

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Cecília Regina Oliveira de Assis

**Análise Comparativa de Técnicas de
Classificação sobre uma Ferramenta de
Detecção de Empacotamento**

Uberlândia, Brasil

2018

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Cecília Regina Oliveira de Assis

**Análise Comparativa de Técnicas de Classificação sobre
uma Ferramenta de Detecção de Empacotamento**

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

Orientador: Prof. Dr. Kil Jin Brandini Park

Coorientador: Prof. Dr. Murillo Carneiro Guimarães

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2018

Cecília Regina Oliveira de Assis

Análise Comparativa de Técnicas de Classificação sobre uma Ferramenta de Detecção de Empacotamento

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

Trabalho aprovado. Uberlândia, Brasil, 10 de dezembro de 2018:

Prof. Dr. Kil Jin Brandini Park
Orientador

Prof. Dr. Murillo Carneiro Guimarães

Prof. Dr. Rodrigo Sanches Miani

Uberlândia, Brasil
2018

Lista de ilustrações

Figura 1 – Visão geral das fases do processo KDD	12
Figura 2 – Exemplo de classificação utilizando a técnica k NN	14
Figura 3 – Rede neural perceptron multicamadas	15
Figura 4 – Árvore de Decisão para classificação de indivíduo provável comprador de computador ou não	17
Figura 5 – Visualização de uma Árvore de Decisão no software WEKA	18
Figura 6 – Exemplo de formatação de resultados exposto pelo BinStat	20
Figura 7 – Exemplo de formatação de resultados exposto pelo BinStat para o WEKA	20
Figura 8 – Cabeçalho adicionado as bases de dados submetidas ao WEKA	21
Figura 9 – Formatação dos resultados exposto ao WEKA	21

Lista de tabelas

Tabela 1 – Configuração de parâmetros, no software WEKA, para Naive-Bayes Neurais	24
Tabela 2 – Configuração de parâmetros, no software WEKA, para Redes Neurais .	25
Tabela 3 – Configuração de parâmetros, no software WEKA, para classificador k NN	26
Tabela 4 – Configuração de parâmetros, no software WEKA, para Árvore de Decisão C4.5	27
Tabela 5 – Metadados da base de dados de treino	29
Tabela 6 – Metadados da base de dados de teste	30
Tabela 7 – Resultados apresentados pelo WEKA, com predição sobre base de treinamento	30
Tabela 8 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste em sua forma original.	31
Tabela 9 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software UPX.	31
Tabela 10 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software FSG	32
Tabela 11 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software Mew 11	32
Tabela 12 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software MPRESS	33
Tabela 13 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software XComp	33
Tabela 14 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software PECompact	34
Tabela 15 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software Themida	34
Tabela 16 – Nível médio de falsos positivos por empacotador	35
Tabela 17 – Métodos com os melhores resultados por base de dados	35
Tabela 18 – Métodos com os mais baixos resultados por base de dados	36

Lista de abreviaturas e siglas

KDD	Descoberta de Conhecimento em Bases de Dados
MD	Mineração de Dados
WEKA	Ambiente Waikato para Análise de Conhecimento
k NN	k -vizinhos mais próximos
RNA	Rede Neural Artificial
FSG	Fast Small Good
UPX	The Ultimate Packer for Executables
MPRESS	Matcode Compressor for Executables

Sumário

1	INTRODUÇÃO	8
2	REVISÃO BIBLIOGRÁFICA	11
2.1	Mineração de Dados	11
2.2	Técnicas de Classificação	13
2.2.1	<i>k</i> -vizinhos mais próximos (<i>k</i> NN)	13
2.2.2	Naive Bayes	14
2.2.3	Redes Neurais	15
2.2.4	Árvores de Decisão	16
2.3	Ambiente Waikato para Análise de Conhecimento	17
2.4	BinStat	18
3	DESENVOLVIMENTO	20
3.1	Pré-processamento	20
3.2	Metodologia	22
3.3	Sobre as bases de dados	22
3.4	Parametrização dos métodos	24
3.4.1	Naive-Bayes	24
3.4.2	Redes Neurais	24
3.4.3	<i>k</i> -vizinhos mais próximos	26
3.4.4	Árvore de Decisão C4.5	27
4	RESULTADOS	29
4.1	Metadados das bases de dados	29
4.1.1	Base de treino	29
4.1.2	Base de teste	29
4.2	Fase de treino	30
4.2.1	WEKA	30
4.2.2	C5.0	30
4.3	Arquivos Originais	31
4.4	UPX	31
4.5	FSG	32
4.6	Mew 11	32
4.7	MPRESS	33
4.8	XComp	33
4.9	PECompact	33

4.10	Themida	34
4.11	Falsos Positivos $\times$ Falsos Negativos	34
4.12	Ranqueamento dos Métodos	35
5	CONCLUSÃO	37
	REFERÊNCIAS	39
	APÊNDICES	43
	APÊNDICE A – CÁLCULOS ESTATÍSTICOS	44

1 Introdução

É notório o crescimento do volume de dados armazenados nos mais diversos servidores espalhados pelo mundo. Esses dados, quando examinados corretamente, podem derivar informações valiosas com as quais é possível a geração de conhecimento. O nome dado a esse processamento é Descoberta de Conhecimento em Bases de Dados (KDD) (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a).

O KDD possui cinco fases distintas (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b), com objetivos variados que conversam transmitindo resultados entre si; uma dessas fases é a Mineração de Dados.

A Mineração, como o próprio nome alude, busca a lapidação dos dados brutos com o propósito de encontrar informações relevantes e passíveis de utilização, por meio de diferentes tarefas, sendo elas:

- Descritivas: onde ocorre, possivelmente, a detecção de padrões por intermédio da correlação dos dados.
- De previsão: responsável pela previsão de comportamento dos dados, baseado em seus atributos.

É a partir do seguimento de previsão que se originam as técnicas de classificação, onde, sobre um conjunto de instâncias pré-catalogadas, denominado conjunto de treinamento, é aplicada uma função de mapeamento baseada nos atributos e rótulos de tais dados e assim novas instâncias cujos rótulos – também intitulados classes – eram desconhecidos se tornam classificadas (CARNEIRO, 2017).

O BinStat (PARK; RUIZ; MONTES, 2011a; PARK; RUIZ; MONTES, 2011b), baseado no trabalho dos autores Tabish, Shafiq e Farooq (2009), objetiva a detecção de uma característica específica amplamente utilizada por desenvolvedores de programas maliciosos, o empacotamento de executáveis binários.

O processo de empacotamento de um arquivo ocorre quando este é comprimido e cifrado, fazendo com que o novo documento se torne além de consideravelmente menor em tamanho, mascarado no que tange o seu conteúdo (YAN; ZHANG; ANSARI, 2008).

Para atingir o seu objetivo, o software constrói uma base de dados a partir da execução de treze cálculos estatísticos e da teoria da informação sobre um conjunto de aplicações. Os valores obtidos através destes são então repassados a uma técnica de mineração de dados que categoriza os executáveis analisados enquanto empacotados ou não.

Em seu estudo Park, Ruiz e Montes (2011a) optaram por adotar a Árvore de Decisão C.50 (QUINLAN, 2004) enquanto motor de classificação da característica de empacotamento dos executáveis analisados. A partir dessa, resultados bastante satisfatórios foram alcançados na rotulagem das aplicações.

Dessa forma, o objetivo geral, ou hipótese, desse trabalho se deu na validação da performance do BinStat frente outros métodos de classificação, a fim de observar se o comportamento de bons resultados preditivos ocorreria também entre diferentes técnicas de classificação.

