Universidade Federal Rural de Pernambuco

Departamento de Estatística e Informática

Comparação de Algoritmos do Aprendizado de Máquina Aplicados na Mineração de Dados Educacionais

Mirela Natali Vieira de Souza

Recife
Dezembro de 2015

Mirela Natali Vieira de Souza

Comparação de Algoritmos do Aprendizado de Máquina Aplicados na Mineração de Dados Educacionais

Orientadora: Roberta Macêdo M. Gouveia
Coorientadora: Andrêza Leite de Alencar

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Universidade Federal Rural de Pernambuco, como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Recife
Dezembro de 2015

A Deus.

Aos meus pais, Elias e Verônica.

À minha irmã, Marília Natalia.

Aos meus amigos.

Agradecimentos

Agradeço primeiramente a Deus. Pelo seu amor para comigo, por ter me concedido alcançar mais uma conquista em minha vida pois sem Ele nada seria possível. A minha família, pela compreensão e palavras de apoio e incentivo nos momentos difíceis, em especial minha querida mãe que esteve sempre junto me dando aquele apoio e teve mais fé do que eu, seu apoio foi muito importante e que me fez prosseguir.

As minhas orientadoras, Roberta Macêdo, que aceitou fazer parte dessa trajetória guiando minhas pesquisas e compartilhando seus conhecimentos com toda sua paciência e palavras de incentivo quando por vezes quis desistir deste trabalho, o meu muito obrigada e a minha coorientadora Andrêza Leite que participou também e me ajudou compartilhando seus conhecimentos e pela sua paciência, obrigada por tudo.

Aos meus amigos, àqueles que fizeram parte da família BSI que sem eles o curso não seria tão alegre mesmo naqueles períodos mais tensos em que passamos juntos, as amizades que se firmaram durante cada novo período que conquistávamos. Nossa saída que nunca aconteceu, os roubos de lanches (risos), os almoços no RU tudo isso vai ficar como boas lembranças. Obrigada por vocês fazerem parte da minha vida acadêmica e pessoal.

A todos os meus amigos que me incentivaram a lutar por isto, e acreditaram nisso quando eu já não achava que conseguiria. Agradeço a compreensão de cada um. O meu muito obrigada, vocês são demais!

A Andreza de Sousa Vieira, essa amiga que conquistei nesse pouco tempo e que foi muito atenciosa comigo e me incentivou bastante a terminar este trabalho. A minha revisora que tanto explorei neste TCC entrando pelas madrugadas. Pelo seu bom grado em me emprestar o notebook para que eu realizasse os testes deste trabalho, não tenho como retribuir tamanho gesto.

Agradeço a todos os professores e coordenadores da Universidade Federal Rural de Pernambuco, todos que contribuíram para minha formação acadêmica.

Resumo

Uma técnica comumente usada para a descoberta de conhecimento em grandes bases de dados relacionadas a qualquer tipo de contexto é chamada Mineração de Dados. Para que esta descoberta de conhecimento seja alcançada, é necessário atender as etapas do processo de Descoberta de Conhecimento em Base de Dados (*Knowledge Discovery in Databases* - KDD). Estas etapas vão desde a seleção, pré-processamento e transformação dos dados, até a própria implantação do conhecimento recém descoberto. Um exemplo de aplicação da mineração de dados é a Mineração de Dados Educacionais (MDE). Trata-se de uma área de pesquisa interdisciplinar que lida com o desenvolvimento de métodos para explorar dados oriundos de contextos educacionais. Uma aplicação da mineração de dados neste contexto significa coletar informações e conhecimentos de alunos com o objetivo de entender como eles estão explorando o ambiente de estudo e o que pode ser feito para tornar o espaço de estudo ainda mais atrativo para eles. Na mineração de dados, diversos algoritmos podem ser executados para a descoberta de conhecimento. A escolha de quais algoritmos executar é uma tarefa não trivial que deve ser feita considerando fatores, tais como, o volume de dados a serem processados, o tipo de aprendizado a ser obtido e as características dos dados. Para contribuir nesse sentindo, este trabalho empregou o processo KDD e executou algoritmos do aprendizado de máquina supervisionado e não-supervisionado na base de dados do ENEM 2013. Com os resultados gerados a partir dos algoritmos, foi realizado um estudo comparativo a partir da análise de métricas pré-estabelecidas, como o tempo de execução, estatística kappa e a taxa de acurácia (matriz de confusão) de cada algoritmo. Desta forma, pode-se tomar este estudo comparativo baseado nas métricas como apoio para a escolha de que algoritmos aplicar.

Palavras-chave: Processo KDD. Mineração de Dados Educacionais. Algoritmos de Aprendizado de Máquina.

Abstract

A technique commonly adopted for knowledge discovery in large databases related to any kind of context is named Data Mining. To achieve this knowledge discovery, it is required to follow the steps of the Knowledge Discovery in Databases - KDD. These steps range from the selection, pre-processing and data transformation to the implementation of the newly discovered knowledge. An example of data mining application is the Educational Data Mining. It is an area of interdisciplinary research that deals with the development of methods to explore data from educational contexts. The application of data mining in this context means to gather information and knowledge of students in order to understand how they are exploring the study of environment and what can be done to make the online study space even more attractive for them. During data mining, several algorithms can be executed for knowledge discovery. The choice of which algorithms perform is a nontrivial task that must be done considering factors such as the volume of data to be processed, the type of learning to be obtained and the characteristics of the data. In this sense, this study have applied the KDD process and implemented algorithms of supervised and unsupervised machine learning to ENEM 2013 database. With the results generated from the algorithms, a comparative study was conducted based on the analysis of pre-established metrics, such as execution time and the rate of accuracy. Thus, one can use this comparative study based on the metrics as a support for choosing which algorithms to apply.

Keywords: KDD Process. Educational Data Mining. Data Mining Algorithms.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A Mineração de Dados (MD) é uma área que vem sendo muito estudada e explorada nos últimos anos por proporcionar a multidisciplinaridade envolvendo Banco de Dados, Estatística, Matemática, Aprendizado de Máquina, Inteligência Artificial, Sistemas de Informação, entre outras áreas. O grande volume de dados que vem sendo gerado e os diferentes tipos (estruturados e não estruturados) requer cada vez mais a aplicação da MD por oferecer suporte para análise desses dados.

Para processar um grande volume de dados no contexto da mineração de dados é necessário atender as etapas do processo de Descoberta de Conhecimento em Base de Dados (*Knowledge Discovery in Databases* - KDD). O KDD compreende etapas ou fases que são importantes para gerar conhecimento ao final do seu processo. Segundo [1],

> Descoberta de conhecimento em bancos de dados é o processo não trivial de identificar padrões em dados que sejam válidos, novos (previamente desconhecidos), potencialmente úteis e compreensíveis, visando melhorar o entendimento de um problema ou um procedimento de tomada de decisão.

A mineração de dados em si é uma das etapas desse processo. Porém, é comum encontrar na literatura o termo "mineração de dados" sendo usado para referenciar o processo KDD como um todo.

Uma das aplicações do processo KDD é no contexto educacional, sendo conhecido na literatura como Mineração de Dados Educacionais (MDE) [2]. A MDE utiliza as mesmas tarefas da MD para gerar conhecimento, e seu objetivo é adequar as tarefas existentes a fim de resolver problemas específicos do contexto educacional. O surgimento desta área dentro da MD se deu devido a expansão do ensino à distância ao passo que cada vez mais dados são gerados em ambiente virtual de aprendizagem.

Considerando que a mineração de dados oferece uma grande variedade de algoritmos para aplicações nos mais diversos tipos de cenários, é possível identificar que há um esforço empregado na escolha de qual algoritmo aplicar na base de dados. A escolha do algoritmo deve

ser feita cuidadosamente, considerando vários fatores, tais como o volume de dados a serem processados, tipo de aprendizado a ser obtido, e características dos dados (tipos numéricos, strings, booleanos ou nominais). Diante do exposto, este trabalho se propõe a executar algoritmos de mineração de dados em um grande volume de dados e comparar os resultados obtidos, analisando algumas métricas pré-estabelecidas, como o tempo de execução, estatística kappa e a taxa de acurácia (matriz de confusão) de cada algoritmo.

Diante de uma variedade de algoritmos existentes e disponíveis na ferramenta WEKA, que será apresentada na seção 2.5, foram escolhidos: Id3, J48, Random Forest, ZeroR, OneR, sendo eles do aprendizado de máquina supervisionado, aplicando tarefas de classificação. Também foram selecionados os algoritmos Apriori e FilteredAssociator do aprendizado não supervisionado, por serem mais conhecidos e utilizados na área de MD, aplicando tarefas de associação.

Os algoritmos foram executados na base de dados do Exame Nacional do Ensino Médio (ENEM) 2013, disponível no portal do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), contendo dados gerais dos participantes da prova, além de dados específicos, que são os questionários socioeconômico dos participantes. Esta base contém um grande volume de dados, haja vista a quantidade de pessoas que realizam a prova do ENEM anualmente, uma média de 7,5 milhões de pessoas por ano, considerando os últimos três anos de aplicação da prova.

1.1 Motivação e Justificativa

Este trabalho está imerso em um contexto que envolve a aplicação da Mineração de Dados, Aprendizado de Máquina e Banco de Dados.

Dentro do estudo da MD existe a área de Aprendizado de Máquina (AM), que trata-se de um conjunto de técnicas capazes de adquirir conhecimento de forma automática, contemplando o estudo de dois tipos de aprendizado: o Aprendizado Supervisionado (AS), no qual o indutor, objeto que classifica os atributos a partir do conjunto de exemplos ou seja, um atributo alvo informado para classificar os atributos a partir do conjunto de exemplos (base de dados), cria grupos com características semelhantes; e o Aprendizado Não Supervisionado (ANS), que é a descoberta de conhecimentos através de um conjunto de exemplos inicialmente não rotulados.

Na área de Mineração de Dados existe pouca informação sobre quais algoritmos usar dentre as diversas opções que estão disponíveis para uso e para qual contexto de dados deve-se usá-los. Existem alguns trabalhos que abordam a comparação de algoritmos de aprendizado de máquina, porém dos identificados apresentam comparação entre algoritmos de um dos aprendizados.

Há uma necessidade de orientação no que diz respeito à escolha do algoritmo de mineração mais adequado para diferentes contextos, como por exemplo, os dados educacionais abordados neste estudo. Este trabalho não tem como objetivo descrever regras, gerar conhecimento a partir de uma base de dados, ou criar um processo para a escolha do melhor algoritmo, mas destacar que existem algumas variáveis importantes que devem ser observadas.

1.2 Objetivos

Este trabalho tem como objetivo geral empregar o processo KDD, com foco na fase de mineração de dados, e comparar os algoritmos do aprendizado de máquina supervisionado e não-supervisionado aplicados aos dados do ENEM do ano de 2013. Os dados coletados utilizados para a execução dos algoritmos são as informações gerais dos estudantes e as informações do questionário socioeconômico.

Foram analisados os algoritmos *OneR, ZeroR, Id3, J48 e Random Forest* do aprendizado de máquina supervisionado; e os algoritmos *Apriori* e *FilteredAssociator* do aprendizado não supervisionado através da execução em uma base de dados previamente definida. Os algoritmos já citados foram escolhidos por serem uns dos mais populares [3].

Dentre os objetivos específicos deste trabalho tem-se:

- Estudar ferramentas de mineração de dados existentes;
- Selecionar a base de dados a ser utilizada;
- Selecionar os algoritmos do aprendizado de máquina a serem analisados;
- Escolher métricas para analisar os algoritmos;
- Comparar entre si o desempenho dos 5 (cinco) algoritmos do aprendizado de máquina supervisionado; assim como os 2 (dois) algoritmos do aprendizado de máquina não supervisionado.

1.3 Organização do Trabalho

Este trabalho está organizado em seis capítulos, incluindo a introdução. A seguir uma breve descrição de cada um dos capítulos que se seguem:

- Capítulo 2: Fundamentação Teórica. Apresenta os conceitos fundamentais para o entendimento deste trabalho. São abordados o processo KDD, Mineração de Dados e seus conceitos, Mineração de Dados Educacionais, Aprendizado de Máquina, Dados Abertos e uma descrição do software WEKA usado para execução dos algoritmos e processamento dos dados;

- Capítulo 3: Metodologia. Apresenta a escolha da ferramenta, os algoritmos escolhidos, e os dados utilizados para o estudo;

- Capítulo 4: Resultados e Discussões. Apresenta a execução dos algoritmos do Aprendizado Supervisionado e Não Supervisionado, comparação e performance dos algoritmos;

- Capítulo 5: Conclusões. Apresenta as conclusões do trabalho desenvolvido, assim como sugestões de trabalhos que podem ser realizados futuramente;

- Referencial Bibliográfico.

Capítulo 2

Fundamentação Teórica

2.1 Processo KDD

É um processo não trivial de identificação de padrões válidos, novos, potencialmente úteis e compreensíveis presentes nos dados [3]. Este processo é composto pelas seguintes etapas: seleção dos dados, pré-processamento, transformação, mineração de dados, avaliação e implantação do conhecimento, como ilustra a Figura 1. Elas são brevemente descritas a seguir.

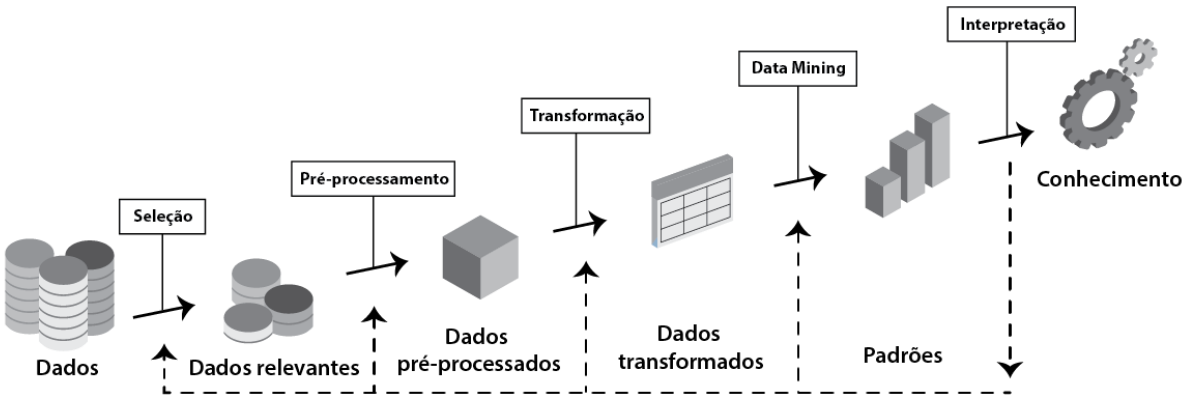

Figura 1: Etapas do processo KDD – Adaptado de Fayyad [1].

- **Seleção dos dados:** um conjunto de dados é escolhido pelo usuário e utilizado para aplicar a descoberta de conhecimento;
- **Pré-processamento:** esta fase é responsável pela limpeza dos dados relevantes, o que chamamos de eliminar os dados com ruídos, ou seja, dados sem padrão ou inconsistentes, assim como a formatação;
- **Transformação:** ocorre logo após a fase de pré-processamento, quando os dados são formatados de maneira adequada;
- **Mineração de Dados:** nesta fase alguns algoritmos são executados a fim de identificar padrões relevantes nos dados;

- **Interpretação e Avaliação:** os padrões extraídos são interpretados e avaliados de acordo com o critério definido pelo usuário, a fim de obter novos padrões de conhecimento;
- **Implantação do Conhecimento:** após a validação dos padrões extraídos pelas etapas anteriores, o conhecimento pode ser utilizado pelo usuário.

Embora seja importante a execução de todas as etapas do processo KDD, dada a importância de cada uma delas, em certos casos algumas dessas etapas podem ser suprimidas. Por exemplo, se os dados já estão limpos, consistentes e sem ruídos, a etapa de pré-processamento pode ser pulada. Dentre as etapas apresentadas, a MD é a mais importante no contexto deste trabalho e será detalhada na próxima seção.

