343-348, 1992.

REXER, K. **2013 Data Mining Survey:** Highlights. In: PREDICTIVE ANALYTICS WORLD. Boston, 30 set. 2013.

SZWARCFITER, Jayme Luiz; MARKENZON, Lilian. **Estruturas de Dados e seus Algoritmos**. Livros Técnicos e Científicos, 1994.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TAN, P.; STEINBACH, M.; KUMAR, V. **Introduction to data mining**. Boston: Pearson Addison Wesley, 2006.

VIERA, Anthony J. et al. **Understanding interobserver agreement**: the kappa statistic. Fam Med, v. 37, n. 5, p. 360-363, 2005.

WAZLAWICK, Raul. **Metodologia de pesquisa para ciência da computação**. 2a edição. Elsevier Brasil, 2014.

WEKA. **Data Mining Software in Java**. Disponível em: < http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em: 4 Abr. 2017.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining:** Practical machine learning tools and techniques. Morgan Kaufmann. 3ª edição, 2011.

WIVES, Leandro Krug. **Tecnologias de descoberta de conhecimento em textos aplicadas à inteligência competitiva**. Exame de Qualificação EQ-069, PPGC-UFRGS, 2002.

APÊNDICES

APÊNDICE A – DOCUMENTO DE REQUISITOS

Documento de Requisitos de Software

Chronological Classify Weka Extension

Versão 1.0

Desenvolvedores/Analistas

Max Willian Soares Lima

Rio Branco – AC

2017

1. Análise do Problema

No cenário de modelagem preditiva, o analista de dados conta com variados algoritmos desenvolvidos para predizer o valor de uma variável específica de uma determinada instância. Os algoritmos geram um modelo preditivo treinando com um conjunto de instâncias da base, e assim testam o modelo tentando predizer o valor de uma variável específica das demais instâncias de um conjunto separado para teste.

Considerando que esses algoritmos utilizam diferentes técnicas para gerar o seu modelo preditivo, suas previsões são avaliadas e os resultados são comparados para que se verifique qual algoritmo tem melhores capacidades preditivas para o domínio em que se pretende aplica-lo. Para isso, a ferramenta WEKA implementa métodos tradicionais de avaliação, como o holdout e o cross-validation, que dividem a base de dados de acordo com parâmetros de entrada. Esses parâmetros definem como as instâncias da base irão formar os conjuntos de treino e teste. O holdout, no entanto, determina o modelo preditivo treinando com apenas uma porção dos dados iniciais, podendo não ser representativo. Já o cross-validation realiza múltiplos testes até treinar com toda a base, fazendo deste um dos métodos mais aceitos e amplamente utilizado.

Há bases de dados em que suas instâncias são organizadas sequencialmente, de forma cronológica, onde os períodos temporais de registro dessas instâncias influenciam no valor de uma determinada variável, como por exemplo uma temperatura coletada em determinada época do ano que apresenta um valor mas que pode apresentar outro valor completamente diferente no ano seguinte. Para esses casos, o cross-validation apresenta uma inconsistência, tendo em vista que em determinado momento este irá treinar os algoritmos com instâncias futuras e depois testar as suas capacidades preditivas em instâncias passadas. Seria como tentar predizer o valor do salário mínimo brasileiro nos tempos da ditadura com base nas informações de instâncias atuais, por exemplo.

2. Necessidades Básicas do Cliente

A possibilidade de usuários da ferramenta WEKA validarem algoritmos de classificação mesmo em bases de dados com dependência cronológica, sem a necessidade de recorrer a métodos externos.

Sendo assim, o software especificado neste documento se dá como uma extensão da ferramenta WEKA, permitindo sua integração e, então, utilização a partir da interface já conhecida. O software deverá possibilitar um novo método de avaliação que satisfaça a limitação exposta na análise do problema.

3. Estudo de Viabilidade

3.1. Viabilidade Técnica

A ferramenta WEKA possibilita sua extensão e dá todo o suporte necessário ao desenvolvimento de plug-ins, assim como documentação e exemplos de implementação. A linguagem de programação especificada para este uso é a linguagem Java, que também apresenta boa documentação e é bem difundida no cenário de desenvolvimento, assim como apresenta grande familiaridade de uso por parte do desenvolvedor deste software. Essas duas parcelas justificam a viabilidade técnica para o desenvolvimento especificado neste documento.