A base para a escolha dos métodos classificatórios que confrontariam àquele elegido por Park, Ruiz e Montes (2011a) foi o trabalho de Wu et al. (2008), que aponta as dez técnicas de classificação mais influentes utilizadas pela comunidade de mineração de dados. Nesse, quatro foram aplicadas:

- Naive-Bayes;
- k -vizinhos mais próximos;
- Árvore de Decisão C4.5 e
- Redes Neurais.

Por sua relevância na comunidade tanto acadêmica quanto empresarial (HALL et al., 2009) o Ambiente Waikato para Análise de Conhecimento, popularmente conhecido como WEKA, foi adotado para o desenvolvimento do trabalho, ao passo que esse se denota como uma aplicação voltada para a mineração de dados dispondo de uma gama de algoritmos com procedimentos conhecidos e já implementados na linguagem JAVA.

Como objetivos específicos deste trabalho pode-se listar:

- A geração das bases de dados de treino e teste a serem manipuladas pelos métodos de classificação.
- O pré-processamento adequado das bases de dados geradas pelo programa BinStat;
- Avaliação do desempenho preditivo das técnicas de classificação escolhidas.

Ao fim do estudo é esperado o levantamento da qualidade do software desenvolvido por Park, Ruiz e Montes (2011a), segundo seu percentual de classificação defrente outros métodos da mineração de dados de modo que os resultados dessa pesquisa sejam empregados pela comunidade acadêmica, assim como o seu alvo de estudo.

A estrutura deste trabalho divide-se em:

- **Introdução:** apresentada no primeiro capítulo, nesta seção consta uma breve contextualização a respeito do estudo; o que o mesmo propõe; sua importância e ferramentas.
- **Revisão Bibliográfica:** aborda de forma mais clara os conceitos expostos na introdução, trazendo especificidades tangentes ao estudo.
- **Desenvolvimento:** retrata o desenvolvimento do trabalho, discorrendo sobre suas etapas e processos.
- **Resultados:** expõe os resultados obtidos pelo estudo, fazendo comparações entre os valores descritos pela árvore de decisão C5.0, adotada pelo BinStat, e as técnicas manipuladas no WEKA.
- **Conclusão:** finalizando o trabalho, nesta seção são apontadas as aquelas conclusões construídas a partir da análise dos resultados anteriores, trazendo uma macrovisão do estudo.

2 Revisão Bibliográfica

Este capítulo é responsável por retratar fundamentos teóricos da pesquisa, embasando recursos e conceitos empregados.

Tal base se divide entre as definições de mineração de dados, explanação sobre os cálculos estatísticos e os métodos classificadores escolhidos para abordagem. Exibe também uma seção que discute o software utilizado para composição dos resultados, o WEKA, assim como trabalhos que fizeram uso do mesmo.

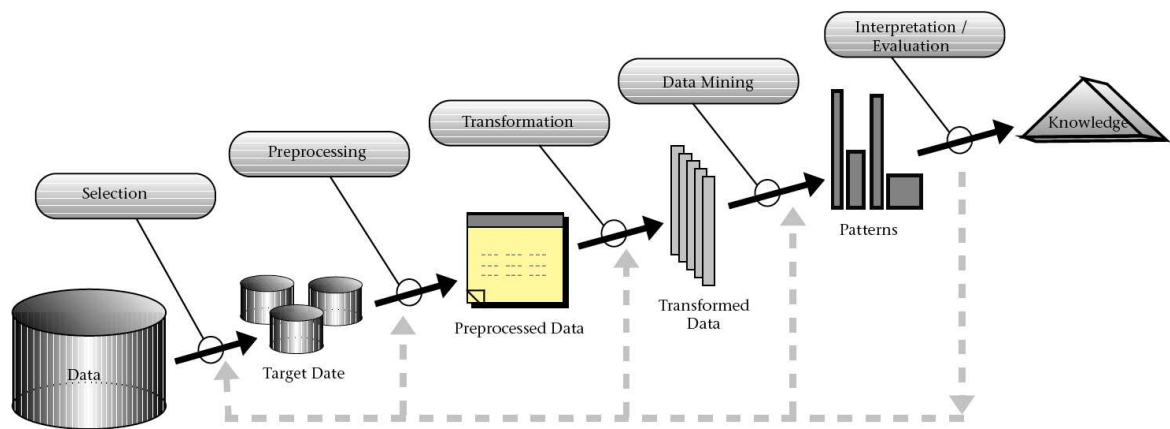
2.1 Mineração de Dados

Conforme abordado na Figura 1, a mineração de dados faz parte do processo de descoberta de conhecimento em bases de dados, do inglês *knowledge discovery in databases*, abreviado para KDD; termo este que nasceu no primeiro Workshop de *Knowledge Discovery in Databases*, sediado no ano de 1989, com o objetivo de enfatizar que a resposta de uma exploração orientada a dados é o conhecimento (PIATETSKY-SHAPIRO, 1995).

Definindo o conceito processo Frawley, Piatetsky-Shapiro e Matheus (1992) dizem que: “Descoberta de conhecimento é a extração não trivial de implícita, anteriormente desconhecida, e potencialmente útil informação dos dados.”

Este possui cinco fases (Figura 1), onde cada uma delas procura trabalhar os dados de forma a aperfeiçoá-los na busca de informações relevantes.

Figura 1 – Visão geral das fases do processo KDD



Fonte: (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a)

A mineração de dados é a etapa responsável pela aplicação de algoritmos sobre os conjuntos de dados em análise, a fim de extrair padrões não conhecidos anteriormente (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b), denotando a descoberta.

Maimon e Browarnik (2009) enumeram as diferentes metodologias presentes na mineração de dados:

- Clusterização: particionamento de um conjunto de dados de forma que aqueles que apresentam características similares pertençam ao mesmo grupo (AACH; CHURCH, 2001);
- Classificação: atribuição de rótulos pré-definidos para diferentes elementos com classes desconhecidas (GEURTS, 2001);
- Regressão: predição de possíveis valores a serem assumidos por um cenário partindo daqueles já apresentados pelo mesmo. (MAIMON; BROWARNIK, 2009);
- Extração de regras de Associação: geração de regras de associação frente padrões de comportamento dentre o conjunto de dados (LIU; HSU; MA, 1999).
- Series Temporais: análise e descoberta de padrões sobre valores de atributos que variam ao longo do tempo (MAIMON; BROWARNIK, 2009).
- Sumarização: sintetização e/ou aproximação dos dados de maneira que a essência desses se mantenha (INDYK; KOUDAS; MUTHUKRISHNAN, 2000);

2.2 Técnicas de Classificação

As técnicas de classificação tem por característica mapear conjuntos de dados não-rotulados em classes pré-definidas (KOHAVI, 1995), para isso, essas constroem modelos de classificação a partir de dados de treino que possuem informação quanto as classes que pertencem. Quando finalizados, tais modelos são aplicados sobre uma base de teste onde nenhuma informação a respeito das classes é disposta, os modelos então os catalogam de acordo com os rótulos previamente conhecidos (HAN; PEI; KAMBER, 2011).