2.2 Mineração de Dados

A mineração de dados é definida como o processo de descoberta de padrões em dados, de forma automático ou semiautomático [4]. Tais padrões são identificados com a aplicação de uma das metas da MD, que podem ser: Previsão ou Descrição. A descoberta desses padrões e novos conhecimentos, é realizada por meio de tarefas pertencentes à etapa de mineração de dados dentro do processo KDD. As tarefas da MD foram criadas para facilitar o entendimento e classificar as técnicas encontradas na mineração, sendo essas tarefas classificadas como direta ou indireta, segundo [5]. A "mineração direta" significa a análise ou categorização de um objetivo, ou seja, um atributo alvo (meta ou objetivo a ser encontrado). Por outro lado, a "mineração indireta" é a busca por padrões sem uma prévia determinação do que se está procurando, isto é, não se tem um objetivo a ser encontrado.

As tarefas comumente utilizadas na mineração são: classificação e regressão, pertencentes a meta de Previsão; já as tarefas de agrupamento ou clustering, associação e sumarização, pertencentes a meta de Descrição [5], conforme ilustra a Figura 2. Na meta de previsão, pertencente ao aprendizado supervisionado, as descobertas de padrões (conhecimento) são guiadas pelos dados, o que significa dizer que a classe alvo, ou seja o objetivo a ser classificado, é definido antes de iniciar a execução do algoritmo escolhido. A meta de descrição, relacionada ao aprendizado não supervisionado, é utilizada para descrever os padrões e tendências revelados pelos dados. Geralmente oferece uma possível interpretação para os resultados obtidos e é muito utilizada em conjunto com as técnicas de análise

exploratória de dados, para comprovar a influência de certas variáveis no resultado obtido. Nesta meta, os algoritmos identificam os padrões (ou seja, conhecimentos ou regas) sem que um atributo classificador, classe para qual os atributos irão convergir, seja informado. Neste trabalho a tarefa utilizada pertencente a esta meta foi a Associação e Classificação, que será brevemente descrita a seguir.

Figura 2: Principais tarefas da mineração de dados.

As tarefas da MD são utilizadas de acordo com o que se deseja obter como resultado da análise dos dados. Algumas delas são apresentadas a seguir [15], [17]:

- Classificação: uma das mais populares tarefas de mineração. Ela analisa os dados e busca correlações entre os atributos – propriedade ou característica de um objeto – a partir de exemplos (dados) para construir regras a fim de classificar novas classes – meta ou objetivo a ser encontrado. Esta tarefa identifica classes e constrói modelos (regras) a fim de predizer classes ainda não identificadas;
- Associação: busca encontrar relação dentro de um conjunto de valores onde é possível identificar padrão de comportamento entre seus atributos. Essa associação acontece quando a interseção dos itens (ou seja, as tuplas com os valores dos atributos) são verificadas, e em um segundo momento as regras de associação são criadas. Essa tarefa encontrar os padrões sequenciais (informação que se repetem dentro do conjunto de dados) ou a probabilidade de ocorrência. Por exemplo, se um cliente compra um bolo isso vai ou pode implicar na compra de outro produto, o refrigerante por exemplo;
- Cluster ou Agrupamento: a análise é feita sobre dados não classificados, tal análise é realizada a fim de criar subgrupos ou cluster com informações semelhantes. A criação desses subgrupos é feita a partir de medidas de similaridade. Por exemplo, a

segmentação de mercado ou interesses de usuários em uma rede social. A cada dado registrado o algoritmo identifica e cria subgrupos separando ou agrupando as informações;

- Regressão ou Estimativa: similar a classificação, esta tarefa busca correlação entre atributos numéricos. Por exemplo, a partir de características de um carro ou imóvel é possível determinar o valor de venda ou revenda;
- Sumarização: Segundo [1] a tarefa de sumarização envolve métodos para encontrar uma descrição compacta para um subconjunto de dados. Por exemplo, sumarizar os significados de padrões para os itens de dados.

Uma vez identificadas as tarefas, é necessário indicar o algoritmo a ser executado, pois ele será responsável pela execução dos dados e obtenção dos padrões, sendo executado de acordo com a tarefa selecionada na etapa da mineração.

2.2.1 Mineração de Dados Educacionais

A Mineração de Dados Educacionais (MDE) é uma área de pesquisa interdisciplinar que lida com o desenvolvimento de métodos para explorar dados oriundos de contextos educacionais [6]. Ela surge do mesmo princípio da mineração de dados, isto é, a partir da necessidade de entendimento de grandes volumes de dados gerados todos os dias por diferentes dispositivos. Na MDE não é diferente, ambientes e plataformas *online* educacionais geram volumes de dados de cada aluno todos os dias.

Aplicar tarefas de mineração de dados neste contexto significa coletar informações e conhecimentos do aluno com o objetivo de entender como ele está explorando o ambiente de estudo e o que pode ser feito para tornar o espaço de estudo *online* ainda mais atrativo para ele. Estas tarefas têm o potencial de fornecer informações úteis para auxiliar o desenvolvimento de ações, como por exemplo: determinar perfil do estudante, cursos e instituições; realizar análise de evasão e retenção de estudantes, obter indicadores de qualidade e planejamentos estratégicos para desenvolvimento educacional; diminuir o risco de evasão dos alunos; entender o estudante no processo de aprendizagem, observando o contexto; e melhorar a qualidade e as estratégias pedagógicas de um curso e instituição.

Com o objetivo de desenvolver as ações já citadas, dentre outros desafios e perguntas que surgem, a MDE é adaptada às tarefas da mineração de dados conforme suas necessidades

[7]. Na etapa de “pré-processamento de dados”, necessária em todo processo KDD, os dados são adequados ao processo de análise considerando que eles são oriundos de diferentes plataformas educacionais, para só assim aplicar análise sobre esses dados. A etapa de “transformação dos dados” também é utilizada em MDE, neste caso, para adequar o formato dos dados usados na plataforma de ensino a distância, tal como Ambientes Virtuais de Aprendizagem (AVA).

A MDE pode ser aplicada em diversas fontes de dados educacionais, tais como: bases de dados educacionais de uma plataforma de ambiente virtual de aprendizagem; dados como ENADE, Censo Escolar, Censo da Educação Superior, Prova Brasil, PNERA (Pesquisa Nacional da Educação na Reforma Agrária), Provão, Sistema Nacional de Avaliação da Educação Básica (Avaliação Nacional da Educação Básica/Prova Brasil) e ENEM, dentre outras.

2.3 Aprendizado de Máquina

Aprendizado de Máquina (AM) é uma subárea do estudo de Inteligência Artificial muito importante, pois tem o foco na extração de conhecimento de maneira automática. Estuda métodos computacionais para adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente [8]. A concepção deste conhecimento dá-se de maneira indutiva, ou seja, o conhecimento é guiado pelos dados.

O aprendizado indutivo é um dos mais úteis dentro do estudo de AM, existindo também o aprendizado Dedutivo, onde o conhecimento é guiado pelos especialistas, contemplando os ambientes de Data Warehouses e as tecnologias OnLine Analytical Processing (OLAP). A indução é uma inferência lógica de conhecimento a partir dos dados. O aprendizado indutivo é o processo de construção de um modelo (representação em escala reduzida), ou seja a base de dados, em que ela é analisada na procura de padrões. A partir deste modelo, os dados são analisados a fim de criar regras (agrupamentos de dados ou informações semelhantes).

O aprendizado de máquina se subdivide em duas técnicas, sendo elas: Aprendizado Supervisionado e Aprendizado não Supervisionado. Cada uma dessas técnicas é descrita a seguir, assim como os algoritmos pertencentes a cada tarefa.

2.3.1 Aprendizado Supervisionado

O Aprendizado Supervisionado (AS) é derivado da meta de previsão da MD, uma vez que essa meta identifica os padrões (conhecimentos) a partir de um atributo classificador informado pelo analista. Este aprendizado é dito supervisionado pois existe um treinamento antes da 'construção' do conhecimento, na qual os dados da base de dados que estão sendo analisados pelos algoritmos são induzidos pelo atributo classificador previamente informado.

2.3.1.1 Algoritmo ID3

Este algoritmo possui uma abordagem indutiva (*top-down induction of decision trees* - TDIDT), Trata-se de um método que deriva conhecimento novo e prediz resultados baseados nos valores dos atributos em sua essência. Este algoritmo pertence ao estudo de AM (técnica de descoberta de conhecimento de forma automatizada)

Proposto por Quilan [4], sua principal função é a construção de uma árvore de decisão de tamanho compacto a fim de obter a melhor regra de decisão. O processo para resultar em uma árvore de decisão é feito pelo particionando do conjunto de treinamento de maneira recursiva, até que os subconjuntos sejam de uma mesma classe (grupo).

A estruturação do ID3 tem o seguinte comportamento: a composição da árvore utiliza o método *top-down*, partindo da raiz até o nó folha, onde resultam classes de um conjunto de treinamento. O algoritmo elege o atributo mais frequente como sendo os nós raiz de forma estatística.

Uma vez o nó raiz identificado seus subconjuntos, serão separados de acordo com as classes pertencentes. A divisão é realizada através de um único atributo, que é selecionado a partir de uma propriedade estatística, denominada ganho de informação, que mede quanto informativo é o atributo[10].

O processo é recursivo, gerando sub-árvores até que se atenda um critério de parada, que no caso ideal seria obter nós contendo apenas exemplos de uma mesma classe [18]. De uma maneira geral, os passos do algoritmo ID-3 são apresentados abaixo.

1. Dado um Nó na árvore e todas as tuplas do conjunto de treinamento S;
2. Selecione o melhor atributo A para o nó;

3. Para cada valor v, de A, cresça uma subárvore, ou uma folha, sob o nó.

2.3.1.2 Algoritmo J48

O algoritmo J48 trata-se de um dos mais conhecidos algoritmos de classificação usado em mineração de dados. O J48 [11] é uma implementação do algoritmo C4.5 releases 8 de Quinlan [12] para gerar uma árvore de decisão podada ou não podada [11]. O J48 é uma evolução do algoritmo ID3, também desenvolvido por Quinlan, surgindo da necessidade de recodificar o C4.5 tendo como finalidade gerar árvore de decisão baseada em um conjunto de dados de treinamento, método usado para classificar instâncias no conjunto de teste.

Dado um conjunto de treinamento, a árvore é montada de forma que o atributo mais frequente seja capaz de formar conjunto de categorias. Qualquer um dos atributos analisados pelo algoritmo pode ser responsável por separar os dados em conjuntos menores pertencentes a mesma categoria. Uma vez escolhido o atributo com a maior entropia (ganho de informação), um nó de decisão é criado a fim de quebrar em sub-árvores menores.

O atributo com maior ‘‘número” de informação (ocorrências) é selecionado como nó raiz a fim de construir a árvore. As folhas são geradas para cada valor do atributo do topo da árvore e cada uma delas possui um subconjunto de vetores associados pertencente a mesma categoria [13].

Um dos aspectos para a grande utilização deste algoritmo é que o mesmo mostra-se adequado para os procedimentos envolvendo as variáveis (dados) qualitativas e variáveis quantitativas contínuas e discretas presentes nas Bases de Dados.

2.3.1.3 Algoritmo Random Forest

Criado por Leo Breiman, este algoritmo tem a premissa básica de criar várias árvores de decisão de maneira aleatória a fim de determinar a classe de uma instância. Dado um conjunto de instâncias (dados) original, este conjunto é dividido de forma randômica gerando outros subconjuntos. Para cada subconjunto gerado uma árvore é criada e um classificador é identificado, que este é utilizado para ser votado e eleger uma classe.

Para eleger uma classe é feita uma votação, onde cada árvore dá seu voto. A classificação final é dada pela classe que recebeu o maior número de votos entre todas as árvores da floresta [14].

O *Random Forest* implementa o método *Bagging*. No modelo de geração deste método, para cada árvore gerada, um número X de instâncias são criadas, e para cada nó dessas instâncias uma quantidade M de atributos são escolhidos de forma também aleatória com a finalidade de "guiar/orientar" o direcionamento (o grupo ou a classe) que o nó pertence.

2.3.1.4 Algoritmo ZeroR

O algoritmo ZeroR classifica a amostra (conjunto de dados) com uma única regra. Em um conjunto de dados a classe mais frequente é prevista nos dados de treinamento para problemas com valores nominais ou valor da classe comum para problemas de predição numéricos [9], criando uma tabela de frequência identificando a classe alvo.

O método de classificação do ZeroR é dependente da classe alvo e ignora os seus atributos preditores para gerar uma regra ao final da execução. A figura 3 ilustra para melhor entendimento os atributos preditores (predictors) e alvo (target). É possível observar para a classe "Yes" que 9 instâncias foram classificadas e para a classe "No" foram classificadas 5 instâncias.

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Figura 3: Exemplo de classificação de regras do algoritmo ZeroR

2.3.1.5 Algoritmo OneR

O algoritmo OneR cria uma regra para cada atributo dos dados de treino e seleciona a regra com menor percentagem de erro como regra única. Para criar uma regra para um atributo é necessário determinar a classe mais frequente para cada atributo.

Este algoritmo realiza testes em cada atributo do seu conjunto de dados criando regras para construção da árvore. Cada atributo é testado e associado à classe mais frequente. Para cada instância da base de dados são analisados os atributos e seus respectivos valores onde uma regra será criada, sendo responsável por contar a quantidade de vezes em que a classe aparece. Ao final de cada uma dessas etapas a regra atribui essa classe ao valor do atributo.

2.3.2 Aprendizado Não Supervisionado

O Aprendizado Não Supervisionado (ANS) é derivado da meta descrição da MD. Este aprendizado identifica os padrões (conhecimentos) através de estudo e análise da base de dados, uma vez que para esta técnica o atributo classificador não é informado.

Dado uma base de dados os algoritmos geram as regras a partir de observações dos atributos e são gerados agrupamentos com informações semelhantes utilizando métodos probabilísticos, pois as classes não são definidas.

2.3.2.1 Algoritmo Apriori

Considerado um clássico na extração de Regras de Associação e foi concebido pelo centro de pesquisa da IBM − "*The Quest Data Mining System, IBM Almaden Research Center*". O algoritmo realiza buscas sucessivas em toda a base de dados, mantendo um ótimo desempenho em termos de tempo de processamento [10]. O algoritmo observa a frequência em que o item aparece dentro do conjunto de itens denominado L1 utilizando o método candidato. Tal método é utilizado para encontrar associações relevantes entre itens de dados. São definidos parâmetros que determinam quais associações são interessantes ou não para o usuário.

Para eliminar regras que não aparecem com frequência no banco de dados é usado o suporte, que indica a porcentagem de registros onde aparecem juntos X e Y, sobre o total de registros. Outra medida, que indica o grau de acerto da regra, é a confiança, a qual indica a

porcentagem de registros que contêm X e Y, sobre o total de registros que possuem X. Este algoritmo tem como premissa utilizar os itens ou atributos mais frequentes de um conjunto para criar as regras reduzindo assim o número de candidatos que seriam comparados em cada transação. No exemplo abaixo os itens são apresentados na Tabela 1, e na Tabela 2 é possível observar que os elementos mais frequentes selecionados são café, pão e manteiga.

Transações
{pão, leite}
{pão, fralda, cerveja, ovos}
{leite, fralda, cerveja, coca}
{pão, leite, fralda, cerveja}
{pão, leite, fralda, coca}

Tabela 1: Transações

Conjunto de itens	Suporte
{pão, leite}	**3**
{pão, cerveja}	2
{pão, fralda}	**3**
{leite, cerveja}	2
{leite, fralda}	**3**
{cerveja, fralda}	**3**

Tabela 2: Conjunto L1 – pares de itens

Após repetidas execuções é possível identificar um conjunto de itens frequentes de tamanho 1 sendo o conjunto de itens {pão, leite, fralda}. Chegou-se a este conjunto considerando as transações que apresentaram suporte maior, ou seja os elementos mais frequentes, sendo eles destacados acima na Tabela 2.

2.3.2.2 Algoritmo Filtered Associator

Este algoritmo é uma classe para a execução de um associador arbitrária sobre os dados que foram passados através de um filtro arbitrário. Como o associador, a estrutura do filtro é baseada exclusivamente nos dados de treino, e exemplos de teste será processada pelo filtro sem alterar a sua estrutura. Em Weka inclui opção como associador com o qual podemos considerar o Apriori, Predictive Apriori, Tércio de regras de associação e algoritmo Filtered Associator, índice de classe e filtro para obter o resultado [21].