3.2. Viabilidade Econômica

As ferramentas adotadas para o desenvolvimento são livres, assim não necessitam de nenhum gasto adicional, assim como não será necessário investimento em nenhum outro equipamento. Além disso, este é um projeto acadêmico e não incorre em nenhum gasto adicional, apresentando sua viabilidade econômica.

3.3. Viabilidade Legal

A solução proposta não possui qualquer impedimento legal, pois não está em desacordo com nenhuma lei, seja na esfera municipal, estadual ou federal. Devido ao WEKA está licenciado sob a GPL (General Public License), o software desenvolvido também deverá utilizar o mesmo licenciamento, bem como atenderá a exigência da GPL de disponibilização do código-fonte quando for distribuído publicamente.

4. Missão do Software

Possibilitar ao usuário da ferramenta WEKA uma maior abrangência e possibilidade de avaliação dos algoritmos de classificação disponíveis nesta.

5. Limites do Sistema

ID	Funcionalidade	Justificativa
L1	Realizar pré-processamento de dados.	Não faz parte dos objetivos do software substituir funcionalidades já atendidas pela ferramenta WEKA.

6. Benefícios Gerais

ID	Benefício
B1	Maior abrangência de bases de dados.
B2	Mais possibilidades de avaliação de algoritmos.
B3	Integração com a ferramenta WEKA.

7. Restrições

ID	Restrição	Descrição
R1	Compatibilidade	O software não será compatível com versões da ferramenta WEKA que não suportam o uso de pacotes para adicionar novas funcionalidades.

8. Atores

ID	Atores	Descrição
A1	Minerador de dados	Usuário que já utiliza a ferramenta WEKA para extrair padrões na forma de regras de associação.

9. Requisitos Funcionais

ID	Funcionalidade	Necessidades	Prioridade
RF1	Implementar uma avaliação para bases do domínio de aplicação do problema	Implementar um método que realize avaliação levanto em conta a dependência cronológicas das instâncias da base	Essencial
RF2	Integrar a implementação a ferramenta WEKA	Possibilitar a utilização da implementação através da	Importante

		ferramenta WEKA, como um painel no Explorer	
RF3	Seleção de algoritmos	Possibilitar a escolha do algoritmo entre os disponíveis na WEKA	Essencial
RF4	Entrada de parâmetros	Possibilitar a entrada dos parâmetros necessários para a avaliação: porcentagem do treino, porcentagem do teste, acúmulo ou não do treino	Essencial
RF5	Identificação e listagem das classes	Disponibilizar as classes para previsão, identificadas por seu tipo precedido de seu nome	Essencial
RF6	Calcular e mostrar as medidas de classificação	Deverá ser calculado e mostrado o resultado, em porcentagem, da acurácia e do Kappa da matriz de confusão para a tarefa de classificação	Essencial
RF7	Matriz de confusão	Mostrar de forma esquematizada a matriz de confusão	Essencial
RF8	Calcular e mostrar as medidas de regressão	Deverá ser calculado e mostrado o resultado do coeficiente de correlação, RMSE, NRMSE, MAE para a tarefa de regressão	Essencial
	Registro do histórico	Registrar o histórico de cada avaliação, em tempo de	Importante

		execução, e permitir suas visualizações	

10. Requisitos Não-Funcionais

ID	Requisitos	Categoria
RNF1	A extensão deve implementar os recursos na janela principal do Explorer da WEKA por meio de uma nova guia.	Integração
RNF2	O software deverá ser distribuído no formato de um pacote para WEKA	Integração
RNF3	O usuário deverá ser informado sobre o andamento da execução das rotinas por meio da barra de status da WEKA.	Usabilidade
RNF4	A interface gráfica deve acompanhar o padrão visual da ferramenta.	Usabilidade
RNF5	A solução deve ser desenvolvida utilizando a linguagem de programação Java, a mesma utilizada no projeto da WEKA.	Implementação
RNF6	O software deve ser compatível com o requisito mínimo de versão do Java exigido pela ferramenta WEKA. Versões a partir da 3.7.1 exigem no mínimo o Java 6.	Implementação

11. Requisitos de Hardware

O desempenho da execução da tarefa de classificação na ferramenta WEKA está diretamente relacionado a quantidade de memória RAM disponível. Em geral, para casos onde um grande número de instâncias é utilizado, o software exige muita memória. No entanto, não há documentação dos requisitos mínimos de hardware para o WEKA.

Desta forma, as configurações apresentadas a seguir estão baseadas nos recursos necessários para funcionamento do ambiente Java, tecnologia na qual o WEKA é baseado, bem como na experiência do uso da ferramenta WEKA.