São várias as técnicas classificadoras disponíveis na literatura, dentre elas temos:

- k -vizinhos mais próximos;
- Redes Neurais;
- Naive Bayes;
- Árvore de decisão;

2.2.1 k -vizinhos mais próximos (k NN)

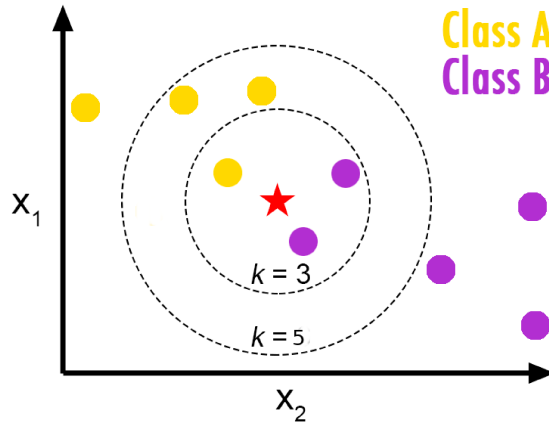
Neste, a classificação ocorre por meio da análise daqueles elementos que estão mais próximos à instância em questão.

O cálculo adotado para reconhecimento da distância entre os elementos, conforme citam Keller, Gray e Givens (1985), Peterson (2009) usualmente se representa pela distância euclidiana, denotando simplicidade a técnica.

Neste algoritmo o valor de k é definido empiricamente, sendo, de forma preferencial, um número ímpar, para que na classificação binária não ocorra problemas de quantidades iguais entre vizinhos com a mesma classe.

A Figura 2 demonstra a escolha de dois valores para k , sendo o primeiro três (3) e o segundo cinco (5). O elemento que deseja por classificação é representado pela estrela, disposta no centro da figura; duas classes, A e B, são passíveis de eleição.

Para k igual a três (3), o rótulo previsto para a instância em análise seria o da classe B, frente a maioria de classes apresentadas pelos vizinhos ao seu redor. Caso o valor selecionado para k fosse cinco, o elemento terminaria por pertencer a classe A.

Figura 2 – Exemplo de classificação utilizando a técnica k NN

Fonte: ADAPTADA. (DEWILDE, 2012)

2.2.2 Naive Bayes

A respeito deste método Murphy (2006) diz que “o pressuposto **Naive Bayes** ou **Idiot Bayes** é que todos os atributos são condicionalmente independentes dado o rótulo da classe”, logo, dado uma instância os atributos dessa não possuem correção entre si. Dessa maneira, a classificação das entidades tem íntima relação com os atributos das classes mais prováveis (RISH, 2001).

Porém ao ler-se “provável” o termo deve ser entendido enquanto probabilístico, sendo feita a utilização da fórmula Bayesiana, i. e., Teorema de Bayes que é dado por:

$$\underbrace{P(c|x)}_{\text{Probabilidade a posteriori}} = \frac{\overbrace{P(x|c)}^{\text{Verossimilhança}} \times \overbrace{P(c)}^{\text{Probabilidade a priori}}}{\underbrace{p(x)}_{\text{Evidência}}} \quad (2.1)$$

ADAPTADO. (DUDA; HART; STORK, 1995)

Onde c representa a classe analisada e x o preditivo (padrão) examinado.

Assim, aquela classe que possuir maior valor probabilístico será tomada na qualidade de classificadora da instância em estudo (RAY, 2015).

Rish (2001), Murphy (2006) ainda apontam que apesar do algoritmo aparentar ser muito simples, devido ao fato de que, geralmente, as propriedades de um elemento possuem correlação, o mesmo registra bons resultados quando comparado com outras estratégias mais sofisticadas (HUANG; LU; LING, 2003).

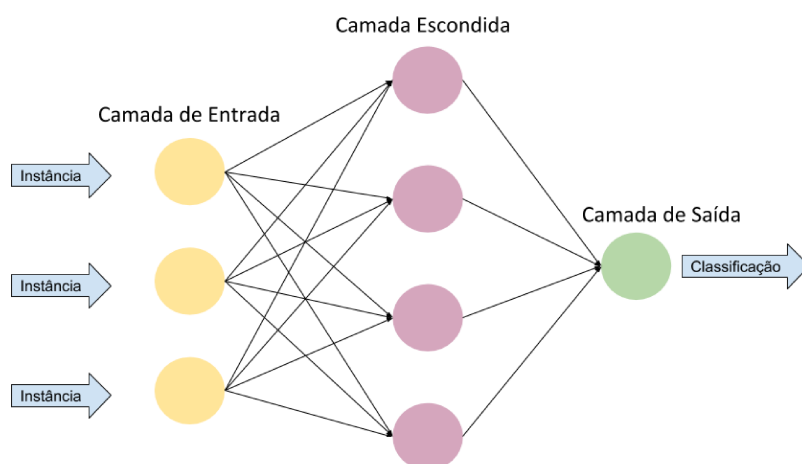
2.2.3 Redes Neurais

Do inglês *Artificial Neural Networks* (ANNs), as redes neurais artificiais (RNAs), conforme Dayhoff e DeLeo (2001), baseiam-se no modelo cerebral humano: na forma como os neurônios se conectam e comunicam.

Advindas da taxonomia *Perceptron* (ROSENBLATT, 1958), estas redes admitem camadas, preenchidas por múltiplos neurônios, que potencialmente possuem total conexão uns com os outros (Figura 3), porém, em contraste com outras metodologias, a *perceptron* multicamadas não levanta nenhuma hipótese a partir da distribuição dos dados (GARDNER; DORLING, 1998).

As conexões nas Redes Neurais são direcionadas, fornecendo valores de saída que, ao longo das camadas, serão utilizados como dados de entrada a àqueles que os recebem. Cada neurônio efetua então o processamento desses números, complementando-os com os pesos das entradas, e retorna um valor. Esse processo é exemplificado na Figura 3.

Figura 3 – Rede neural perceptron multicamadas



Fonte: Produzida pela autora

Por possuir múltiplas camadas, é possível que a configuração destas ocorra de diversas maneiras. Em geral, a organização utilizada é a seguinte: a primeira camada (*input*) é responsável pelas informações de entrada, ao passo que a última coordena os rótulos de saída (*output*): são estes que dirão a qual classe pertence cada elemento. Por fim, a camada intermediária (*hidden*) executa os cálculos de previsão. A quantidade de neurônios dispostos em cada camada, assim como o número de camadas intermediárias, são decididos de forma empírica.

Os cálculos de treinamento do modelo também poder ser operados de diferentes formas, algumas delas são:

- *Feed-forward*: onde os valores computados seguem somente uma direção: da camada de entrada para a de saída (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).
- *Backpropagation*: no qual os resultados apurados viajam entre ambos os sentidos da rede, indo e voltando, sendo que na volta é efetuado o cálculo do erro (ABRAHAM, 2005).

2.2.4 Árvores de Decisão

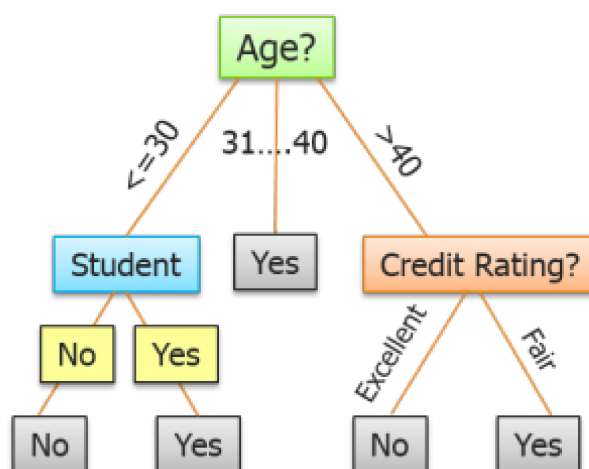
De maneira geral, as árvores de decisão, possuem a seguinte estrutura: os nós representam os atributos presentes no dado, enquanto os ramos ditam o fluxo condicional de decisão: valores que as propriedades podem revelar e que influenciam nos seus rótulos (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). Por fim, os nós finais da árvore, aqueles que não apresentam filhos, chamados de nós folhas, determinam a classe final de cada instância analisada.