2.4 Dados Abertos Governamentais

Os dados abertos governamentais é uma forma de disponibilizar dados e informações para a sociedade com objetivo de que valorizar a comunicação entre cidadão e governo. A iniciativa deste projeto é do Portal Brasileiro de Dados Abertos [13], que vem disponibilizando os dados por meio de uma ferramenta *online*, permitindo fácil acesso às informações dos dados públicos. Esse mecanismo de disponibilização dos dados permite que novas soluções sejam criadas para e pela sociedade. Algumas soluções podem ser citadas como resultado da disponibilização desses dados, como por exemplo: gastos governamentais, indicadores de educação, processo eleitoral, sistema de transporte, entre outros [13].

De acordo com a Constituição Federal e a Declaração Universal dos Direitos Humanos, Dados Abertos é a publicação e disseminação dos dados e informações públicas na Internet, organizados de tal maneira que permite a reutilização em aplicativos digitais desenvolvidos pela sociedade [13].

David Eaves [15] propôs um conjunto de leis para os Dados Abertos Governamentais, como seguem:

- Se o dado não pode ser encontrado e indexado na Web, ele não existe;
- Se não estiver aberto e disponível em formato compreensível por máquina, ele não pode ser reaproveitado;
- Se algum dispositivo legal não permitir sua replicação, ele não é útil.

Entretanto, pode-se dizer que estas leis se aplicam aos Dados Abertos de forma geral [13]. Sabendo que todo dado público tem vocação para ser dado aberto, contanto que não possua

restrições de sigilo, o Governo está cada vez mais se valendo desse formato de disponibilização de dados, com o objetivo de facilitar o uso dessas informações pelo cidadão. Uma das possíveis utilizações dos Dados Abertos é no desenvolvimento de aplicativos, que exibem as informações de forma gráfica e interativa. Um exemplo é Ministério da Justiça que já realizou dois concursos de aplicativos de dados abertos, incentivando o desenvolvimento, pela sociedade, de aplicativos de fácil visualização e cruzamento de dados de interesse público extraídos de bases de dados do Ministério.

Um dos formatos abertos mais comuns de publicação é o CSV (*Comma Separated Values*, ou "valores separados por vírgula"), para armazenamento de dados tabulares em texto. Outros exemplos de formatos não proprietários são: JSON, XML, ODS e RDF. Formatos populares como PDF, por exemplo, não são estruturados, dificultando o pleno aproveitamento dos dados contidos no documento. Várias órgãos/instituições já possuem a cultura de divulgação de seus dados, e exemplo do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) [14] que disponibiliza dados estruturados educacionais para realização de estudos e consultas, tais como: ENADE, Censo Escolar, Censo da Educação Superior, Prova Brasil, PNERA, Provão, Saeb (Aneb/Prova Brasil), ENEM [14], entre outras fontes de dados educacionais que seguem os princípios e leis [15] para a disponibilização como Dados Abertos.

2.5 Ferramentas de MD

Atualmente existem diversas ferramentas de mineração de dados que auxiliam na execução das etapas de processo KDD. A Tabela 3 apresenta as principais ferramentas destacando, as tarefas do processo KDD que elas contemplam e seus fabricantes.

Ferramenta	**Tarefas**	**Fabricante**
Oracle Data Minig	Classificação, Regressão, Associação, clusterização, Mineração de Textos.	Oracle http://www.oracle.com/
IntelligentMiner	Classificação, Associação, Sequência, Clusterização, sumarização.	IBM corporation www.ibm.com
R	Clusterização, Associação, Classificação.	https://www.r-project.org/
WEKA	Classificação, Associação, Clusterização.	University of Waikato http://www.cs.waikato.ac.nz/ml/weka

Tabela 3: Ferramentas de mineração de dados.

As principais ferramentas de mineração de dados foram pesquisadas no intuito de escolher a mais apropriada para a execução dos algoritmos neste trabalho.

Desenvolvido por um time de pesquisadores, este software provê fácil interação, aplicação com aprendizado de máquina, fornecendo autonomia a WEKA é um software gratuito escrito na linguagem Java, desenvolvido pela Universidade de Waikato na Nova Zelândia e é um software livre. O software contém ferramentas para visualização de algoritmos para análise de dados e predição de modelos [16]. Apresenta uma interface gráfica de fácil manuseio das funcionalidades.

A versão mais atual do software WEKA foi desenhada para utilização de diversos tipos de dados em diferentes áreas de aplicação, oferecendo suporte a diversos algoritmos, apresentação gráfica de seus resultados e suporte a execução de tarefas de mineração, e ajuste no core dos algoritmos. Oferece também implementação das etapas do KDD que são pré-processamento, classificação, regressão, agrupamento ou cluster, regras de associação. A Figura 3 apresenta a etapa de pré-processamento do WEKA, e visualização gráfica das árvores como mostra a Figura 4, além de executar algoritmos de Classificação, Associação e Agrupamento ou Cluster.

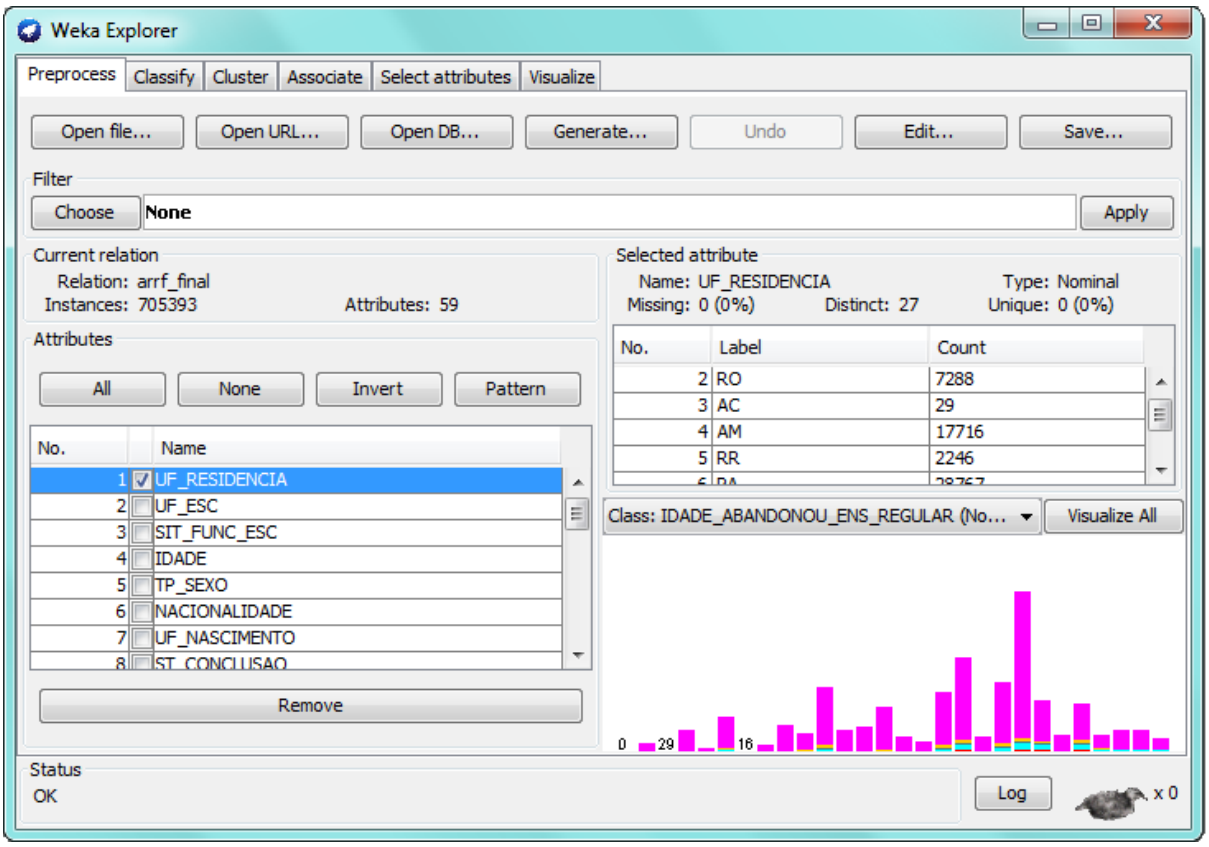

Figura 4: Visualização da Etapa do KDD (Pré-processamento).

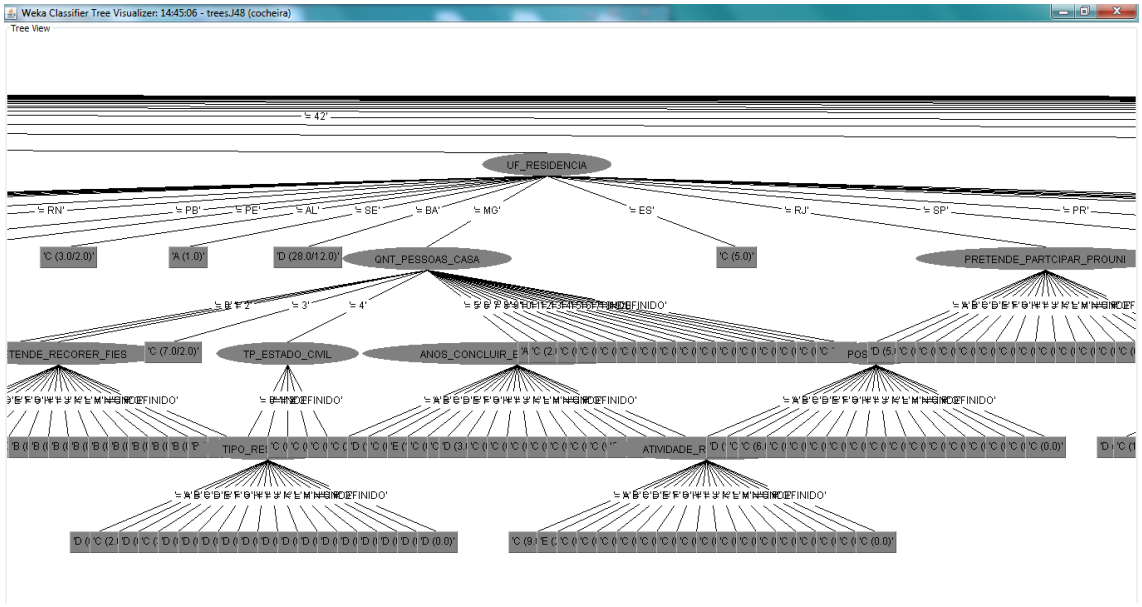

Figura 5: Visualização da árvore na ferramenta WEKA.

2.6 Trabalhos Relacionados

No cenário atual da internet colaborativa, os dados são produzidos de várias fontes, sejam eles produzidos pela internet, por dispositivos de armazenamento, negócios, medicina ou qualquer aspecto da nossa vida diária [12], ou ainda por dados educacionais que estão presentes em toda parte e faz-se necessário tirar conhecimento desses dados. Observando este cenário e conhecendo as tarefas da Mineração de Dados assim como cada um de seus algoritmos, este trabalho propôs a estudar os algoritmos de aprendizado supervisionado e não supervisionado.

O artigo “Análise dos Algoritmos de Mineração J48 e Apriori Aplicados na Detecção de Indicadores da Qualidade de Vida e Saúde” teve uma importante colaboração para a fundamentação teórica deste trabalho de conclusão de curso, uma vez que as definições das tarefas de classificação e associação foram suficientes pelas autoras [20] para o entendimento a fim de desenvolver este trabalho.

O trabalho de Solange Rubert Librelotto, Patricia Mariotto Mozzaquatro [23] tem como objetivo o estudo das tarefas de MD, classificação e associação comparando J48 e Apriori, aplicadas na identificação e classificação de indicadores de saúde a fim de gerar um perfil de usuário. O trabalho [23] realiza uma análise do algoritmo J48 e uma análise do algoritmo Apriori, o que difere da abordagem de estudo deste trabalho de conclusão de curso, onde são realizadas comparações com um número maior de algoritmos entre eles: ID3, J48, OneR, ZeroR e Random Forest testando os mesmos em um sistema de aprendizado de máquina, WEKA, em

três configurações distintas, e comparações entre algoritmos de associação: Apriori e FilteredAssociator utilizando-se também da mesma ferramenta.

A monografia “Estudo Comparativo dos Algoritmos de Classificação da Ferramenta WEKA” apresentou definições do aprendizado de máquina e algumas descrições sobre a WEKA e os algoritmos de classificação, usados no trabalho de Lima [27], como também algumas configurações de saída dos dados processados que foram minerados. O trabalho contribui para o entendimento da ferramenta utilizada para o desenvolvimento deste trabalho de conclusão assim como a definição do conceito de Aprendizado de Máquina. Este trabalho utilizou dados fornecidos pela ferramenta WEKA com uma base de dados contendo 70 tuplas e 30 tuplas, não sendo necessário criar um programa para transformar os dados.

Capítulo 3

Metodologia

Inicialmente, foi realizado uma pesquisa bibliográfica em busca de trabalhos semelhantes com aplicações do estudo do processo de KDD, Mineração de Dados, Aprendizado de Máquina e Mineração de Dados Educacionais com a finalidade de identificar métricas e diretrizes para comparação dos algoritmos do Aprendizado Supervisionado e Não Supervisionado.

O objetivo é identificar diretrizes para comparação de algoritmos de Aprendizado de Máquina, assim como analisar as saídas geradas pela ferramenta WEKA. E para compor este trabalho foi utilizado uma base de dados educacional aberta em formato CSV disponível no site do INEP.

Para o desenvolvimento do trabalho fez-se necessário: (I) realizar as etapas de seleção, pré-processamento e transformação dos dados pertencentes ao processo KDD (seção 3.1); (II) selecionar os algoritmos do Aprendizado Supervisionado e Aprendizado Não Supervisionado (seção 3.2); e (III) definir as métricas para avaliação dos algoritmos (seção 3.3).

3.1 Seleção, Pré-processamento e Transformação dos Dados

Esta seção descreve as atividades que foram executadas neste trabalho para realizar às etapas de seleção, pré-processamento e transformação dos dados. Desta forma, possibilitando a execução dos algoritmos de mineração.

Seleção

No início deste trabalho, várias bases de dados educacionais obtidas a partir do INEP foram analisadas a fim de selecionar uma para a execução dos algoritmos. A seleção da base de dados foi realizada de acordo com dois critérios: (i) o tamanho da base – ela deveria contemplar um grande número de atributos e possuir uma extensa quantidade de tuplas; e (ii) a consistência dos dados – eles deveriam ser os mais íntegros possíveis, de forma a economizar tempo e esforço na etapa de pré-processamento.

Como resultado da etapa de “seleção” do processo KDD, levando em consideração estes dois critérios, foi escolhida a base de dados do ENEM 2013. Esta base contempla os dados do questionário socioeconômico respondido pelos participantes, bem como os seus dados gerais, tais como informações pessoais e de necessidades especiais.

A base de dados do ENEM 2013 é um arquivo de 4,7 GB, que possui mais de 7 milhões de tuplas e um total de 166 atributos. Dentre estes atributos foram selecionados 59 para a realização do estudo, como mostra a Tabela 2 (dados socioeconômicos) e Tabela 3 (dados gerais dos estudantes).

Esses atributos foram selecionados pois considerou-se mais relevantes para execução dos algoritmos, a fim de testar o desempenho com uma quantidade considerável de atributos.

grau_estudo_pai	anos_concluir_ens_medio
grau_estudo_mae	tp_esc_cursou_ens_medio
renda_mes_familia	pretende_partcipar_prouni
qnt_pessoas_casa	pretende_recorer_bolsa
tipo_residencia	pretende_recorer_fies
localizacao_residencia	ch_trabalho
possui_pc	fez_vestibular
acesso_internet	fez_faz_eja
atividade_remunerada	frequentou_ens_regular
anos_concluir_ens_fundamental	idade_abandonou_ens_regular
tp_esc_cursou_ens_funda	

Tabela 4: Atributos referentes aos dados socioeconômicos dos estudantes.

uf_residencia	in_deficiencia_mental
uf_esc	in_deficit_atencao
sit_func_esc	in_dislexia
Idade	in_gestante
tp_sexo	in_lactante
Nacionalidade	in_idoso
uf_nascimento	in_autismo
st_conclusao	in_braille
ano_concluiu	in_ampliada_24
tp_escola	in_ampliada_18
in_tp_ensino	in_ledor
tp_estado_civil	in_acesso
tp_cor_raca	in_transcricao

in_baixa_visao	in_libras
in_cegueira	in_leitura_labial
in_surdez	in_mesa_cadeira_rodas
in_deficiencia_auditiva	in_mesa_cadeira_separada
in_surdo_cegueira	in_apoio_perna
in_deficiencia_fisica	in_guia_interprete

Tabela 5: Atributos referentes aos dados gerais dos estudantes.