11.1. Configuração Mínima do Servidor

- 512 MB de memória RAM;
- 256 MB de espaço em disco.

11.2. Configuração Recomendada

- 8GB de memória RAM;
- 10 GB de espaço em disco.

12. Ferramentas de Desenvolvimento e Licença de Uso

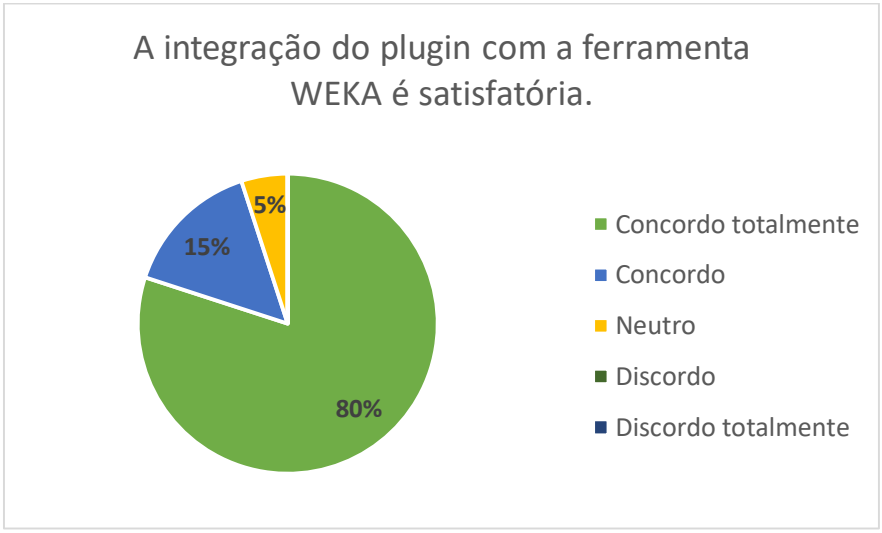
a) Eclipse Luna 4.4.1 (Eclipse Public License - v1.0);

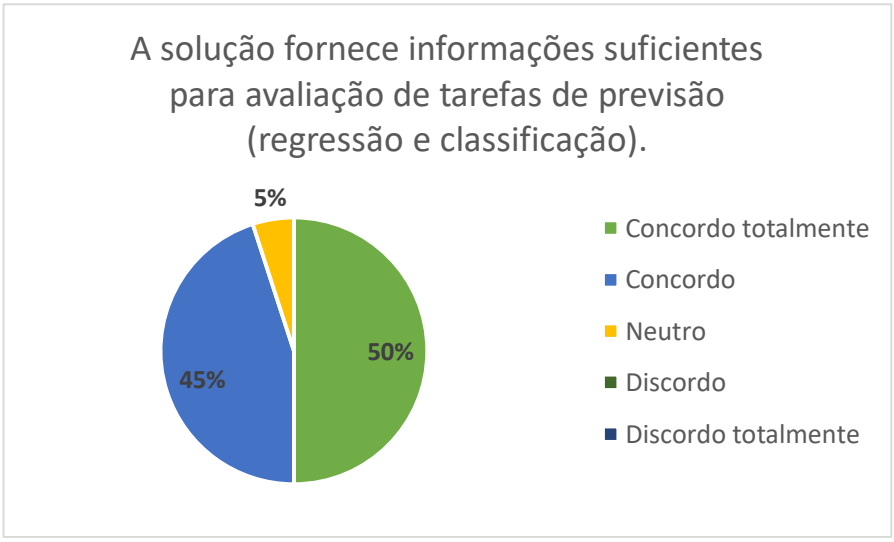
b) WEKA 3.7.12 (General Public License – v3.0);

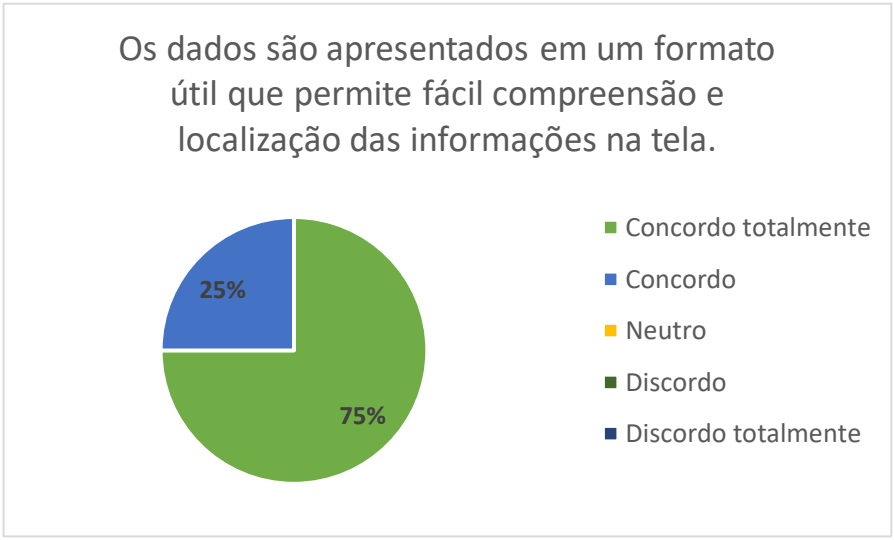
c) Java SE Development Kit 7 (Oracle Binary Code License);