A Figura 4 demonstra uma árvore de decisão que classifica uma entidade enquanto provável compradora de um produto ou não. Com cada nó representando um atributo, primeiro a idade do indivíduo é examinada, podendo essa passar por três valores:

- menor ou igual a trinta (30) anos;
- entre trinta e um (31) e quarenta (40) anos;
- e maior que quarenta (40) anos.

Se a idade em questão recair na primeira categoria (menor ou igual a trinta (30) anos) o indivíduo é classificado como estudante e se assim o for, é então demarcado como um não provável comprador; caso contrário sua classe é invertida. Estando o valor entre trinta e um (31) e quarenta (40) anos, um nó-folha é acessado e o indivíduo já recebe o rótulo de provável comprador, porém se o valor for maior que quarenta (40) anos, o atributo *Avaliação de Crédito* é analisado, e assim, sendo a avaliação excelente o personagem não será um provável comprador; em contrapartida se sua avaliação de crédito se demonstrar razoável a classificação será para provável comprador do produto.

Figura 4 – Árvore de Decisão para classificação de indivíduo provável comprador de computador ou não



Fonte: (SHARMA, 2015)

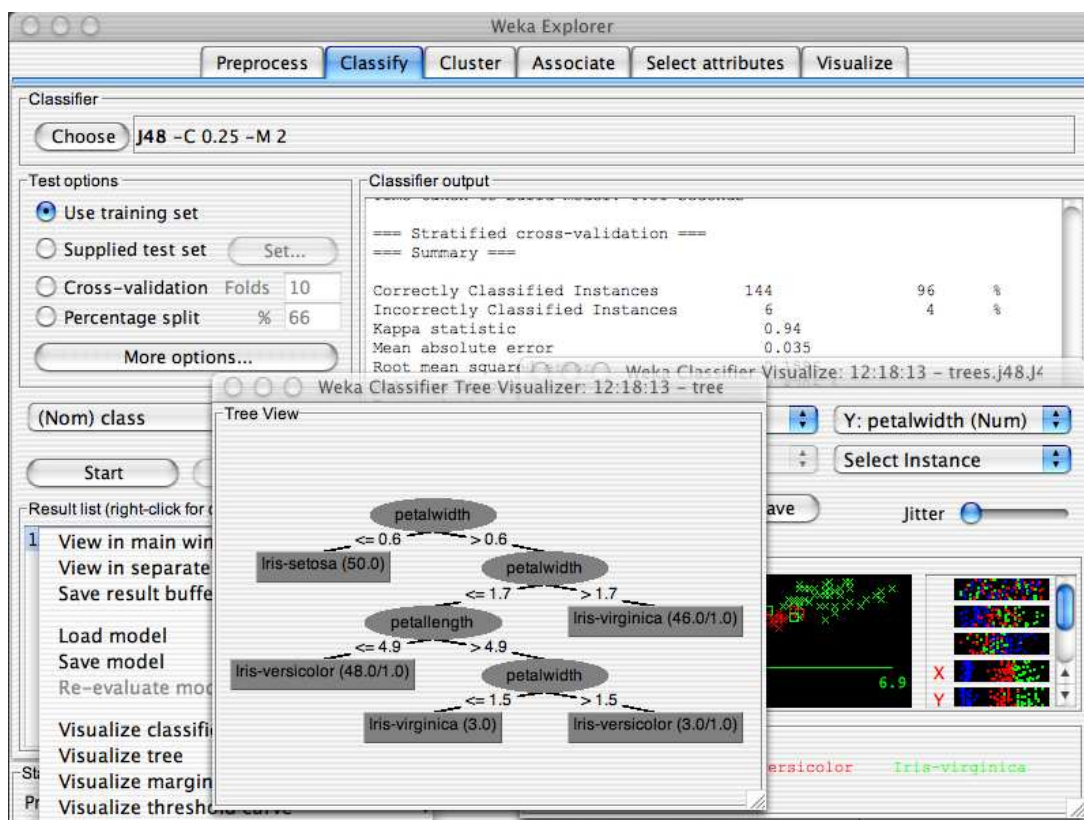
2.3 Ambiente Waikato para Análise de Conhecimento

O software, abreviado para **WEKA**, é uma aplicação escrita na linguagem JAVA (WITTEN et al., 1999) composta por uma vasta variedade de algoritmos, voltados para as tarefas de descoberta de conhecimento.

O objetivo da ferramenta, que possui distribuição livre, é facilitar a utilização de técnicas que abordam o aprendizado de máquina, popularmente conhecido como *machine learning*, de modo que qualquer pessoa possa aproveitar suas vantagens (HOLMES; DONKIN; WITTEN, 1994) ao efetuar o *KDD*.

Denotada enquanto referência para o aprendizado de máquina e a mineração de dados (HALL et al., 2009), a plataforma suporta diversas etapas do processo de descoberta de conhecimento, como o pré-processamento, a filtragem e a visualização de dados e modelos (Figura 5), assim como variadas técnicas da mineração de dados: SVM, Redes Neurais, Simple CART, dentre outros (WITTEN et al., 2016).

Figura 5 – Visualização de uma Árvore de Decisão no software WEKA



Fonte: (WAIKATO, 2011)

2.4 BinStat

O BinStat (*Binary Statistics*), uma aplicação desenvolvida por Park, Ruiz e Montes (2011a), tem por intuito classificar executáveis *Windows* enquanto prováveis softwares empacotados ou não.

A utilização do empacotamento para a ofuscação do conteúdo de arquivos é uma prática significativamente difundida entre os malwares, também conhecidos como vírus, que são programas maliciosos que infectam computadores alheios em busca de recompensas (BROSCH; MORGENSTERN, 2006).

O empacotamento, conforme explicam Yan, Zhang e Ansari (2008), é alcançado através do processo de compressão e cifragem de aplicações que, por sua vez, tem o seu tamanho original potencialmente reduzido e seu conteúdo, como já citado, encoberto.

O BinStat baseou-se na metodologia desenvolvida por Tabish, Shafiq e Farooq (2009) que efetua operações a nível de byte nos arquivos que examina e posteriormente submete seus valores à técnica de classificação Árvore de Decisão para validação dos programas.

Para isso, cada executável é analisado através da sua segmentação em blocos de 1024 bytes. Sobre estes são efetuados treze (13) cálculos estatísticos e da teoria da informação. Os resultados dessas operações são então expostos ao algoritmo de árvore de decisão C5.0 (QUINLAN, 2004), que por sua vez informa a qual classe pertence cada bloco analisado. A árvore de decisão C5.0 é uma variante da C4.5, também desenvolvida por Quinlan (1993).

3 Desenvolvimento

Nesta seção são abordadas as etapas de desenvolvimento do trabalho, assim como informações quanto aos parâmetros aplicados nos algoritmos e o processo de levantamento dos resultados.

3.1 Pré-processamento

Por ser a formatação dos dados aceita pela aplicação de Quinlan (2004), o BinStat apresenta os resultados de seus cálculos seguindo o modelo exibido pela Figura 6, onde cada linha apresenta, separados por vírgula, os valores dos treze cálculos estatísticos relativos a um determinado bloco, o status do bloco (“yes” caso empacotado, “no” caso contrário), a identificação do bloco dentro do executável e o MD5 do executável original (PARK; RUIZ; MONTES, 2011a).