Pré-processamento

Durante a etapa de "pré-processamento" do processo KDD foram realizadas atividades tanto para extrair da base de dados apenas os atributos selecionados para este estudo quanto para realizar a limpeza dos dados.

No intuito de auxiliar esta etapa do processo, um programa foi implementado na linguagem PHP. O programa recebe como entrada um arquivo do tipo CSV referente à base de dados do ENEM com os seus 166 atributos. Em seguida, ele extrai apenas os dados referentes aos atributos selecionados para a realização deste estudo, apresentados anteriormente nas Tabelas 2 e 3. A partir destes dados o programa realiza a limpeza, isto é, a eliminação de dados inconsistentes. A limpeza dos dados foi feita pela substituição de cada campo vazio, tendo em vista que alguns algoritmos não trabalham com valores de atributos vazios, pelo valor "Indefinido" do tipo String, assim como cada pergunta do questionário socioeconômico foi transformado em atributo. Por fim, o programa gera como saída um arquivo do tipo CSV com os 59 atributos selecionados e com todos os dados consistentes.

Transformação

A ferramenta WEKA recebe como entrada um arquivo do tipo ARFF com uma estrutura específica, onde devem ser definidos os tipos de cada atributo. Portanto, para que a base de dados do ENEM 2013 seja executada pelos algoritmos da ferramenta WEKA foi necessário realizar a etapa de "transformação" do processo KDD.

Um script foi implementado em PHP, para auxiliar na transformação dos dados, como mostra o Anexo I deste documento. Este programa recebe como entrada um arquivo do tipo CSV com os dados já pré-processados, isto é, o arquivo gerado como saída do programa usado na etapa de pré-processamento dos dados. Para este estudo escolheu-se usar apenas o tipo "nominal", dado que todos os algoritmos da WEKA que foram executados aceitam este tipo de dado.

3.2 Seleção dos Algoritmos

Dentre os algoritmos do Aprendizado Supervisionado foram selecionados ID3, J48 e o Random Forest (como mostra a Figura 5) – sendo eles de árvore de decisão (Classify→Trees), OneR e ZeroR – algoritmos de regra(Classify→Rules). Estes algoritmos foram escolhidos devido ao vasto uso em tarefas de mineração de dados e geração de conhecimento, quantidade de material disponibilizado assim como o guia de estudos desses algoritmos fornecido no site do WEKA.

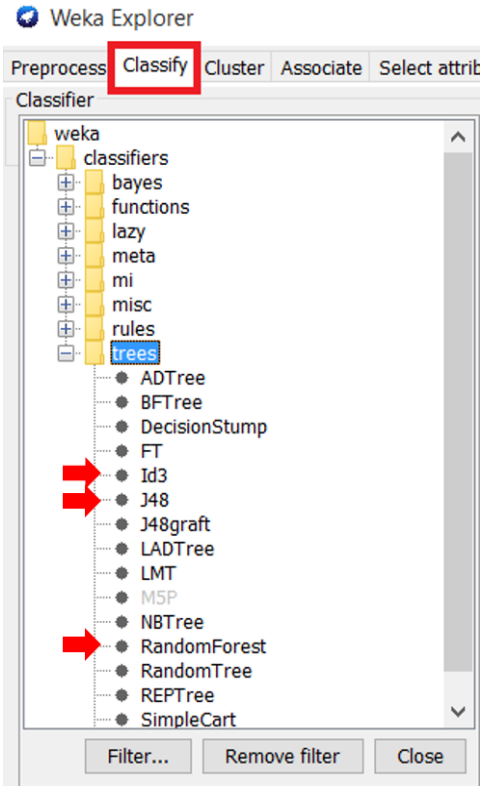

Figura 6: Algoritmos do aprendizado supervisionado selecionados da ferramentaWEKA.

A escolha dos algoritmos foi baseada nas características apresentadas por cada um deles tais como: ID3 que utiliza o método *informationgain* ideal para definição do atributo mais informativo para a construção da árvore de decisão, J48 faz uso da técnica dividir para conquistar mostrando-se adequado na decomposição de um problema em subproblemas, e o Random Forest que possui em sua implementação o método *bagging* selecionando de maneira aleatória atributos com a finalidade de direcionar à classe.

No Aprendizado Não Supervisionado foram escolhidos o Apriori e o Filtered Associator, como mostra a Figura 6. São algoritmos clássicos para extração de regras de associação, e apresentam bom desempenho em termos de processamento devido ao uso da

técnica dividir para conquistar, utilizada para realizar sucessivas buscas na base de dados a fim de encontra regras de associação.

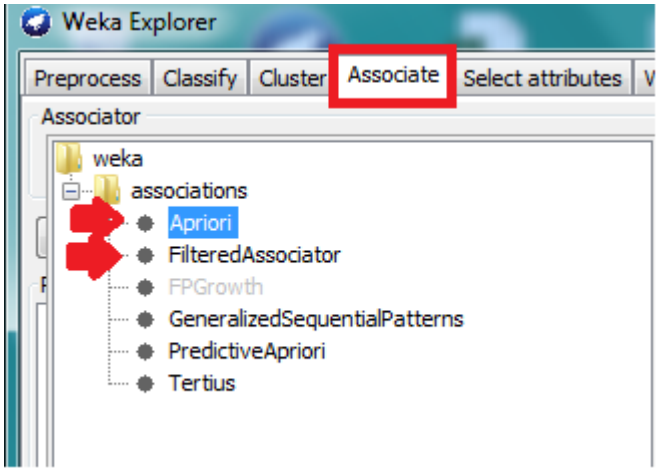

Figura 7: Algoritmos do aprendizado não supervisionado selecionados da WEKA.

Os algoritmos foram executados observando os resultados de saída gerados pela ferramenta WEKA utilizada neste trabalho para realizar os testes de comparação de cada um dos algoritmos já citados. Os testes de comparação são abordados no capítulo seguinte.

3.3 Definição das Métricas

Para realizar a comparação entre os algoritmos, fez-se necessário definir um conjunto de métricas a serem observadas tanto para os algoritmos de classificação quanto para os algoritmos de associação. As métricas utilizadas foram adotadas com base nos resultados apresentados na ferramenta WEKA, ferramenta comumente utilizada para estudo e execução de algoritmos de mineração de dados.

3.3.1 Métricas para Algoritmos de Classificação

As métricas adotadas para os algoritmos de classificação são descritas a seguir:

- **Classificação de corretude das instâncias**: também conhecida como amostra de acurácia, esse dado informa a porcentagem das instâncias classificadas de maneira

correta, e representa a acurácia resultante após a execução do algoritmo em um conjunto de dados;

- **Classificação de instâncias incorretas**: identifica a quantidade de instâncias classificadas incorretamente e é demonstrada na matriz de confusão;
- **Tempo gasto para análise dos dados e criação da árvore**: o tempo gasto para análise da base de dados, ou seja gerar as regras e construção do modelo;
- **Estatística Kappa (K)**: é uma métrica que avalia o nível de concordância de uma tarefa de classificação. A estatística K é uma medida de concordância usada em escalas nominais que nos fornece uma ideia do quanto as observações se afastam daquelas esperadas, fruto do acaso, indicando-nos assim o quão legítimas as interpretações são [19], ou seja, confirma em termos de porcentagem o quão correto as instâncias foram classificadas correta e incorretamente. O índice Kappa estabelece intervalos de valores nos quais resumem a taxas como sedo; boas, ruins, moderadas, substanciais e excelentes. Esses intervalos são listados na Tabela 6

Valor do Kappa	Concordância
0	Ruim
0 – 0,20	Ligeira
0,21 – 0,40	Considerável
0,41 – 0,60	Moderada
0,61 – 0,80	Substancial
0,81 – 1	Excelente

Tabela 6: Intervalos de valores do Kappa [19].

- **Matriz de confusão**: também conhecida como tabela de contingência, permite uma melhor visualização da performance do algoritmo em termos de distinguir os resultados apresentados pela classe. A representação dos resultados é apresentada em uma matriz de confusão onde os valores da diagonal principal indicam as instâncias classificadas corretamente e a diagonal secundária as instâncias classificadas incorretamente.

Utilizar a matriz de confusão como parâmetro para analisar a performance do algoritmo é entender como as instâncias de seus dados estão sendo analisadas, sejam elas corretas ou incorretamente classificadas pelo algoritmo. É importante atentar para a matriz de confusão, pois essa métrica não garante 100% de acurácia para o algoritmo utilizado, uma vez que o sistema de classificação pode não distinguir entre uma instância ou outra da classe informada pelo usuário. Porém, é possível identificar todas as hipóteses geradas na diagonal da matriz. Abaixo segue um exemplo da matriz de confusão.

Para este exemplo a figura 7 mostra o atributo classificador (classe que representa o objetivo a ser encontrado ao final das regras geradas) com ocorrência de dois domínios (valores) para melhor entendimento, sendo eles: ‘M’ identificando o domínio masculino, representado pela letra ‘a’ e ‘F’ identificando o domínio feminino representado pela letra ‘b’.

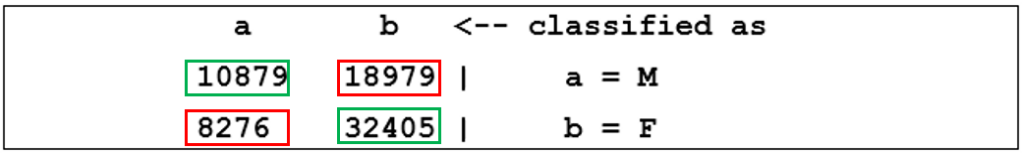

Figura 8 Resultado Matriz de Confusão

A matriz de confusão faz parte dos algoritmos classificadores, em especial da tarefa de classificação do *Data Mining* e oferece meios efetivos para a avaliação do modelo com base em cada classe. As instâncias apresentadas na matriz foram classificadas corretamente (cor verde) são 10879 + 32405 = 43284 e as instâncias classificadas incorretamente (cor vermelha) são: 8276 + 18979 = 27255. Esses resultados apresentam uma taxa de acerto de 0,61 (61%) sendo calculadas dividindo-se a quantidade total de instâncias corretas pelo total de instâncias do conjunto de dados: 43284/70539= 0,61, e uma taxa de erro de 0,39 (39%) sendo calculadas dividindo-se quantidade total de instâncias incorretas pelo total de instâncias do conjunto de dados: 27255/70539 = 0,39.

3.3.2 Métricas para Algoritmos de Associação

As métricas utilizadas para avaliar as regras geradas pelos algoritmos de associação utilizados neste trabalho são o Suporte e a Confiança. Sendo definidas a seguir:

- **Suporte**: calcula o percentual de vezes em que um determinado conjunto X aparece no conjunto de transações, ou seja, contabiliza quantas vezes determinados itens apareceram na base de dados. Dado que o conjunto (a base de dados) é informada à ferramenta pelo usuário, o algoritmo de associação executa e realiza o cálculo do suporte para a base referente. A base de dados é analisada varrendo-se as tuplas (um registro em uma base de dados contendo atributos e valores), isto é, a transação (conjunto de itens frequentes identificados), e para cada transação verifica-se os candidatos suportados pela base, ou seja, a ocorrência de um atributo é contabilizada e armazenada para criar uma regra de associação;

- **Confiança**: é o percentual de ocorrência de uma regra. Uma regra de associação é uma regra na forma X => Y, onde X e Y são conjuntos de itens[12]. Lê-se: se ocorre X então Y tende a ocorrer também. O cálculo para descobrir essa métrica é feito da seguinte maneira: o número de registros contendo X e Y dividido pelo número de registros contendo X [18].

A tarefa de associação foi introduzida por [19] e tem a finalidade de determinar os grupos de itens que tendem a ocorrer ao mesmo tempo, em uma mesma transação, gerando-se as regras de associação. Elas podem ser vistas como regras do tipo SE-ENTÃO integrada a duas medidas de interesse: *confiança* e *suporte*. A primeira medida corresponde à probabilidade condicional e a segunda corresponde à fração que sustenta a regra. Ambas serão definidas mais adiante.

A regra de associação é um relacionamento X (antecedente) ⇒ Y (consequente), onde X e Y são conjuntos de itens da transação e a interseção X ∩ Y é o conjunto vazio. Cada regra está associada a um Fator de Suporte Superior "Fs" (medida de interesse *suporte*) e a um Fato de Confiança "Fc" (medida de interesse *confiança*). Seguem as fórmulas dos dois fatores:

$$Fs = \frac{|X \cup Y|}{N}, \text{ onde N é o número total de tuplas} \qquad Fc = \frac{|X \cup Y|}{X}$$

O fator de suporte[1] pode ser descrito como a probabilidade de uma transação qualquer satisfazer tanto X como Y, ao passo que o fator de confiança[2] é a probabilidade de que uma transação satisfaça Y, dado que ela satisfaça X. A tarefa de descobrir regras de associação consiste em extrair do banco de dados todas as regras com "Fs" e "Fc" maiores ou iguais a um "Fs" e "Fc" especificado pelo analista.

A descoberta de regras de associação segue normalmente em dois passos. Primeiramente, o algoritmo determina todos os conjuntos de itens que têm "Fs" maior ou igual ao "Fs" especificado pelo analista. Estes conjuntos são chamados conjuntos de {Itens Frequentes}. Em seguida, todas as possíveis regras candidatas são geradas e testadas para cada conjunto de {Itens Frequentes} com relação ao "Fc". Apenas as regras candidatas com "Fc" maior ou igual ao "Fc" especificado pelo analista são dadas como saída do algoritmo.

[1] O numerador se refere ao número de transações em que X e Y ocorrem simultaneamente e o denominador ao total de transações.
[2] O numerador se refere ao número de transações em que X e Y ocorrem simultaneamente e o denominador se refere à quantidade de transações em que o item X ocorre.

Capítulo 4

Resultados e Discussões

Esta seção apresenta a execução de cada um dos algoritmos de MD do aprendizado supervisionado e aprendizado não supervisionado utilizados para a construção do estudo de caso deste trabalho. Para cada um dos algoritmos executados nesta seção, utilizou-se uma quantidade considerável de dados entre 700 mil e 4 milhões de tuplas, observando a capacidade das máquinas em uso, e o comportamento de cada algoritmo da ferramenta WEKA.

Para alguns algoritmos, tais como o: J48, Random Forest, Apriori e Filtered Associator, foram utilizados uma base de dados contendo 700 mil tuplas devido à alta requisição dos recursos das máquinas, como também o funcionamento desses algoritmos para gerar as regras.

4.1 Execução dos algoritmos de Mineração de Dados

Para realizar os experimentos os dados foram selecionados e transformados para o tipo de dados adequado dos algoritmos, tais dados são do tipo nominal, ou seja atributos multivalorados. A execução foi inicialmente realizada com a configuração *Cross-validation,* com 10 *folds* pois estudos comprovam que esse número é bastante eficiente [20], sendo esta configuração uma opção de análise mais criteriosa. Os dados são executados k vezes (10 vezes, indicado por 10 *folds*) utilizando-se 90% da base de dados completa o que não é muito longe de estar utilizando 100% da base. Utilizar um número maior que 10 implica em consumo de mais recursos para execução de cada algoritmo

A configuração *Cross-validation,* não apresenta variações de performance se comparado as configurações de treinamento e teste pois executa uma análise de treinamento e teste dos dados respectivamente. A performance para o treinamento e teste por sua vez apresentam variações pois a base de dados é dividida. Considerando uma base de dados e a utilização de 90% dos dados para treinamento e os 10% restantes para teste, haverá uma variação na performance do algoritmo uma vez que a base de teste é pequena.

De acordo com os tipos de configurações disponíveis na ferramenta WEKA e com o objetivo de analisar a performance desses algoritmos observou-se a importância de utilizar as três configurações da ferramenta sendo elas: *Cross-validation*, *Training set e Percentage Split*, como ilustra a Figura 8.