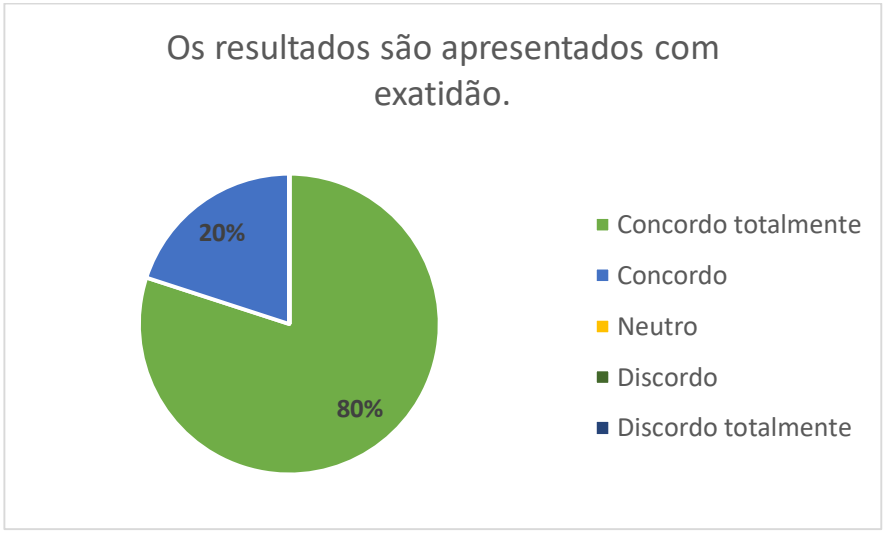
d) Apache Ant (Apache License - v2.0).