Figura 6 – Exemplo de formatação de resultados exposto pelo BinStat

```
0.005388,95.159503,0.029534,0.646484,0.019531,0.326108,  
0.746169,0.344504,7.592288,0.130506,0.041571,101.373204,  
0.323242,yes,BLOCK29,3afb6adccb65b1c4284833080e878db3
```

Fonte: (PARK; RUIZ; MONTES, 2011a)

Porém tal formato não é aceito pelo WEKA e em virtude disso se fez necessária a manipulação do BinStat para que esse gerasse suas saídas em um formato aceito pela ferramenta onde ocorreria a aplicação dos métodos de classificação.

Sendo assim, o programa foi modificado para que os números expostos por ele seguissem a formatação disposta na Figura 7, onde os treze (13) primeiros valores denotavam aqueles apresentados pelas cálculos estatísticos e o último enumerava a classe a qual pertencia o bloco em questão (“yes” caso empacotado, “no” caso contrário):

Figura 7 – Exemplo de formatação de resultados exposto pelo BinStat para o WEKA

```
0.543606,98.479515,0.727562,1.075195,0.727539,0.858145,0.216714,0.132232,2.3730  
19,1.465185,0.143676,122.393136,0.537598,yes
```

Fonte: Produzida pela autora

Após a criação das bases de dados a serem utilizadas no estudo, um cabeçalho específico ao WEKA (Figura 8) foi adicionado em cada uma destas. Nesse residiam as

informações quanto a qual tipo de dados essas bases se referiam e de qual maneira elas eram formadas.

Figura 8 – Cabeçalho adicionado as bases de dados submetidas ao WEKA

```
@relation pcec2-test-weka

@attribute si numeric
@attribute candis numeric
@attribute mindis numeric
@attribute manhdis numeric
@attribute chedis numeric
@attribute bradis numeric
@attribute angsep numeric
@attribute corrcoef numeric
@attribute entropy numeric
@attribute kulldiv numeric
@attribute jendiv numeric
@attribute itadiv numeric
@attribute totvar numeric
@attribute class {yes,no}
```

Fonte: Produzida pela autora

Sendo assim, a estrutura final das bases de dados exposta ao ambiente Waikato foi a seguinte:

Figura 9 – Formatação dos resultados exposto ao WEKA

```
@relation pcec2-test-weka

@attribute si numeric
@attribute candis numeric
@attribute mindis numeric
@attribute manhdis numeric
@attribute chedis numeric
@attribute bradis numeric
@attribute angsep numeric
@attribute corrcoef numeric
@attribute entropy numeric
@attribute kulldiv numeric
@attribute jendiv numeric
@attribute itadiv numeric
@attribute totvar numeric
@attribute class {yes,no}

@data
0.543606,98.479515,0.727562,1.075195,0.727539,0.858145,0.216714,0.132232,2.3730
19,1.465185,0.143676,122.393136,0.537598,yes
```

Fonte: Produzida pela autora

É importante observar que diferentemente do formato original de saída do BinStat, esse não exibe informações quanto ao número do pacote ao qual a instância em questão se relaciona, assim como o MD5 de seu executável. Tal característica se deu devido ao

fato de que quando apresentadas ao ambiente tais informações impossibilitavam o uso das técnicas de classificação elegidas e como somente os valores levantados pelas estatísticas eram pertinentes ao estudo, apenas esses foram mantidos.

3.2 Metodologia

Os passos para o desenvolvimento do estudo foram:

1. Modificação do formato de geração dos resultados apresentado pelo BinStat;
2. Produção das bases de dados de treino e teste;
 - Submissão ao BinStat dos arquivos de treino e teste em seu formato original e empacotado.
3. Pré-processamento das bases de dados;
4. Criação dos modelos de classificação;
5. Submissão das bases de dados ao modelos e
6. Análise dos resultados preditivos dos métodos.

Todas as etapas foram realizadas tanto para o algoritmo de Quinlan (2004) quanto para cada uma das técnicas de classificação empregadas a partir do ambiente Waikato.

3.3 Sobre as bases de dados

Para a criação das bases de dados, o recolhimento dos resultados e reprodução do trabalho de Park, Ruiz e Montes (2011a), os mesmos empacotadores de executáveis binários apontados no artigo, foram utilizados:

- The Ultimate Packer for Executables (UPX), freeware, versão 3.05 (OBERHUMER; MOLNAR; REISER, 1996–2010);
- Fast Small Good (FSG), freeware, versão 2.0 (Bart, 2009);
- Mew 11 SE, freeware, versão 1.2 (Northfox, 2004);
- Matcode Compressor for Executables (MPRESS), freeware, versão 2.18 (MAT-CODE Software, 2007–2010);
- XComp/XPack: A freeware PE32-imagefile packer/rebuilder, freeware, versão 0.97 (JoKo, 2007);

- PECompact, shareware, versão 3.00.2 (Bitsum Technologies, 2009) e
- Themida, shareware, versão 2.2.7.0 (Oreans Technologies, 2009).

Sendo o primeiro voltado para sistemas operacionais Linux, e o restante para Windows.

Quatrocentos e cinquenta e seis (456) executáveis advindos de instalações Windows, escolhidos aleatoriamente denotaram a composição da base de dados de treino (PARK; RUIZ; MONTES, 2011b). Sobre esses os seis (6) primeiros empacotadores foram aplicados e assim, somados aos arquivos em sua forma original, resultaram em um conjunto de duzentos e três mil e novecentos e quarenta e oito (203948) instâncias, onde cada uma delas representava um bloco dos três mil e cento e noventa e dois (3192) arquivos analisados.

Na fase de teste vinte e dois (22) executáveis ainda não vistos pelos modelos foram abordados, de igual maneira advindos de instalações Windows, escolhidos aleatoriamente e empregados em sua forma original e empacotada, porém, para esses a qualidade de empacotamento foi alcançada através de todos os empacotadores citados acima.

Assim, trinta e um mil e setecentos e dois (31702) elementos no total constituíram as bases de dados de teste, através da segmentação de cento e setenta e seis (176) executáveis binários.

A base de dados de treinamento combinou àqueles blocos que haviam sido empacotados com blocos não empacotados, enquanto as de teste os apresentaram de forma individual sendo divididas, quando necessário, de acordo com o empacotador que as originou em busca de uma melhor visualização dos resultados.

Park, Ruiz e Montes (2011a) não fizeram o uso de nenhum critério específico para a escolha dos arquivos que comporiam as bases de dados de seu trabalho, porém, optaram por adotar o Themida somente na geração dos dados de teste devido ao seu comportamento específico no empacotamento dos arquivos, constando histogramas vastamente distintos dos demais.

Dessa forma, para aumentar a eficácia do trabalho, somente os arquivos originais utilizados pelos autores foram empregados, assim como, todas as bases de dados (treino e teste) foram regeradas e ambas as etapas tiveram execução tanto no método original de classificação do BinStat, a árvore de decisão C5.0, quanto nas quatro (4) novas técnicas classificatórias elegidas.

3.4 Parametrização dos métodos

Através do software WEKA, todos os classificadores escolhidos para o desenvolvimento do estudo suportam a configuração de parâmetros a fim da modificação de sua execução, contudo, tal parametrização exige diversos estudos e testes a fim da obtenção do melhor valor para cada um deles.

Logo, por entender essa dificuldade, o WEKA oferece um conjunto padrão de configurações que com a finalidade de facilitar o trabalho foram adotadas.