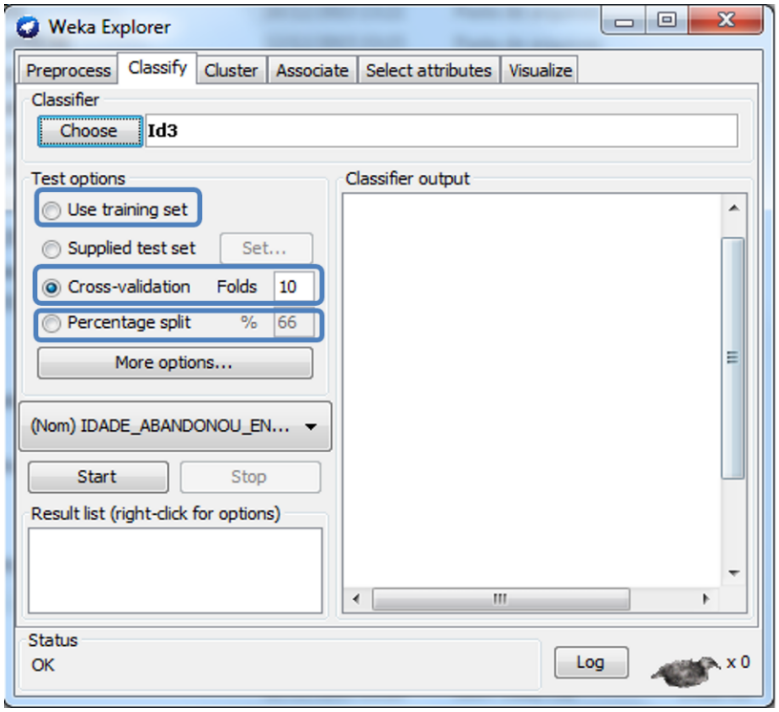

Figura 9: Configurações Utilizadas para Execução dos Algoritmos

A opção *Cross-validation* funciona dividindo o conjunto de dados em subconjuntos mutuamente exclusivos de acordo com número de sub conjuntos definido na ferramenta, indicada pela opção *Folds*, como mostra a Figura 9. A *Cross-validation* é uma técnica que avalia a capacidade de generalização de um modelo. A premissa básica desta técnica é dividir os dados em conjuntos mutuamente exclusivos, ou seja, garante que a quantidade de dados é a mesma em cada subconjunto criado e as informações não se repitam. Posteriormente alguns dos subconjuntos (partição com dados para treinamento) são utilizados para estimar os parâmetros do modelo, e os demais subconjuntos (partição com dados para teste) são utilizados para validar o modelo, este processo é realizado *n* vezes de acordo com o que foi definido na opção *Folds* da ferramenta, e ao final das iterações é calculada a acurácia dos erros encontrados obtendo um resultado mais confiável do modelo.

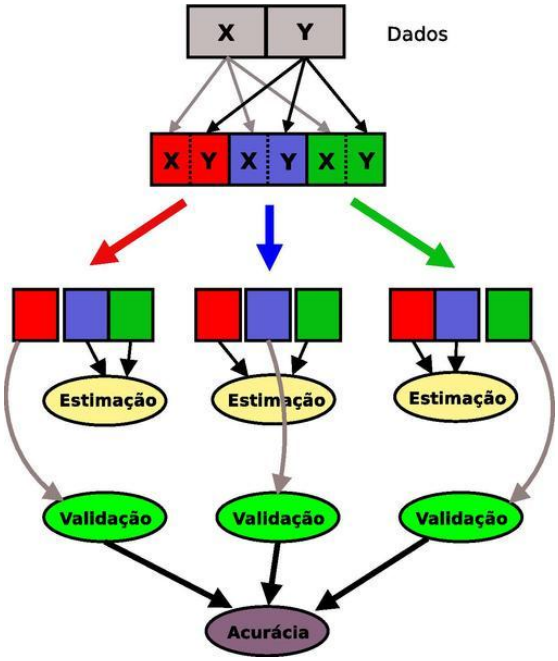

Figura 10: Funcionamento *Cross-validation*

As opções *Training set (conjunto de treinamento)* e *Percentage Split (conjunto de teste)* são configurações que representam um indicador do que vai ser observado posteriormente com a análise dos algoritmos. Para tais conjuntos utilizam-se os dados completo e uma porcentagem dos dados, respectivamente, que deve ser indicado na configuração percentage split da ferramenta. No conjunto de treinamento o algoritmo descobre ou identifica as regras de classificação enquanto que no conjunto de teste é utilizado para verificar as regras descobertas através do conjunto de treinamento.

4.1.1 Execução dos Algoritmos do AS

Nesta seção foram executados os algoritmos do aprendizado supervisionado utilizando uma base com dados reais do ENEM do ano 2013 a fim de analisar o desempenho de cada algoritmo utilizado neste trabalho. Os tipos de dados que os algoritmos interpretam são do tipo nominal (descreve os valores que pode representar), sendo assim a base foi modificada na etapa de pré-processamento do processo KDD. A modificação realizada na base de dados teve como objetivo substituir os campos vazios (null) pelo caractere “?”, pois o algoritmo consegue ser mais eficiente em interpretar e melhor classificar os campos vazios definidos por “?”.

Esses algoritmos foram avaliados quanto ao tempo, acurácia *(instâncias classificadas corretamente)*, taxa de erro (*instâncias classificadas incorretamente)*, matriz de confusão e estatística Kappa. A WEKA oferece as opções de aprendizado dos algoritmos conhecidas como: conjuntos de treinamento e conjunto de testes, tais opção são identificados pelas configurações *Cross-validation, Training set e Percentage Split*, que são utilizadas para avaliar as métricas definidas na seção 3.3. Essas configurações são utilizadas com o objetivo de verificar a construção da árvore e testar o modelo a fim de determinar se o conceito foi aprendido, respectivamente.

As configurações da máquina utilizada para realização dos testes foram: processador Intel core i5-4570S com 2.90 GHz, sistema operacional Windows versão de 64 bits, e memória física de 8 Gigabytes. Para auxiliar a execução de cada algoritmo em termos de performance fez-se necessário ajustar a memória virtual da ferramenta WEKA para 5 Gigabytes, o arquivo modificado foi o *RunWeka.ini* e está localizada no diretório do WEKA.

Algoritmo OneR

A execução deste algoritmo foi efetuada utilizando-se as opções de configuração: *Cross-validation, Training Set e Percentage Split* da ferramenta WEKA. Para este algoritmo foi selecionada uma base de dados contemplando 4 milhões de tuplas com 33 atributos, tendo como atributo classificador o tipo de sexo do candidato, indicado pelo atributo TP_SEXO.

O *OneR* apresentou resultados de acordo com as métricas definidas na seção 3.3.2. A taxa de acurácia e taxa de erro são porcentagens de instâncias testadas correta e incorretamente respectivamente, enquanto que a matriz de confusão apresenta o atributo classificador da classe prevista dentro de um conjunto de dados real.

Na tabela 7 o OneR é executado na configuração *Cross-validation* e as informações dadas mais atenção nesse trabalho foram a matriz de confusão e as classes classificadas correta (*Correctly Classified Instances*) e incorretamente (*Incorrectly Classified Instances*) e o tempo necessário de execução pois tais números nos informam a precisão do modelo. Na matriz de confusão os resultados representam classes que foram classificadas de maneira correta (apresentada na diagonal principal) e classe classificadas incorretamente (representada na diagonal secundária).

Para o algoritmo executado a ferramenta de aprendizado de máquina gerou as informações do *Run information que* indicam o algoritmo utilizado, neste caso é o OneR, o nome da relação (base de dados) que foi utilizada, dadosEnem, o número de instâncias da base

de dados, 4000001, a quantidade de atributos que são 33, o tempo para construir o modelo e o tipo de configuração utilizado para analisar os dados no *Cross-validation* como indicado na Tabela 7. As informações em *Stratified cross-validation* indicam estimativas de performance para o algoritmo de regas em questão. Essas são informações simplificadas sobre a acurácia do modelo. Da matriz de confusão é possível observar que 1170402 instâncias da classe M foram classificadas para a classe F, e 479481 instâncias da classe F foram classificadas para a classe M. A matriz gera percentuais de classes classificadas corretamente sendo 58,75%, e 41,24% para classes incorretamente classificadas.

```
=== Run information ===
Scheme:weka.classifiers.rules.OneR -B 6
Relation:     dadosEnem
Instances:    4000001
Attributes:   33

Time taken to build model: 4.29 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances     2350118               58.7529 %
Incorrectly Classified Instances   1649883               41.2471 %
Kappa statistic                    0.1005
Mean absolute error                0.4125
Root mean squared error            0.6422
Relative absolute error            84.7455 %
Root relative squared error        130.1887 %
Total Number of Instances          4000001


=== Confusion Matrix ===

      a       b   <-- classified as
 503612 1170402 |      a = M
 479481 1846506 |      b = F
```

Tabela 7: Execução do algoritmo OneR aplicado aos dados do ENEM 2013 utilizando a configuração *Cross-validation.*

Para o algoritmo executado a ferramenta de aprendizado de máquina gerou alguns resultados. As informações do *Run information* indicam o algoritmo utilizado, neste caso é o OneR, o nome da relação (base de dados) que foi utilizada, dadosEnem, o número de instâncias da base de dados, 4000001, a quantidade de atributos sendo 33, e o tempo para construir o modelo e o tipo

de configuração utilizado para analisar os dados, o *Percentage Split* utilizando 40% dos dados totais, o que totaliza em 2400001 números de instâncias como indicado na Tabela 8. As informações em *Evaluation on test split* indicam estimativas de performance para o algoritmo de regas em questão. São informações simplificadas sobre a acurácia do modelo. Da matriz de confusão é possível observar que 702750 instâncias da classe M foi classificada para a classe F, e 288222 instâncias da classe F foi classificada para a classe M. A matriz gera percentuais de classes classificadas corretamente sendo 58,70%, e 41,29% para classes incorretamente classificadas.

```
=== Run information ===
Scheme:weka.classifiers.rules.OneR -B 6
Relation:     dadosEnem
Instances:    4000001
Attributes:   33

Test mode:split 40.0% train, remainder test

Time taken to build model: 4.69 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances     1409029               58.7095 %
Incorrectly Classified Instances    990972               41.2905 %
Kappa statistic                      0.0996
Mean absolute error                  0.4129
Root mean squared error              0.6426
Relative absolute error             84.835  %
Root relative squared error        130.2564 %
Total Number of Instances          2400001


=== Confusion Matrix ===

      a       b   <-- classified as
 301704  702750 |       a = M
 288222 1107325 |       b = F
```

Tabela 8: Execução do algoritmo *OneR* aplicado aos dados do ENEM 2013 utilizando a configuração *Percentage Split*.

O OneR utilizou 4000001 instâncias, 33 atributos, o tipo de configuração utilizado para analisar os dados foi o *Training Set*, como indicado na Tabela 9. A matriz de confusão gerou alguns dados sendo possível observar que 702750 instâncias da classe M foi classificada para a classe F, e 288222 instâncias da classe F foi classificada para a classe M. A matriz gera percentuais de classes classificadas corretamente sendo 58,75%, e 41,24% para classes incorretamente classificadas.

```
=== Run information ===
Scheme:weka.classifiers.rules.OneR -B 6
Relation:     dadosEnem
Instances:    4000001
Attributes:   33

Time taken to build model: 4.46 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances      2350118               58.7529 %
Incorrectly Classified Instances    1649883               41.2471 %
Kappa statistic                     0.1005
Mean absolute error                 0.4125
Root mean squared error             0.6422
Relative absolute error            84.7455  %
Root relative squared error       130.1887 %
Total Number of Instances          4000001


=== Confusion Matrix ===

      a       b   <-- classified as
 301704  702750 |       a = M

288222 1107325 |       b = F
```

Tabela 9: Execução do algoritmo *OneR* aplicado aos dados do ENEM 2013 utilizando a configuração *Training Set*.

Algoritmo ZeroR

O algoritmo ZeroR apresenta algumas informações conforme mostra a Tabela 10. As informações apresentadas em *Stratified cross-validation* indicam estimativas de performance. São informações simplificadas sobre a acurácia do modelo, como a matriz de confusão que apresenta a quantidade de instâncias 1674014 da classe M que foram classificadas para a classe F, e 0 instâncias da classe F classificada para a classe M. Esta matriz gera percentuais de classes classificadas corretamente sendo 58,14%, e 41,85% para classes incorretamente classificadas.

```
=== Run information ===
Scheme:weka.classifiers.rules.ZeroR
Relation:       dadosEnem
Instances:      4000001
Attributes:   33

Time taken to build model: 0.54 seconds


=== Stratified cross-validation ===
=== Summary ===

ZeroR predicts class value: F

Correctly Classified Instances      2325987               58.147 %
Incorrectly Classified Instances   1004454               41.8503 %
Kappa statistic                          0
Mean absolute error                      0.4867
Root mean squared error                  0.4933
Relative absolute error                100      %
Root relative squared error            100      %
Total Number of Instances          4000001


=== Confusion Matrix ===

      a       b   <-- classified as
      0 1674014 |       a = M
      0 2325987 |       b = F
```

Tabela 10: Execução do Algoritmo *ZeroR* aplicado aos dados do ENEM 2013 utilizando a configuração *Cross-validation.*

Com a configuração *Training Set* o ZeroR indica na Tabela 11 a quantidade de instâncias executadas, quantidade de instâncias classificadas corretamente e classificadas incorretamente, resumidamente apresentadas pela matriz de confusão. A matriz de confusão apresenta que 1674014 instâncias da classe M foram classificada para a classe F, e 0 instâncias da classe F classificada para a classe M. A matriz gera percentuais de classes classificadas corretamente sendo 58,14%, e 41,85% para classes incorretamente classificadas.

```
=== Run information ===
Scheme:weka.classifiers.rules.ZeroR
Relation:     dadosEnem
Instances:    4000001
Attributes:   33

ZeroR predicts class value: F

Time taken to build model: 0.69 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances     2325987               58.1497 %
Incorrectly Classified Instances   1674014               41.8503 %
Kappa statistic                    0
Mean absolute error                0.4867
Root mean squared error            0.4933
Relative absolute error            100      %
Root relative squared error        100      %
Total Number of Instances          4000001


=== Confusion Matrix ===

      a       b   <-- classified as
      0 1674014 |       a = M
      0 2325987 |       b = F
```

Tabela 11: Execução do Algoritmo *ZeroR* aplicado aos dados do ENEM 2013 utilizando a configuração *Training Set*.

A configuração *Percentage Split* mostra em *Run information* o algoritmo utilizado, o nome da relação (base de dados) que foi utilizada, dadosEnem, o número de instâncias da base de dados, a quantidade de atributos utilizados, o tempo para construir o modelo e o tipo de configuração utilizado para analisar os dados. A Tabela 12 apresenta as estimativas de performance para o algoritmo de regas em questão simplificadas pela matriz de confusão mostrando que 1004454 instâncias da classe M foram classificada para a classe F, e 0 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===
Scheme:weka.classifiers.rules.ZeroR
Relation:       dadosEnem
Instances:      4000001
Attributes:   33

Test mode:split 40.0% train, remainder test

=== Classifier model (full training set) ===
ZeroR predicts class value: F
Time taken to build model: 0.32 seconds

=== Evaluation on test split ===
=== Summary ===
Correctly Classified Instances     1395547              58.1478 %
Incorrectly Classified Instances   1004454              41.8522 %
Kappa statistic                    0
Mean absolute error                0.4867
Root mean squared error            0.4933
Relative absolute error            100%
Root relative squared error        100%
Total Number of Instances          2400001

=== Confusion Matrix ===

      a       b   <-- classified as
      0 1004454 |       a = M
      0 1395547 |       b = F
```

Tabela 12: Execução do Algoritmo *ZeroR* aplicado aos dados do ENEM 2013 utilizando a configuração *Percentage Split*.

Algoritmo ID3

A execução deste algoritmo foi efetuada utilizando-se as opções de configuração *Cross-validation, Training Set e Percentage Split* da ferramenta WEKA, apresentado na Tabela 13, Tabela 14 e Tabela 15 respectivamente. Para o algoritmo em questão foi selecionado uma base de dados contemplando 700 mil tuplas com 33 atributos, e seu atributo classificador sendo TP_SEXO. Para este algoritmo foi necessário reduzir a base de dados devido ao alto consumo de recursos para realizar a execução. Isso ocorre com o ID3 pois a implementação do seu algoritmo constrói várias árvores ao longo de sua execução a fim de gerar uma regra ao final, o que não acontece com os algoritmos de regra OneR e ZeroR.

A Tabela 13 apresenta a quantidade de instâncias da base de dados utilizadas, sendo 1003724, e 33 atributos. Na Tabela 13 é possível observar a performance de maneira simplificada pela matriz de confusão observando que 184786 instâncias da classe M foi classificada para a classe F, e 184400 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===

Scheme:weka.classifiers.trees.Id3
Relation:     dadosEnem
Instances:    1003724
Attributes:   33

Time taken to build model: 68.21 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      449445               44.7777 %
Incorrectly Classified Instances    369186               36.7816 %
Kappa statistic                      0.071
Mean absolute error                  0.4506
Root mean squared error              0.6696
Relative absolute error            113.6721%
Root relative squared error        150.4798 %
UnClassified Instances               185093             18.4406 %
Total Number of Instances          1003724


=== Confusion Matrix ===

     a       b   <-- classified as
```

```
154847   184786 |      a = M
184400   294598 |      b = F
```

Tabela 13: Execução do algoritmo *ID3* aplicado aos dados do ENEM 2013 utilizando a configuração *Cross-validation*.