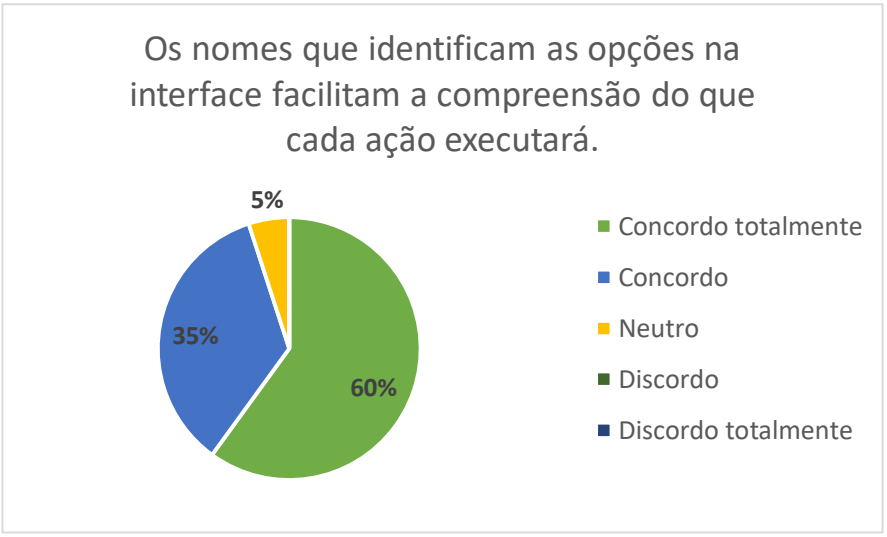
APÊNDICE B – RESULTADO DA AVALIAÇÃO DOS USUÁRIOS

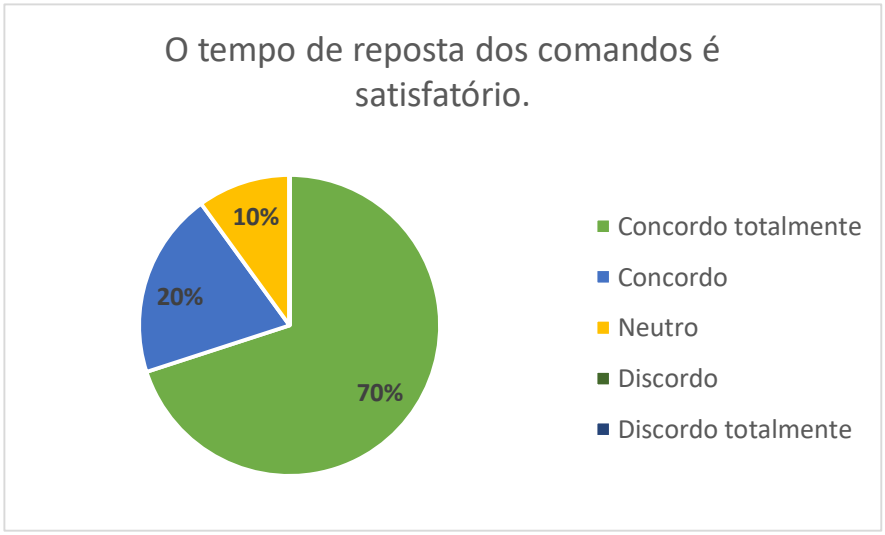

A integração do plugin com a ferramenta WEKA é satisfatória.
5%
15%
80%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

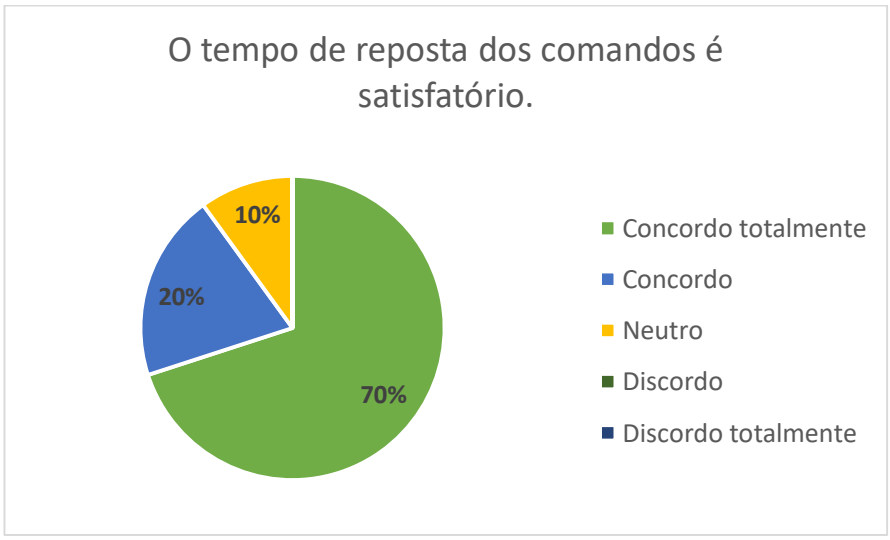

A solução fornece informações suficientes para avaliação de tarefas de previsão (regressão e classificação).
5%
50%
45%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

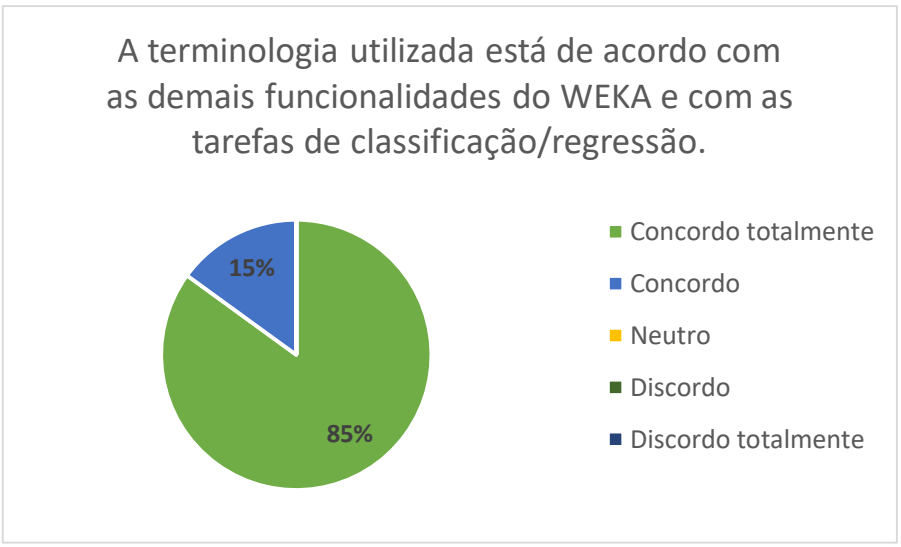

Os dados são apresentados em um formato útil que permite fácil compreensão e localização das informações na tela.
25%
75%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