3.4.1 Naive-Bayes

Para o classificador Naive-Bayes, os parâmetros empregados com seus respectivos valores foram:

Tabela 1 – Configuração de parâmetros, no software WEKA, para Naive-Bayes Neurais

<i>useKernelEstimator</i>	-	False
<i>numDecimalPlaces</i>	-	2
<i>batchSize</i>	-	100
<i>debug</i>	-	False
<i>displayModelInOldFormat</i>	-	False
<i>doNotCheckCapabilities</i>	-	False
<i>useSupervisedDiscretization</i>	-	False

Constituindo aqueles mais relevantes:

- **useKernelEstimator**: define a utilização, ou não, do estimador de kernel para os atributos ao invés da distribuição normal.
- **useSupervisedDiscretization**: determina se a discretização dos valores dos atributos ocorrerá.

3.4.2 Redes Neurais

Para as redes neurais, os parâmetros aplicados com seus respectivos valores foram:

Tabela 2 – Configuração de parâmetros, no software WEKA, para Redes Neurais

<i>seed</i>	-	0
<i>momentum</i>	-	0.2
<i>nominalToBinaryFilter</i>	-	True
<i>hiddenLayers</i>	-	a
<i>validationThreshold</i>	-	20
<i>GUI</i>	-	False
<i>normalizeAttributes</i>	-	True
<i>numDecimalPlaces</i>	-	2
<i>batchSize</i>	-	100
<i>decay</i>	-	False
<i>validationSetSize</i>	-	0
<i>trainingTime</i>	-	500
<i>debug</i>	-	False
<i>autoBuild</i>	-	True
<i>normalizeNumericClass</i>	-	True
<i>learningRate</i>	-	0.3
<i>doNotCheckCapabilities</i>	-	False
<i>reset</i>	-	True

Sendo os mais preponderantes:

- **seed**: define a semente de inicialização do gerador de números aleatórios.
- **momentum**: estipula o *Momentum* utilizado na atualização dos pesos da rede.
- **nominalToBinaryFilter**: informa se o filtro de conversão de valores nominais para binários deverá ser aplicado.
- **hiddenLayers**: configura a quantidade de camadas escondidas na rede. Quando igual a **a**, significa que o valor deve ser o somatório da quantidade de atributos com o número de classes dividido por dois (2).
- **validationThreshold**: estabelece o montante de vezes possíveis que o erro da rede pode aumentar, antes da fase de treinamento do algoritmo ser finalizada.
- **normalizeAttributes**: determina se o valor dos atributos deverá passar por um filtro de normalização.
- **decay**: possibilita a diminuição da taxa de aprendizado do algoritmo, aplicando um cálculo que divide o valor inicial pelo número de épocas.
- **validationSetSize**: especifica o tamanho, em porcentagem, do conjunto de validação. Quando igual a zero (0), o algoritmo executa de acordo com o número de épocas.

- **trainingTime**: total de épocas, também chamadas de iterações, do algoritmo.
- **autoBuild**: determina se ocorrerá a adição e conexão das camadas escondidas da rede.
- **normalizeNumericClass**: informa se classes numéricas serão submetidas ao filtro de normalização.
- **learningRate**: define a taxa de aprendizado do algoritmo.
- **reset**: permite ao programa reiniciar o algoritmo, com um valor menor para a taxa de aprendizado, caso a classificação indicada pelo modelo seja diferente da real.

3.4.3 *k*-vizinhos mais próximos

O Ambiente Waikato muitas vezes apresenta diversas técnicas de mineração de dados sob nomenclaturas diferentes daquelas popularmente conhecidas, frente ao fato de que este disponibiliza a reimplementação dos métodos já existentes. Tal situação ocorre com o algoritmo *k*NN, sendo esse chamado de *IBk* no software.

Para este método, o WEKA enumera variáveis menos complexas quando comparadas a àquelas disponíveis para as RNAs. Os valores aplicados foram:

Tabela 3 – Configuração de parâmetros, no software WEKA, para classificador *k*NN

<i>numDecimalPlaces</i>	-	2
<i>batchSize</i>	-	100
<i>KNN</i>	-	1
<i>distanceWeighting</i>	-	False
<i>nearestNeighbourSearchAlgorithm</i>	-	LinearNNSearch
<i>debug</i>	-	False
<i>windowSize</i>	-	0
<i>doNotCheckCapabilities</i>	-	False
<i>meanSquared</i>	-	False
<i>crossValidate</i>	-	False

Significando aqueles mais salientes:

- **numDecimalPlaces**: quantidade de casas decimais apresentadas para os resultados do modelo.
- **KNN**: número de vizinhos a serem analisados.
- **distanceWeighting**: método matemático para cálculo do peso das distâncias.

- **nearestNeighbourSearchAlgorithm**: algoritmo de procura do vizinho mais próximo.
- **windowSize**: montante máximo de instâncias na fila de treinamento.
- **meanSquared**: utilização do erro quadrado médio ao invés do erro absoluto médio durante as validações cruzadas.
- **crossValidate**: indicação sobre o uso do método de validação **deixa-um-fora** (*hold-one-out*) nas validações cruzadas, a fim de eleger o melhor valor entre 1 e aquele especificado no parâmetro KNN.

3.4.4 Árvore de Decisão C4.5

Esta técnica, de igual forma à anterior é abordada no WEKA sob um diferente nome: *J4.8*, e para essa o software possibilita a configuração de dezessete parâmetros. Àquela empregada foi:

Tabela 4 – Configuração de parâmetros, no software WEKA, para Árvore de Decisão C4.5

<i>seed</i>	-	1
<i>unpruned</i>	-	False
<i>confidenceFactor</i>	-	0.25
<i>numFolds</i>	-	3
<i>numDecimalPlaces</i>	-	2
<i>batchSize</i>	-	100
<i>reducedErrorPruning</i>	-	False
<i>useLaplace</i>	-	False
<i>doNotMakeSplitPointActualValue</i>	-	False
<i>debug</i>	-	False
<i>subtreeRaising</i>	-	True
<i>saveInstanceData</i>	-	False
<i>binarySplits</i>	-	False
<i>doNotCheckCapabilities</i>	-	False
<i>minNumObj</i>	-	2
<i>useMDLcorrection</i>	-	True
<i>collapseTree</i>	-	True

Constituindo aqueles mais significantes:

- **seed**: define a semente de inicialização do gerador de números aleatórios.
- **unpruned**: aponta se a árvore será podada.
- **confidenceFactor**: informa o fator de confiança para a podagem da árvore.

- **numFolds**: especifica a quantidade de dados empregada para a podagem de erro reduzido.
- **reducedErrorPruning**: denota se a podagem de erro reduzido será aplicada ao invés da podagem C4.5.
- **useLaplace**: expõe se a contabilização das folhas será suavizada segundo método de *Laplace*.
- **doNotMakeSplitPointActualValue**: sinaliza se o ponto de divisão da árvore será um valor de dado real.
- **subtreeRaising**: informa se a elevação do número de sub-árvores será considerada na podagem.
- **binarySplits**: reflete o uso, ou não, da divisão de binários para atributos nominais durante a construção da árvore.
- **minNumObj**: caracteriza o número mínimo de instâncias por folha.
- **useMDLcorrection**: indica se a correção MDL será aplicada quando encontradas divisões em atributos numéricos.
- **collapseTree**: aponta se ocorrerá a remoção de partes que não reduzem o erro da fase de treinamento.

A técnica de classificação utilizada pelo BinStat também teve a aplicação de parâmetros, no que tangia a característica de *adaptive boosting*, ou “impulsionador adaptativo”, do método. Por apresentar o melhor resultado dentre os analisados (10, 20, 30) o valor 30 foi adotado (PARK; RUIZ; MONTES, 2011b).