A execução em *Training set* apresentou melhores taxas para as classes corretas e incorretas indicam a performance do classificador. A estatística kappa comprovado com o valor de 0.9, representando um valor substancial de acordo com a tabela da seção 3.3.1. Obteve-se este resultado pois os dados estão sendo treinados (e essa configuração não criteriosa como a *Cross-validation*) pelo algoritmo e o algoritmo não calcula as instâncias não classificada (UnCLassified Instances) para este tipo de treinamento. Porém obter tais resultados não é uma regra.

Para o algoritmo executado a ferramenta de aprendizado de máquina gerou alguns resultados. As informações do *Run information* indicam o algoritmo utilizado, neste caso é o ID3, o nome da relação (base de dados) que foi utilizada, dadosEnem, o número de instâncias da base de dados, 1003724, a quantidade de atributos, que são 33. O tempo para construir o modelo e o tipo de configuração utilizado para analisar os dados, o *Training Set* como indicado na Tabela 14. As informações em, *Evaluation on test split*, indicam estimativas de performance para o algoritmo de regas em questão. São informações simplificadas sobre a acurácia do modelo. Da matriz de confusão é possível observar que 1024 instâncias da classe M foi classificada para a classe F, e 3320 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===

Scheme:weka.classifiers.trees.Id3
Relation:     dadosEnem
Instances:    1003724
Attributes:   33

Time taken to build model: 68.21 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances      999380               99.5672 %
Incorrectly Classified Instances    4344                 0.4328 %
Kappa statistic                     0.9911
Mean absolute error                 0.0048
```

```
Root mean squared error                  0.0492
Relative absolute error                  0.9946%
Root relative squared error              9.9731 %
Total Number of Instances                1003724


=== Confusion Matrix ===

     a             b   <-- classified as
418459        1024 |     a = M
3320        580921 |     b = F
```

Tabela 14: Execução do algoritmo *ID3* aplicado aos dados do ENEM 2013 utilizando a configuração *Training Set*.

No conjunto de treinamento *Percentage Split* utilizou-se 40% da base de dados completa, resultando em um conjunto com 602234 mil tuplas. Mesmo os dados sendo treinados as instancias classificadas como corretas não obtiveram um resultado tão alto quanto no *Training set* e foram contabilizadas instâncias não classificadas pelos algoritmos.

Para o algoritmo executado a ferramenta de aprendizado de máquina gerou alguns resultados. As informações do *Run information* indicam o algoritmo utilizado, neste caso é o ID3, o nome da relação (base de dados) que foi utilizada, dadosEnem, o número de instâncias da base de dados, 1003724, a quantidade de atributos, que são 33. O tempo para construir o modelo e o tipo de configuração utilizado para analisar os dados, o *Percentage split* como indicado na Tabela 15. As informações em, *Evaluation on test split*, indicam estimativas de performance para o algoritmo de regas em questão. São informações simplificadas sobre a acurácia do modelo. Da matriz de confusão é possível observar que 110061 instâncias da classe M foi classificada para a classe F, e 108971 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===
Scheme:weka.classifiers.trees.Id3
Relation:     dadosEnem
Instances:    1003724
Attributes:   33

Time taken to build model: 67.49 seconds

=== Evaluation on test split ===
=== Summary ===
```

```
Correctly Classified Instances      264524          43.9238 %
Incorrectly Classified Instances    219032          36.3699 %
Kappa statistic                     0.066
Mean absolute error                 0.6718
Root mean squared error             0.6718
Relative absolute error             116.0066 %
Root relative squared error         152.1963 %
UnClassified Instances              118678          19.7063%
Total Number of Instances           602234

=== Confusion Matrix ===

     a       b   <-- classified as
 90330  110061 |     a = M
108971  174194 |     b = F
```

Tabela 15: Execução do algoritmo *ID3* aplicado aos dados do ENEM 2013 utilizando a configuração *Percentage Split.*

Algoritmo J48

A execução do J48 foi efetuada utilizando-se também as opções de configuração *Cross validation, Training set e Percentagem Split* configurado para usar 40% da ferramenta WEKA. A base contempla 711438mil tuplas com 33 atributos, e seu atributo classificador escolhido foi tipo de sexo identificado como, indicado pelo atributo TP_SEXO. Para as configurações da ferramenta, esse algoritmo não utilizou uma base com 4 milhões de tuplas devido ao alto desempenho requisitado à máquina, desempenho este não suportado para as configurações da máquina usada para realizar os experimentos.

A Tabela 16 indicam estimativas de performance para o algoritmo J48 de maneira simplificada pela matriz de confusão, indicando que 195153 instâncias da classe M foram classificadas para a classe F, e 79530 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:       dadosEnem
Instances:      711438
Attributes:   33
```

```
Number of Leaves  :  96857
Size of the tree :  102885

Time taken to build model: 34.17 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      436755               61.3905 %
Incorrectly Classified Instances    274683               38.6095 %
Kappa statistic                      0.1667
Mean absolute error                  0.4592
Root mean squared error              0.4951
Relative absolute error             94.058  %
Root relative squared error        100.2009 %
Total Number of Instances          711438


=== Confusion Matrix ===

     a      b   <-- classified as
 106006 195153 |      a = M
  79530 330749 |      b = F
```

Tabela 16: Execução do algoritmo *J48* aplicado aos dados do ENEM 2013 utilizando a configuração *Cross-validation*.

A execução deste algoritmo para a configuração *Training set* gera como resultados instâncias corretas e incorretas, indicando a performance do classificador, a estatística kappa, matriz de confusão e também exibe o número de folhas, ou seja a quantidade de nós criados para a construção da árvore, sabendo que apenas este algoritmo exibe uma árvore gráfica na ferramenta WEKA. Os números da matriz de confusão indicam o quão exato foi a classificação da classe escolhida tal como indicam esses números: 176321 instâncias da classe M foi classificada para a classe F, e 62896 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     dadosEnem
Instances:    711438
Attributes:   33

Number of Leaves  :  96857
```

```
Size of the tree :        102885


Time taken to build model: 32.31 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances      472221               66.3756 %
Incorrectly Classified Instances    239217               33.6244 %
Kappa statistic                      0.275
Mean absolute error                  0.4275
Root mean squared error              0.4623
Relative absolute error             87.5575 %
Root relative squared error         93.5637 %
Total Number of Instances           711438


=== Confusion Matrix ===

     a         b   <-- classified as
 124838 176321 |      a = M
62896 347383 |      b = F
```

Tabela 17: Execução do algoritmo *J48* aplicado aos dados do ENEM 2013 utilizando a configuração *Training Set*.

A execução J48 na configuração *Percentage Split* utilizando 40% da base de dados o que significa treinar 426863 tuplas, gera como resultados instâncias corretas e incorretas, indicando a performance do classificador, a estatística kappa, matriz de confusão e também exibe o número de folhas, ou seja a quantidade de nós criados para a construção da árvore, sabendo que apenas este algoritmo exibe uma árvore gráfica na ferramenta WEKA.

Após execução utilizando 1003724 instâncias com 33 atributos, o tempo para construir o modelo e o tipo de configuração utilizado para analisar os dados na configuração de *Percentage split* como indicado na Tabela 18, as estimativas de performance para o algoritmo são informações simplificadas sobre a acurácia do modelo conforme mostra a matriz de confusão indicando que 119212 instâncias da classe M foi classificada para a classe F, e 46941 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===
Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     dadosEnem
Instances:    711438
Attributes:   33

Number of Leaves  :  96857
Size of the tree :        102885

Time taken to build model: 33.11 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances      260710               61.0758 %
Incorrectly Classified Instances    166153               38.9242 %
Kappa statistic                      0.1586
Mean absolute error                  0.4606
Root mean squared error              0.4971
Relative absolute error             94.3429 %
Root relative squared error        100.5926 %
Total Number of Instances          426863

=== Confusion Matrix ===

     a         b   <-- classified as
 61706 119212 |      a = M
 46941 199004 |      b = F
```

Tabela 18: Execução do algoritmo *J48* aplicado aos dados do ENEM 2013 utilizando a configuração *Percentage Split*.

Algoritmo Random Forest

Para a execução do Random Forest utilizou-se uma base de dados com 50 mil tuplas e 33 atributos. Essa quantidade de dados foi selecionada devido à alta requisição de recursos de memória pois o processamento da máquina utilizada para o experimento não suportou a execução de uma base com 4 milhões. O algoritmo em questão requer mais processamento pelo fato de sua estrutura construir várias árvores de decisão de maneira aleatória, dentro da base de dados para gerar as regras ao final da execução. Para esta execução utilizou nas configurações de *Cross-validation, Training set* e *Percentage Split* a construção de 10 árvores, valor default

da ferramenta WEKA. A escolha para o algoritmo construir as 10 árvores ocorreu em virtude da máquina não uportou a alta requisição de processamento.

As informações descritas em Run information descrevem o algoritmo utilizado, o nome da relação (base de dados), a quantidade de instâncias e a quantidade de atributos respectivamente. O tipo da configuração também é apresentado e indicada por Test mode. O tempo para execução do Random Forest é estimado em 3,06 segundos para a construção do modelo. É possível observar que o mesmo foi configurado para construir 10 árvores dentro da base utilizada, assim como foram selecionadas de maneira aleatória 6 características dos atributos dentre os 33.

Este algoritmo calcula um valor indicado por Out of bag, que estima a generalização do erro. Tal valor é gerado de baseado nos subconjuntos de árvores que são criados ao longo da execução. Conforme os algoritmos anteriores este também gera uma matriz de confusão, com valores 13035 instâncias da classe M foi classificada para a classe F, e 8617 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===
Scheme:weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1
Relation:     dadosEnem
Instances:    50000
Attributes:   33

Test mode:10-fold cross-validation

=== Classifier model (full training set) ===

Random forest of 10 trees, each constructed while considering 6 random features.
Out of bag error: 0.4571

Time taken to build model: 3.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       28348               56.696  %
Incorrectly Classified Instances     21652               43.304  %
Kappa statistic                          0.1064
Mean absolute error                      0.4724
Root mean squared error                  0.5006
```

Relative absolute error	95.603 %
Root relative squared error	100.7172 %
Total Number of Instances	50000

=== Confusion Matrix ===

a	b	<-- classified as
9265	13035	a = M
8617	19083	b = F

Tabela 19: Execução do algoritmo *Random Forest* aplicado aos dados do ENEM 2013 utilizando a configuração *Cross validation*.

As informações descritas em Run information descrevem o algoritmo utilizado, o nome da relação (base de dados), a quantidade de instâncias e a quantidade de atributos respectivamente. O tipo da configuração também é apresentado e indicada por Test mode. O tempo para execução do Random Forest é estimado em 3,48 segundos para a construção do modelo. É possível observar que o mesmo foi configurado para construir 10 árvores dentro da base utilizada, assim como foram selecionadas de maneira aleatória 6 características dos atributos dentre os 33.

Este algoritmo calcula um valor indicado por Out-of-bag, que estima a generalização do erro. Tal valor é gerado de baseado nos subconjuntos de árvores que são criados ao longo da execução. Conforme os algoritmos anteriores este também gera uma matriz de confusão, com valores 312 instâncias da classe M foi classificada para a classe F, e 146 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===

Scheme:weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1
Relation:     dadosEnem
Instances:    50000
Attributes:   33

Test mode:evaluate on training data

=== Classifier model (full training set) ===

Random forest of 10 trees, each constructed while considering 6 random features.
Out of bag error: 0.4571

Time taken to build model: 3.48 seconds

=== Evaluation on training set ===
=== Summary ===

Correctly Classified Instances       49542               99.084  %
Incorrectly Classified Instances       458               0.916  %
Kappa statistic                          0.9815
Mean absolute error                      0.1826
Root mean squared error                  0.2157
Relative absolute error                 36.9423 %
Root relative squared error             43.3993 %
Total Number of Instances            50000

a     b   <-- classified as
 21988   312 |     a = M
   146 27554 |     b = F
```

Tabela 20: Execução do algoritmo *Random Forest* aplicado aos dados do ENEM 2013 utilizando a configuração *Training set.*

As informações descritas em *Run information* descrevem o algoritmo utilizado, o nome da relação (base de dados), a quantidade de instâncias e a quantidade de atributos respectivamente. O tipo da configuração também é apresentado e indicada por Test mode. O tempo para execução do Random Forest é estimado em 3,98 segundos para a construção do modelo. É possível observar que o mesmo foi configurado para construir 10 árvores dentro da base

utilizada, assim como foram selecionadas de maneira aleatória 6 características dos atributos dentre os 33.

Este algoritmo calcula um valor indicado por Out of bag, que estima a generalização do erro. Tal valor é gerado de baseado nos subconjuntos de árvores que são criados ao longo da execução. Conforme os algoritmos anteriores este também gera uma matriz de confusão, com valores 8013 instâncias da classe M foi classificada para a classe F, e 5203 instâncias da classe F foi classificada para a classe M.

```
=== Run information ===

Scheme:weka.classifiers.trees.RandomForest -I 10 -K 0 -S 1
Relation:     dadosEnem
Instances:    50000
Attributes:   33

Test mode:split 40.0% train, remainder test

=== Classifier model (full training set) ===

Random forest of 10 trees, each constructed while considering 6 random features.
Out of bag error: 0.4571

Time taken to build model: 3.98 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances       16784               55.9467 %
Incorrectly Classified Instances     13216               44.0533 %
Kappa statistic                          0.0909
Mean absolute error                      0.476
Root mean squared error                  0.5047
Relative absolute error                 96.3324 %
Root relative squared error            101.4925 %
Total Number of Instances            30000

a     b   <-- classified as
 5408  8013 |     a = M
 5203 11376 |     b = F
```

Tabela 21: Execução do algoritmo *Random Forest* aplicado aos dados do ENEM 2013 utilizando a configuração *Percentage split*.

4.1.2 Execução dos Algoritmos do ANS

Nesta seção é apresentada a execução dos algoritmos do Aprendizado Não Supervisionado utilizando uma base com dados reais do ENEM do ano 2013 a fim de analisar o desempenho de cada algoritmo utilizado neste trabalho. Assim como para a execução dos algoritmos do Aprendizado Supervisionado. Os tipos de dados que os algoritmos interpretam são também do tipo nominal (descreve os valores que pode representar).

Esses algoritmos foram avaliados quanto ao suporte e confiança. Esses parâmetros analisam o percentual de vezes em que o conjunto de dados aparece no conjunto de transação, e o percentual de ocorrência de uma regra respectivamente.

Algoritmo Apriori

Para o algoritmo em questão foi selecionado a base de dados contemplando 4 milhões de tuplas contendo 33 atributos. A Tabela 20 mostra os resultados após execução do Apriori. O suporte configurado neste algoritmo é de 85% e uma confiança de 60%, e algumas regras foram geradas indicando confianças de 91%, 86%, 89% e 100% respectivamente. Algumas regras foram selecionadas, dentre as 30 regras escolhidas para a execução deste algoritmo, para melhor entendimento das regras que este algoritmo gerou. Para as regras abaixo o algoritmo criou as regras com confiança de 100% para candidatos que não tem alguma deficiência e pretendem concorrer a alguma bolsa de estudos, eles não são candidatos idosos.

Os resultados gerados para essa execução foram modificados na configuração da ferramenta para que fossem geradas as 30 melhores regras. Para a regra de número 3 com confiança de 89%, os candidatos que não são idosos e não possuem algum tipo de deficiência moram em áreas urbanas, representado pela letra B. A regra com confiança de 86% apresentou que candidato do ENEM que não são idosos e não tem alguma deficiência pretendem concorrer a alguma bolsa de estudos.

Assim como os algoritmos de classificação o WEKA gera algumas informações para o Apriori. Os dados de execução indicam qual o algoritmo escolhido para a análise dos dados, o nome da relação, a quantidade de instâncias e a quantidade de atributos, como mostra a última parte da tabela 22.