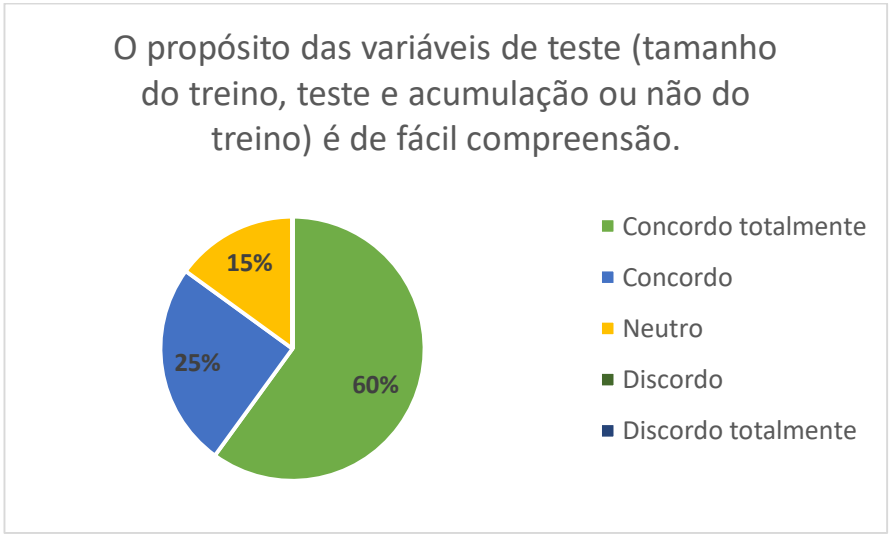

Os resultados são apresentados com exatidão.
20%
80%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente

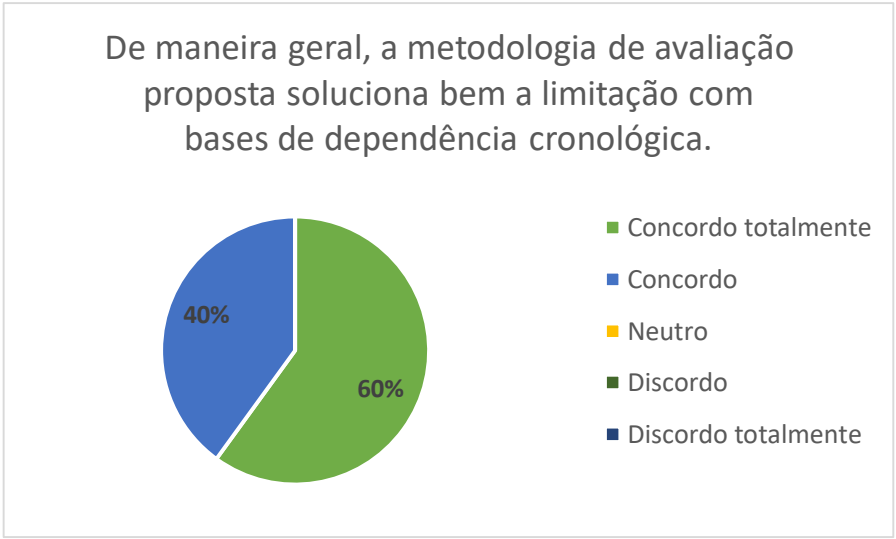

Os nomes que identificam as opções na interface facilitam a compreensão do que cada ação executará.
5%
35%
60%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente


O tempo de reposta dos comandos é satisfatório.
10%
20%
70%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente


O tempo de reposta dos comandos é satisfatório.
10%
20%
70%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente


A terminologia utilizada está de acordo com as demais funcionalidades do WEKA e com as tarefas de classificação/regressão.
15%
85%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente


O propósito das variáveis de teste (tamanho do treino, teste e acumulação ou não do treino) é de fácil compreensão.
15%
25%
60%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente


De maneira geral, a metodologia de avaliação proposta soluciona bem a limitação com bases de dependência cronológica.
40%
60%
Concordo totalmente
Concordo
Neutro
Discordo
Discordo totalmente