4 Resultados

Neste capítulo, serão apresentados e comparados os resultados da árvore de decisão C5.0 frente àqueles demonstrados pelos quatro (4) métodos classificatórios elegidos e utilizados através do software WEKA.

É importante salientar que nem todas as bases de teste descreveram a mesma quantidade de instâncias em consequência das limitações de cada empacotador, e que as considerações efetuadas levaram em conta a não otimização dos parâmetros aplicados aos métodos de classificação.

4.1 Metadados das bases de dados

4.1.1 Base de treino

Para cada um dos empacotadores e os arquivos originais, a base de dados de treino configurou as seguintes informações a respeito das instâncias:

Tabela 5 – Metadados da base de dados de treino

Origem das Instâncias	Quantidade de Instâncias	Atributos
Arquivos Originais	55035	13
UPX	23251	13
FSG	27852	13
Mew 11	24203	13
MPRESS	26026	13
XComp	22767	13
PECompact	24814	13

A porcentagem de blocos não empacotados do conjunto de treino foi de vinte e sete por cento (27%), enquanto àquela para os que apresentavam a característica de empacotamento denotou setenta e três por cento (73%) da composição da base.

4.1.2 Base de teste

Já para a base de teste, as informações acerca das instâncias constituíram:

Tabela 6 – Metadados da base de dados de teste

Origem das Instâncias	Quantidade de Instâncias	Atributos
Arquivos Originais	1427	13
UPX	656	13
FSG	650	13
Mew 11	576	13
MPRESS	663	13
XComp	655	13
PECompact	712	13
Themida	26363	13

O percentual de blocos originais nas bases de teste foi de aproximadamente quatro por cento (4%).

4.2 Fase de treino

Logo na fase de treinamento foi possível observar resultados satisfatórios nas classificações apresentadas pelos modelos por meio da realização de uma validação parcial dos mesmos. Para isso, parte dos dados de treinamento foram utilizados de forma que os modelos efetuassem classificações sobre as instâncias e avaliassem sua acurácia.

4.2.1 WEKA

Os algoritmos do ambiente, utilizando cem por cento (100%) dos dados de treinamento – duzentos e três mil e novecentos e quarenta e oito (203948) instâncias – demonstraram os seguintes resultados:

Tabela 7 – Resultados apresentados pelo WEKA, com predição sobre base de treinamento

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	86.1798
Naive-Bayes	79.2364
k NN (IB k)	96.0367
C4.5 (J48)	89.5415

4.2.2 C5.0

O algoritmo de Quinlan (2004), consoante ao software WEKA, relatou resultados pertinentes, onde, analisando a mesma base de treino, obteve 91.3111 por cento (%) de acerto.

4.3 Arquivos Originais

Primeiramente os arquivos em seu estado original, ou seja, não empacotados foram submetidos ao BinStat e seus respectivos valores analisados. Mil e quatrocentos e vinte e sete (1427) elementos constituíram a base, e para essa os modelos apresentaram os seguintes resultados:

Tabela 8 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste em sua forma original.

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	81.78
Naive-Bayes	89.7687
k NN (IB k)	76.0367
C4.5 (J48)	75.9636
C5.0	80.2097

Aquele método que se saiu melhor foi o Naive-Bayes, sendo o único a apresentar acurácia acima de oitenta e cinco por cento (85%).

A técnica obteve 13.8051 pontos percentuais de diferença em relação a àquela que apresentou o pior resultado: a árvore de decisão C4.5.

4.4 UPX

Após a avaliação dos arquivos originais, os empacotados foram examinados. A base gerada pelos executáveis empacotados através do software UPX contou com seiscentos e cinquenta e seis (656) instâncias para análise e levantou as seguintes informações quanto a propriedade de empacotamento dos blocos:

Tabela 9 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software UPX.

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	92.2256
Naive-Bayes	79.4207
k NN (IB k)	94.5122
C4.5 (J48)	96.0366
C5.0	94.3854

Ao observar os resultados exibidos para esse empacotador, em relação a àqueles levantados pela análise anterior, nota-se que os valores foram contrários: o método C4.5, que havia relatado resultados inferiores no estudo dos arquivos originais, obteve o maior valor dentre os demais.

A diferença entre o segundo e terceiro colocado foi pequena: somente 0.1268 décimos.

4.5 FSG

Com uma base composta por seiscentos e cinquenta (650) elementos, os valores relatados para o empacotador *FSG* foram:

Tabela 10 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software *FSG*

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	95.0769
Naive-Bayes	84
k NN (IB k)	96
C4.5 (J48)	97.5385
C5.0	96.7840

Igualmente ao ocorrido para o empacotador *UPX*, o método árvore de decisão C4.5 exibiu o melhor resultado, sendo aproximadamente 13% superior ao último colocado.

4.6 Mew 11

A base de teste construída por esse empacotador compreendeu quinhentos e setenta e seis (576) instâncias. Os resultados à ela apresentados foram:

Tabela 11 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software *Mew 11*

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	98.0903
Naive-Bayes	91.8403
k NN (IB k)	95.3125
C4.5 (J48)	98.9583
C5.0	97.9274

A acurácia dos métodos para o empacotador *Mew 11* foi significativa, e até mesmo o classificador que exibiu o resultado mais baixo obteve números acima de noventa por cento (90%) de acerto.

4.7 MPRESS

Para o MPRESS, a base registrou seiscentos e sessenta e três (663) instâncias, para as quais a acurácia obtida foi:

Tabela 12 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software MPRESS

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	89.4419
Naive-Bayes	75.8673
k NN (IB k)	93.5143
C4.5 (J48)	97.5385
C5.0	96.3963

Novamente, os valores de acurácia dos modelos foram consideráveis, contudo o método classificador Naive-Bayes relatou uma diferença significativa em relação ao melhor colocado, a árvore de decisão C4.5: aproximadamente vinte e um (21) pontos percentuais.

4.8 XComp

Os blocos dos executáveis binários empacotados pelo software XComp foram responsáveis por seiscentos e cinquenta e cinco (655) registros, que computaram os seguintes valores:

Tabela 13 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software XComp

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	92.6718
Naive-Bayes	79.084
k NN (IB k)	93.2824
C4.5 (J48)	97.0992
C5.0	95.7446

A acurácia dos modelos para o empacotador XComp foi satisfatória: com resultados acima de noventa por cento (92%) de acerto. O Naive-Bayes repetidamente apresentou a mais baixa resposta, sendo aproximadamente vinte e dois por cento (22%) inferior que o melhor ranqueado.

4.9 PECompact

Os dados gerados pela aplicação PECompact enumeraram setecentos e doze (712) elementos, que, ao serem submetidos aos modelos classificatórios, exibiram os valores

abaixo:

Tabela 14 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software **PECompact**

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	92.5562
Naive-Bayes	79.9157
k NN (IB k)	92.2753
C4.5 (J48)	95.7865
C5.0	95.3846

A mais baixa acurácia foi a registrada pelo método Naive-Bayes. Em contra partida, a diferença entre o segundo (2º) e o terceiro (3º) colocados foi pequena, somente 0.23%. Contudo, as técnicas continuaram a relatar valores consideráveis.