```
=== Run information ===

Scheme:       weka.associations.Apriori -N 30 -T 0 -C 0.6 -D 0.05 -U 1.0 -M 0.3 -S -1.0 -c -1
Relation:     dadosEnem
Instances:    4000001
Attributes:   33


1.   IN_ACESSO=0
     PRETENDE_RECORER_BOLSA=A 3444979
==> IN_IDOSO=0 3444747    conf:(1)
     IN_ACESSO=0 3998619 ==> IN_IDOSO=0
3998327    conf:(1)

2. IN_IDOSO=0 IN_ACESSO=0 3998327
==> PRETENDE_PARTCIPAR_PROUNI=A 3622017    conf:(0.91)

3. IN_IDOSO=0 IN_ACESSO=0 3998327
==> LOCALIZACAO_RESIDENCIA=B 3558133    conf:(0.89)

4. IN_ACESSO=0 3998619 ==>
IN_IDOSO=0 LOCALIZACAO_RESIDENCIA=B 3558133    conf:(0.89)

5. IN_IDOSO=0 3999703 ==> IN_ACESSO=0 LOCALIZACAO_RESIDENCIA=B 3558133
conf:(0.89)

6. IN_IDOSO=0 IN_ACESSO=0 3998327
==> PRETENDE_RECORER_BOLSA=A 3444747    conf:(0.86)
```

Tabela 22: Regras geradas pelo algoritmo Apriori.

Filtered Associator

A execução deste algoritmo se deu baseado na opção de conjunto completo de treinamento (*full training set*) fornecida pela ferramenta WEKA. Foi selecionado a base de dados contemplando 4 milhões de tuplas contendo 33 atributos.

A Tabela 23 mostra os resultados após execução do Filtered Associator com a configuração do suporte mínimo e a confiança Filtered Associator para esta execução foram de 85% e 90% respectivamente, para as instancias classificadas como corretas. E para obter as dez melhores regras para a base de dados foi necessário executar três vezes *(Number of cycles performed).*

A execução do Filtered Associator foi configurada com o suporte em 85% e a confiança em 90% respectivamente. Configurou-se também a quantidade de regras (as 10 melhores) a

serem geradas pelo algoritmo. Para esta configuração as regras geradas pelo Filtered Associator foram de 100%, como demonstra a regra: para os casos de que o aluno não fez curso EJA ele não é um estudante idoso. E para os candidatos que não apresentam alguma deficiência, frequentaram o ensino regular e os mesmos não são idosos.

1. FEZ_FAZ_EJA=B 3832713 ==> IN_IDOSO=0 3832459 conf:(1) 2. SIT_FUNC_ESC=1 FEZ_FAZ_EJA=B 3830158 ==> IN_IDOSO=0 3829904 conf:(1) 5. IN_ACESSO=0 FREQUENTOU_ENS_REGULAR=A 3942139 ==> IN_IDOSO=0 3941861 conf:(1)

Tabela 23: Regras geradas pelo algoritmo Filtered Associator.

4.2 Comparação e Performance dos Algoritmos do AS e ANS

Esta seção apresenta uma tabela com os resultados das execuções dos algoritmos assim como também um gráfico de cada um deles estudado exibindo as três configurações da ferramenta WEKA. A tabela apresenta os resultados de todos os algoritmos na configuração *Cross-validation.*

Os gráficos abaixo apresentam de maneira clara e simples a performance de cada um dos algoritmos. Em cada gráfico uma legenda informa a quantidade de tuplas da base dados que foi utilizada para executar cada algoritmo.

Para as três configurações abordadas ao longo deste trabalho observa-se que a taxa de acurácia para o OneR a variação foi mínima entre as três configurações. A taxa de acurácia executada em *Percentage Split* apresentou uma acurácia um pouco mais elevada, o que significa que o algoritmo classificou de forma correta os domínios da classe TP_SEXO.

Na Figura 11 o resultado deste algoritmo não apresentou muitas diferenças para cada uma das configurações, e isto pode ser facilmente concluído observando o comportamento do OneR no gráfico. O tempo gasto por este algoritmo é resultado do seu simples funcionamento se comparado aos demais algoritmos, uma vez que este não cria árvores ou faz chamadas

recursivas ao longo da execução. Este algoritmo cria regras a partir da análise de cada atributo. E cada atributo é associado a classe mais frequente.

Era de se esperar que houvesse uma diferenciação no *Percentage Split*, uma vez que foi utilizado 40% da base de dados original (contendo 4 milhões de tuplas), resultando em uma base com 2400001 milhões. Os valores de 4 milhões e 2400001 milhões estão representados numericamente no gráfico como 4 e 2,4 respectivamente pois os valores ficam melhor apresentados no gráfico.

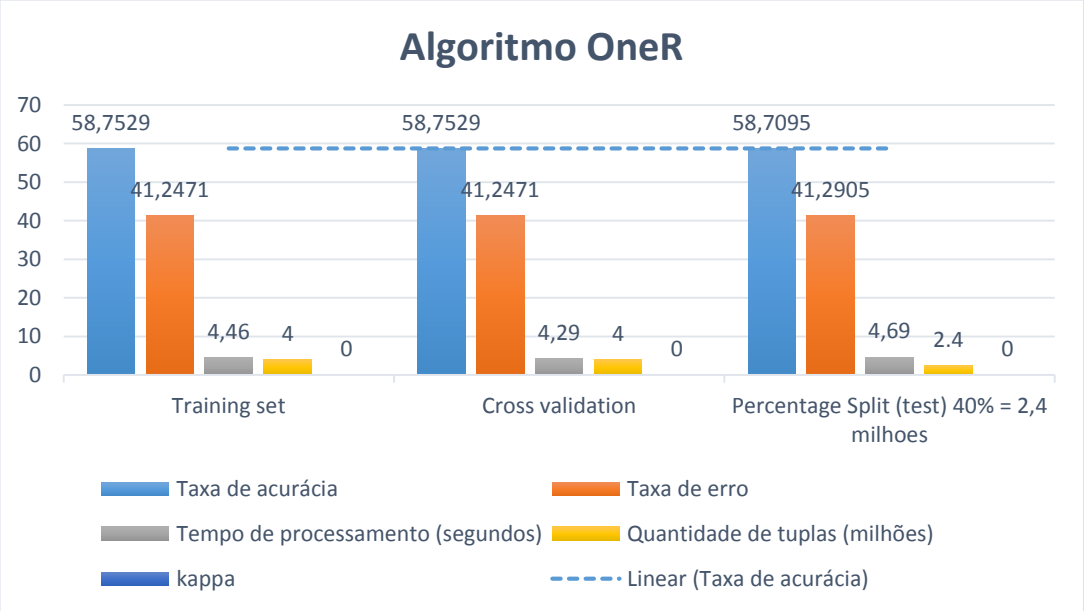

Figura 11 Gráfico com resultados do algoritmo OneR

Na Figura 12 o algoritmo ZeroR apresenta taxas de acurácia semelhantes para a configuração *Training set* e *Cross-validation.* É possível observar que os resultados gerados apresentam tais performances, como a taxa de acurácia e a taxa de erro, devido ao funcionamento do mesmo na forma como ele cria as regras. Seu funcionamento é considerado trivial comparando-o aos demais algoritmos do Aprendizado Supervisionado aqui estudados. Porém o ZeroR apresenta resultado satisfatório considerando o tempo de execução de 0,54, em contrapartida as taxas de acurácia não rederam bons resultados assim como a estatística kappa de 0. Este algoritmo utilizou 4 milhões de dados para as configurações *Cross-validatio*n e *Training set* representado por 4 no gráfico e 2,4 milhões para a configuração de *Percentage Split* representado por 2,4 no gráfico.

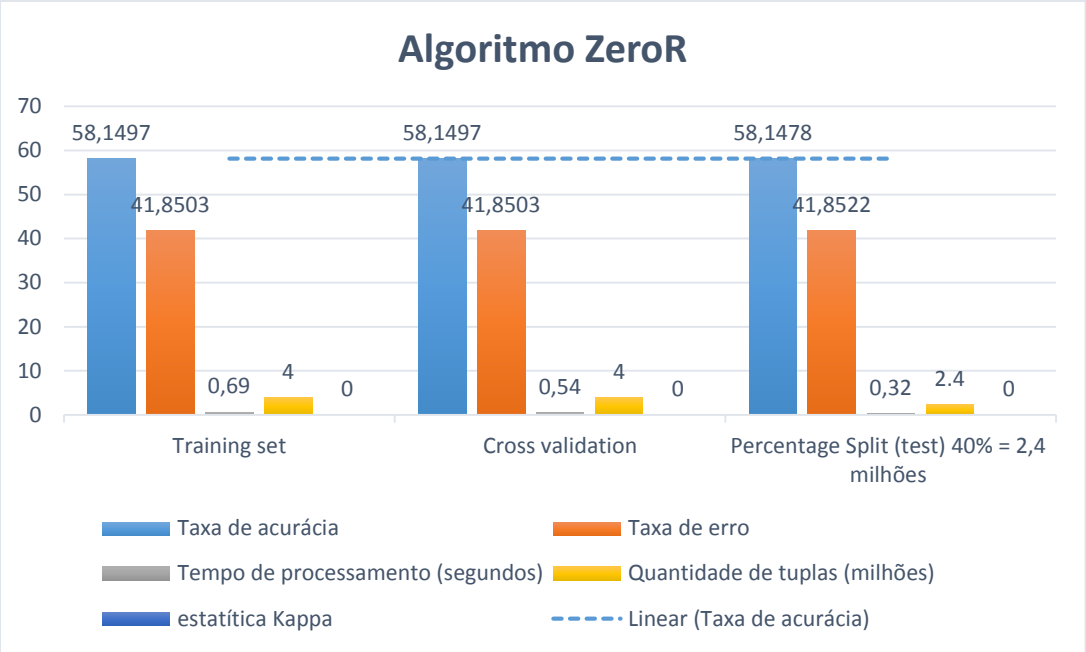

Figura 12: Gráfico com resultados do algoritmo ZeroR

Na Figura 13 os resultados para o ID3 foram distintos para cada configuração utilizada, e isto ocorre principalmente pela forma como ele analisa os dados da base. O tempo de execução apresenta-se diferente se comparado aos demais algoritmos do Aprendizado Supervisionado pelo fato de o ID3 gerar várias árvores de decisão dentro da base de dados ocasionando mais tempo para analisar cada nova árvore gerada. A melhor acurácia se deu com a configuração do *Trainig set* o que justifica tal resultado pois os dados são apenas treinados. Este algoritmo utilizou 1003724 milhões de dados para as configurações *Cross-validatio*n e *Training set* representado por 1,3 no gráfico e 602234 mil para a configuração de *Percentage Split* representado por 0,6 (ou apenas 0 que foi o que o gráfico conseguiu apresentar).

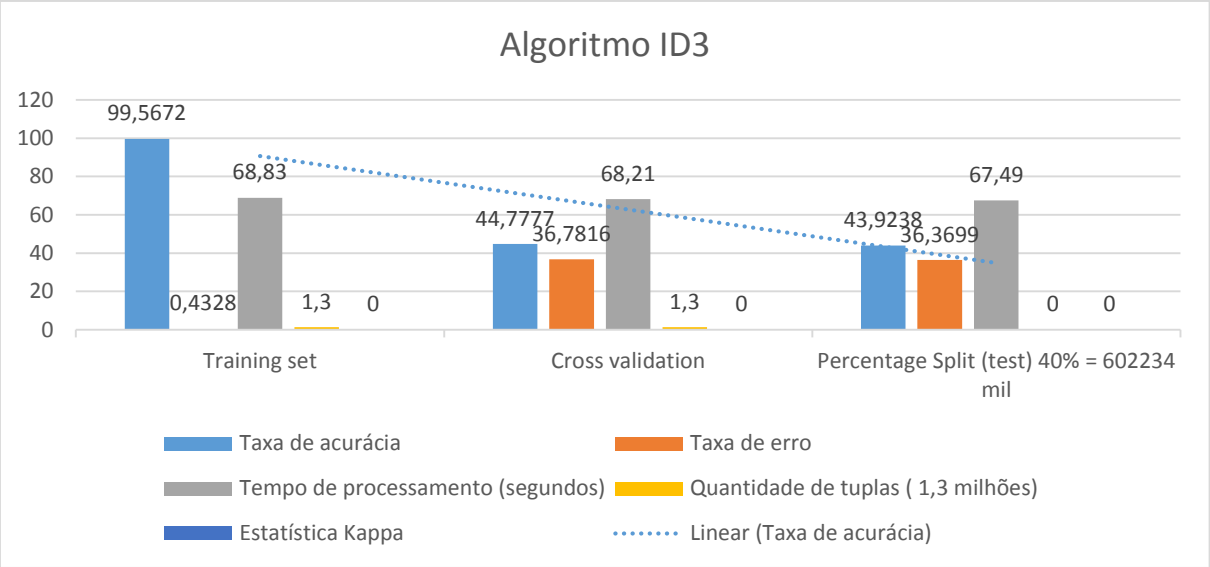

Figura 13: Gráfico com resultados do algoritmo ID3

Na Figura 14 os resultados para o J48 foram um pouco distintos para cada configuração utilizada, e isto ocorre principalmente pela forma como ele analisa os dados da base. Para a base de dados são criadas diversas árvores ao longo de todo o banco de dados, o que requer mais tempo de execução e uma quantidade de tuplas inferior, contendo 700 mil tuplas (representado no gráfico pelo número 7), e 426863 (representado no gráfico pelo número 4,2).

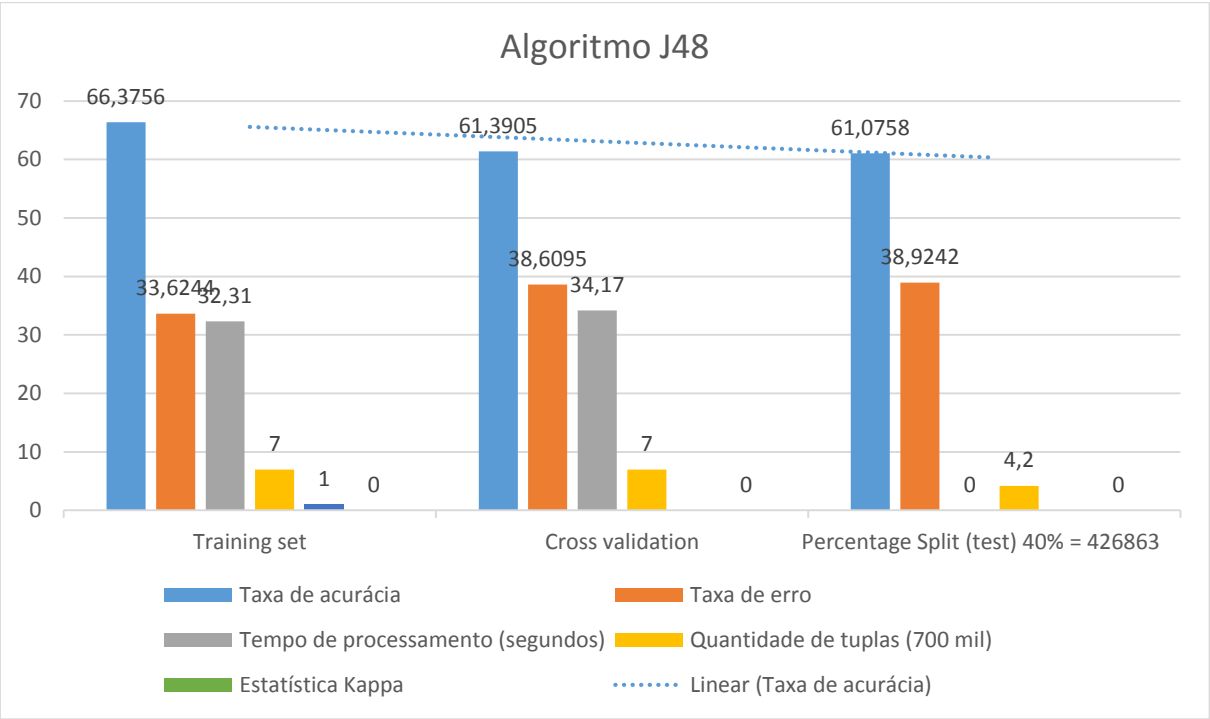

Figura 14: Gráfico com resultados do algoritmo J48

Na Figura 15 os resultados do Random Forest não apresentaram muitas diferenças para cada uma das configurações. Sua execução foi realizada com uma base de dados contendo 50 mil tuplas (representado no gráfico pelo número 5) e no conjunto de *Percentage Split* foram executados 30000 tuplas (representado no gráfico pelo número 3) o que representa 40% da base respectivamente apresentados no gráfico.

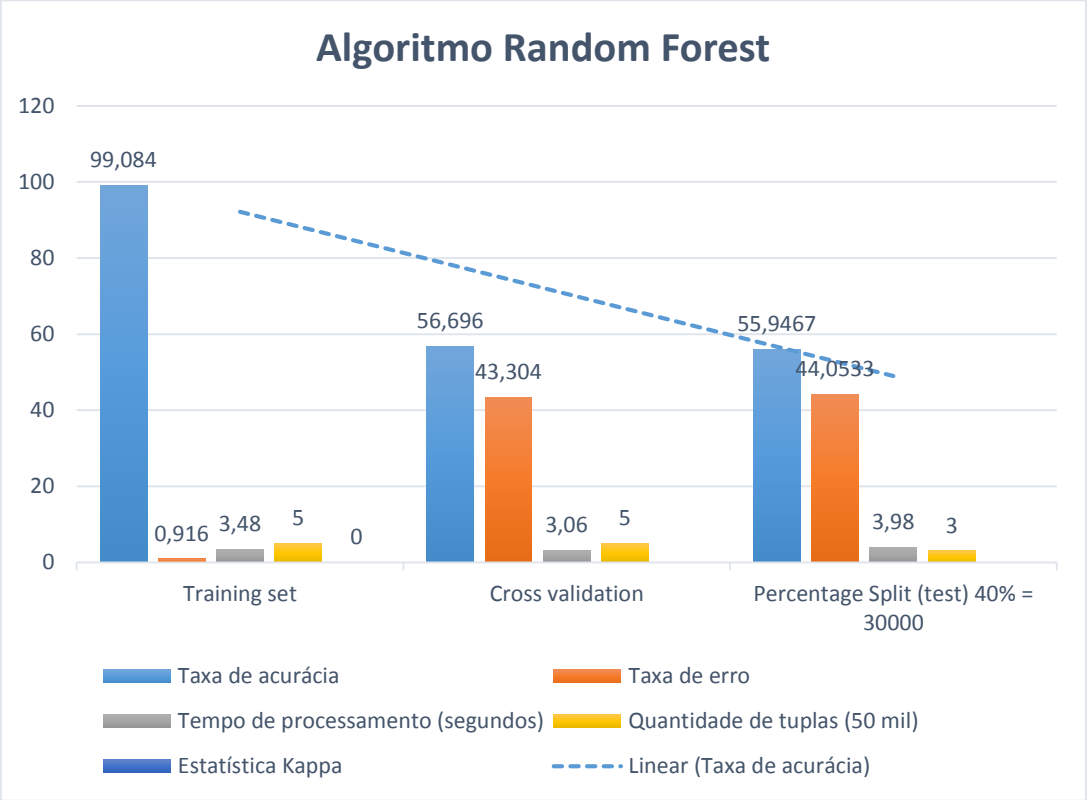

Figura 15: Gráfico com resultados do algoritmo Random Forest

Nas tabelas a seguir são exibidos os resultados de cada um dos algoritmos do Aprendizado Supervisionado utilizando a configuração Cross-validation, Training set e Percentage Split respectivamente.

Aprendizado Supervisionado						
Algoritmo	**Taxa de Acurácia (%)**	**Taxa de Erro (%)**	**Kappa**	**Tempo de Processamento (segundos)**	**Qtd. de tuplas executadas**	**Visualização**
OneR	58,7	41,2	0,10	4,29	4 milhões	--
ZeroR	58,1	41,8	0	0,54	4 milhões	--
ID3	44,7	36,6	0,07	68,21	1,3 milhões	--
J48	61,3	38,6	0,16	34,17	700 mil	Árvore de decisão
Random Forest	56,6	43,3	0,10	3,06	50 mil	--

Tabela 24: Resultado execução dos algoritmos na configuração Cross-validation

Aprendizado Supervisionado						
Algoritmo	**Taxa de Acurácia (%)**	**Taxa de Erro (%)**	**Kappa**	**Tempo de Processamento (segundos)**	**Qtd de Tuplas Executadas**	**Visualização**
OneR	58,7	41,2	0,10	4,46	4 milhões	--
ZeroR	58,1	41,8	0	0,69	4 milhões	--
ID3	99,5	0,43	0,99	68,83	1003724	--
J48	66,3	33,6	0,27	32,31	700 mil	Árvore de decisão
Random Forest	99,0	0,91	0,98	3,48	30 mil	--

Tabela 25: Resultado execução dos algoritmos na configuração Training set

Aprendizado Supervisionado						
Algoritmo	**Taxa de Acurácia (%)**	**Taxa de Erro (%)**	**Kappa**	**Tempo de Processamento (segundos)**	**Qtd de Tuplas Executadas**	**Visualização**
OneR	58,7	41,2	0,09	4,69	4 milhões	--
ZeroR	58,1	41,8	0	0,32	2400000 milhões	--
ID3	43,9	36,3	0,06	68,49	602234 mil	--
J48	61,0	38,9	0,15	33,11	280000 mil	Árvore de decisão
Random Forest	55,9	44,0	0,09	3,9	30 mil	--

Tabela 26: Resultado execução dos algoritmos na configuração Percentage split

As regras geradas pelos algoritmos do Aprendizado não supervisionado demonstram resultados discordantes conforme apresentado na seção 4.1.2 nas tabelas 22 e 23, sendo tais resultados gerados pela ferramenta de acordo com a configuração de cada algoritmo especificado. A melhor regra é identificada pelo analista de acordo com o objetivo que se deseja encontrar, podendo portanto cada algoritmo gerar conhecimento mais satisfatória do que outro.

O desempenho varia de acordo com o funcionamento do algoritmo. Porém o que pode ser levado em consideração para indicar qual o melhor deles é observando a taxa de acurácia de cada um. O algoritmo que apresentar maior acurácia é um bom algoritmo pois apresentará melhores regras, sendo assim melhores resultados. Foi possível entender de acordo com esse estudo que um tempo de processamento pequeno não é inversamente proporcional as regras geradas. O parâmetro mais adequado a se avaliar são a classificação das instâncias, matriz de confusão e estatística kappa. Cada parâmetro possui critérios relevantes para gerar os melhores padrões.

Alguns algoritmos apresentam tempo de processamento baixo e o que se conclui sobre é que a implementação não exige consultas recursivas, como é o exemplo do OneR, ZeroR, mesmo executando uma base com 4 milhões de registros. Tais características não encontradas nos algoritmos ID3 e J48, o que é explicado por suas implementações. Cada um desses

algoritmos de árvores realiza análises recursivas dos dados, gerando árvores e regras a fim de encontrar os padrões (conhecimento).

Capítulo 5

Conclusões

Este trabalho propôs o estudo de algoritmos do aprendizado supervisionado e do aprendizado não supervisionado aplicados a base de dados educacional. Diante do que foi proposto é possível compreender de maneira simples o funcionamento dos algoritmos, assim como também o desempenho de cada um deles de acordo com as características da base apresentada neste trabalho.

O estudo deste trabalho não se propõe a sinalizar qual algoritmo é melhor ou pior, desta forma são estudados o desempenho e performance sob as métricas adotadas para cada um deles. Sobre cada resultado obtido é possível selecionar qual deles se encaixa adequadamente ao contexto que o usuário pretende aplicar o seu estudo.

Após o estudo deste trabalho de conclusão de curso, estabeleceu-se como trabalhos futuros realizar estudo comparativo dos algoritmos utilizando computação distribuída a fim de obter maior eficiência na execução de cada algoritmo. O emprego de processamento sequencial, posto em prática apresentou algumas dificuldades devido aos recursos da máquina disponíveis para análise.

Para cada algoritmo aqui estudado pode-se tirar resultados satisfatórios embora alguns levem mais tempo para realizar o processamento. Com o objetivo de informar as características dos algoritmos estudados, assim sendo o foco não é discutir qual o melhor deles, porém extrair as melhores performances de cada um.

De acordo com o objetivo de estudo deste trabalho não foi aqui apresentado o melhor algoritmo dos sete estudados, foram apresentadas características de cada algoritmo, a performance de cada um deles em diferentes configurações e quantidade de instâncias mostrando o desempenho individualmente. As performances apresentadas ao longo do estudo indicaram diferenças consideráveis e algumas sutis como por exemplo o tempo, alguns tempos de execução foram mais elevados e outros não tão elevados, a quantidade de tuplas executadas pelos algoritmos foram diferentes, assim como a quantidade de regras geradas pelos algoritmos de associação.

Tais resultados foram gerados por determinação de alguns fatores que devem ser considerados no momento da escolha do melhor algoritmo para executar em qualquer base de dados. Assim como a quantidade de dados a ser analisada pelo algoritmo, é preciso observar como cada um deles é implementado pois este fator pode influenciar na performance de tempo de execução por exemplo, critério não ser aconselhável para um experimento em que o tempo de processamento importa.

Referências Bibliográficas

[1] Fayyad, U.; Piatesky-Shapiro, G.; Smyth, P. From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996.

[2] Romero, C., & Ventura, S. (2010). Educational data mining: a review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *40*(6), 601-618.

[3] IEEE International Conference on Data Mining, 2006. Disponível em http://www.cs.uvm.edu/~icdm/algorithms/index.shtml. Acessado em 9 abr. 2015.

[4] Fayyad, U.; Piatesky-Shapiro, G.; Smyth, P. From Data Mining to Knowledge Discovery: An Overview. In: Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996.

[5] Michael J. A. Berry e Gordon S. Linoff, Data Mining Techniques for Marketing Sales and Customer Relationship Management, 3rd Edition

[6] Romero, C. Ventura, S. "Educational Data Mining: A Review of the State of Art", 2010.

[7] Bayer and Yocef. The State of Educational Data Mining in 2009: A Review and Future Visions, 2009.

[8] Mitchell, T. M. Machine Learning.McGraw-Hill, 1997.

[9] Batista, Gustavo Enrique de Almeida Prado Alves. Pré-processamento de Dados em Aprendizado de Máquina Supervisionado, 2003.

[10] Simon O. Haykin. Neural Networks and Learning Machines. 3rd Edition.

[11] Ross Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[12] Ross J. Quinlan: Learning with Continuous Classes. In: 5th Australian Joint Conference on Artificial Intelligence, Singapore, 343-348, 1992.

[13] InformationGain. Disponível em: <http://homes.cs.washington.edu/~shapiro/EE596/notes/InfoGain.pdf>. Acessado em: dez. 2015

[14] Algoritmo Random Forest. Disponível em: <http://randomforest2013.blogspot.com.br/2013/05/randomforest-definicion-random-forests.html>. Acessado em 15 ago. 2015.

[15] Mineração de dados com o WEKA, Parte 2: Classificação e armazenamento em cluster. Disponível em: <http://www.ibm.com/developerworks/br/opensource/library/os-weka2/> Acessado em: 20 nov. 2015.

[16] Cássio Oliveira Camilo e João Carlos da Silva. Mineração de Dados: Conceitos, Tarefas, Métodos e Ferramentas. Disponível em: <http://www.inf.ufg.br/sites/default/files/uploads/relatorios-tecnicos/RT-INF_001-09.pdf> Acessado em: 2 jun. 2015.

[17] VIEIRA, Andreza S. Identificação de Diretrizes para a Construção de Meta-modelos na Infraestrutura de MDA, Campina Grande, Paraíba, 2010.

[18] GOUVEIA, Roberta Macedo Marques. Mineração de dados em data warehouse para sistema de abastecimento de água. Dissertação (Mestrado em Informática) – Universidade Federal da Paraíba, João Pessoa, 2009.

[19] AGRAWAL, R., T. IMIELINSKI, e A. SWAMI. *Mining Association Rules Between Sets of Itens in Large Databases.* Editora: Int. Conf. Management of Data, 1993.

[20] KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: LAWRENCE ERLBAUM ASSOCIATES LTD. International joint Conference on artificial intelligence. [S.l.], 1995. v. 14, p.1137–1145

[21] Rajeev Kumar e Arvind Kalia. A Comparative Study of Association Rule Algorithms for Investment in Related Sector of Stock Market

[22] KANASHIRO, Augusto. *Um data warehouse de publicações científicas: indexação automática da dimensão tópicos de pesquisa dos data marts.* Dissertação de Mestrado, USP - São Carlos, São Paulo, SP, 2007.

[23] REZENDE, Solange O. **Sistemas Inteligentes: Fundamentos e Aplicações**. Barueri, SP: Manole, 2003.

[24] Ian H. Witten et al. 2011. Data Mining. **Pratical Machine Learning Tools and Techniques**.

[25] Romero, C. Ventura, S. “Educational Data Mining: A Review of the State of Art”, 2010.

[26] Aprendizado de máquina: Árvore de Decisão Indutiva. Disponível em:< http://www.cin.ufpe.br/~pacm/SI/ArvoreDecisaoIndutiva.pdf>. Acessado em: 2 set. 2015

[27] LIMA, Tânia Dos Santos. Estudo Comparativo Dos Algoritmos De Classificação Da Ferramenta Weka, 2005.

Apêndice

O script foi desenvolvido utilizando a linguagem PHP. O script faz a limpeza de dados alterando os dados vazios por “?” e seleciona as colunas nas quais se deseja trabalhar. Para que a limpeza dos dados do ENEM foi necessário configurar o limite de leitura dos dados que são realizados em memória devido ao grande volume de dados utilizados, indicados por ini_set e set_limit.

```
ini_set('memory_limit', '-1');
set_time_limit(0);

$infos = array();
$file_name = "DadosEnem.csv";  //O arquivo é informado.

//O arquivo é aberto para leitura.
if (($handle = fopen($file_name, "r")) !== FALSE) {
   echo '<table border="1"><thead>';
   echo '<th>Tipo Escola</th><th>UF Escola </th><th>
Idade</th><th>Nacionalidade</th><th>UF Nacimento</th><th>Situação Conclusão
Estudo</th>;
   echo '</thead>';
   while (($data = fgetcsv($handle, 0, ";")) !== FALSE) {

//Em quanto o arquivo não foi lido completamente, será lido cada coluna e os campos vazios
serão preenchidos por uma string ou um sinal de interrogação, por exemplo.

        for($i = 0; $i <= 4; $i++){
                if ($data[$i] == "INDEFINIDO") {
                        $data[$i] = "vazio";
                }
        }

      echo '<tr>

As linhas que se seguem selecionam as colunas do arquivo .csv que se deseja trabalhar. Não
foi adicionado todos os índices escolhidos aqui pois são várias colunas e o arquivo ficaria
extenso.
                <td>' . $data[0] . '</td><td>' . $data[1] . '</td>
                <td>' . $data[2] . '</td><td>' . $data[3] . '</td>
                <td>' . $data[4] . '</td>
            </tr>';
      $infos[] = $data[0]. ';' . $data[1]. ';' .$data[2].';' .$data[3]. ';' .$data[4];
   }
```

```
    echo '</tbody></table>';
    fclose($handle);
}

$fp = fopen('resultados.csv', 'w');  //Um novo arquivo é criado e os dados serão escritos neste arquivo.

foreach ($infos as $info) {
    fputcsv($fp, array($info), ';', ' ');
}

fclose($fp);

?>
```