4.10 Themida

Por fim, o último empacotador pelo qual os arquivos de teste foram manipulados foi o Themida. Seus resultados foram:

Tabela 15 – Resultados apresentados pelas técnicas de classificação, para os arquivos de teste empacotados através do software **Themida**

Método	Instâncias Classificadas Corretamente (%)
Redes Neurais	99.4917
Naive-Bayes	99.0631
k NN (IB k)	96.5634
C4.5 (J48)	98.9076
C5.0	99.5107

As instâncias fruto deste empacotador foram, notoriamente, as com mais êxito identificadas. Todos os valores alcançaram patamares acima de noventa e cinco por cento (95%) de acurácia, sendo que àqueles melhor colocados superaram a marca de noventa e nove por cento (99%) de acerto.

Para esse software, foi interessante observar que a técnica original utilizada pelo BinStat, foi aquela que se sobressaiu nos resultados.

4.11 Falsos Positivos $\times$ Falsos Negativos

O nível médio de falsos positivos, por empacotador, relatado pelos modelos foi:

Tabela 16 – Nível médio de falsos positivos por empacotador

Empacotador	Média de Falsos Positivos(%)
UPX	8.6839
FSG	6.12012
Mew 11	3.5741
MPRESS	9.4483
XComp	8.4236
PECompact	8.8613
Themida	1.2927

Enquanto o nível médio de falsos negativos, para os arquivos em estado original, valorou-se em 19.24826 por cento (%).

A taxa média, em ordem geral, de falsos positivos computada pelos métodos foi de 6.62273 por cento (%).

4.12 Ranqueamento dos Métodos

A seguir serão apresentados os métodos que exibiram os resultados mais e menos satisfatórios, quando comparados aos demais. Os rankings foram segmentados por bases de dados.

Tabela 17 – Métodos com os melhores resultados por base de dados

Base de Dados	Classificador com Melhor Resultado
Arquivos Originais	Naive-Bayes
UPX	C4.5 (J48)
FSG	C4.5 (J48)
Mew 11	C4.5 (J48)
MPRESS	C4.5 (J48)
XComp	C4.5 (J48)
PECompact	C4.5 (J48)
Themida	C5.0

A partir da visão anterior, fica claro o destaque que o algoritmo de C4.5 ostentou em relação ao demais, obtendo o primeiro lugar entre os colocados em setenta e cinco por cento (75%) das bases, ao passo que através da análise da Tabela 18, o método Naive-Bayes se mostrou inferior em igual proporção.

Quanto as técnicas que não apresentaram resultados tão significativos durante as análises, têm-se:

Tabela 18 – Métodos com os mais baixos resultados por base de dados

Base de Dados	Classificador com Mais Baixo Resultado
Arquivos Originais	k NN (IB k)
UPX	Naive-Bayes
FSG	Naive-Bayes
Mew 11	Naive-Bayes
MPRESS	Naive-Bayes
XComp	Naive-Bayes
PECompact	Naive-Bayes
Themida	k NN (IB k)

5 Conclusão

Após o alcance dos objetivos específicos propostos e a análise dos resultados apresentados, pode-se concluir que as metodologias de detecção de empacotamento escolhidos por Park, Ruiz e Montes (2011a) foram bem fundamentados, haja visto as significativas acurácias expostas pelas técnicas de classificação abordadas e seus respectivos modelos.

Nota-se também que para os métodos empregados no estudo, a base que caracterizou maior dificuldade de rotulação foi a de blocos oriundos dos arquivos originais, onde o erro médio alcançou valor maior que quinze por cento (15%). Tal resultado foi possivelmente observado devido a baixa proporção de blocos não empacotados em relação a àqueles que apresentavam a característica dentre a base de dados de treino.

Em contrapartida, aqueles arquivos empacotados através do software **Themida** foram mais fáceis de terem os seus blocos detectados com resultados tangendo quase cem por cento (100%) de acurácia em diferentes técnicas de classificação conhecidas e renomadas.

Estudos não foram realizados a fim de compreender possíveis diferenças na estrutura de execução do software **Themida** em relação aos demais, porém acredita-se que o resultado significativo das classificações para os blocos originados desse ocorreram devido às características difundidas da base de treino, que contou com informações tanto a respeito de blocos não empacotados, quanto empacotados, estando os últimos em maior proporção.

Outro ponto importante é que a técnica de classificação árvore de decisão C4.5, do mesmo autor daquela utilizada no trabalho base desse estudo: Quinlan (1993), foi a que relatou o melhor desempenho preditivo, corroborando com a escolha apropriada de materiais adotada por Park, Ruiz e Montes (2011a). Tal desempenho, possivelmente, advém da maneira a qual o modelo desta técnica é construído: a partir da análise fina dos atributos das instâncias de modo que estes definem os ramos, e assim pontos de decisão, da mesma.

Já o método de classificação Naive-Bayes, para o tipo de abordagem empregada pelo BinStat, supostamente, não seja o mais indicado para a detecção da característica de empacotamento de executáveis por demonstrar resultados pouco satisfatórios segundo as análises apresentadas. Performance essa, porventura, consequente da natureza do algoritmo que pressupõe a independência condicional dos atributos de uma instância, dependendo então da probabilidade das classes e valores relacionados a cada uma delas.

A respeito do software BinStat, é crível dizer que esse, por seus pertinentes resultados frente outros métodos de classificação, pode ser seguramente recomendado para o alcance de seu objetivo e que as informações levantadas por este estudo são de grande valia para a comunidade acadêmica e também para àquela fora do âmbito universitário, dado que tanto o aprendizado de máquina, quanto a segurança da informação permeiam diversas áreas da era digital.

Enquanto trabalhos futuros são visados a utilização e análise do BinStat frente empacotadores mais recentes; a investigação do comportamento do software perante vírus, assim como o empacotamento dos mesmos e aplicações *ransomware*. A detecção de cifragens e classificação de tráfego de rede também serão alvo de estudo e desenvolvimento de futuras pesquisas.

Referências

AACH, J.; CHURCH, G. M. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, Oxford University Press, v. 17, n. 6, p. 495–508, 2001. Citado na página 12.

ABRAHAM, A. Artificial neural networks. *Handbook of measuring system design*, Wiley Online Library, 2005. Citado na página 16.

Bart. *Fast Small Good*. 2009. Disponível em: <<http://www.woodmann.com/collaborative/tools/index.php/FSG>>. Citado na página 22.

Bitsum Technologies. *PECompact*. 2009. Disponível em: <<http://www.bitsum.com/pecompact.php>>. Citado na página 23.

BROSCH, T.; MORGENSTERN, M. Runtime packers: The hidden problem. *Black Hat USA*, 2006. Citado na página 18.

BUGATTI, P. H. *Análise da influência de funções de distância para o processamento de consultas por similaridade em recuperação de imagens por conteúdo*. Tese (Doutorado) — Universidade de São Paulo, 2008. Citado na página 44.

CARNEIRO, M. G. *Redes complexas para classificação de dados via conformidade de padrão, caracterização de importância e otimização estrutural*. 161 p. Tese (Doutorado) — Universidade de São Paulo, 2017. Citado na página 8.

COVER, T. M.; THOMAS, J. A. *Elements of information theory*. [S.l.]: John Wiley & Sons, 2012. Citado 3 vezes nas páginas 44, 45 e 46.

DAYHOFF, J. E.; DELEO, J. M. Artificial neural networks. *Cancer*, Wiley Online Library, v. 91, n. S8, p. 1615–1635, 2001. Citado na página 15.

DEWILDE, B. *Classification of Hand-written Digits (3)*. 2012. <<http://bdewilde.github.io/blog/blogger/2012/10/26/classification-of-hand-written-digits-3/>>. (Accessed on 07/10/2017). Citado na página 14.

DUDA, R. O.; HART, P. E.; STORK, D. G. Pattern classification and scene analysis 2nd ed. *ed: Wiley Interscience*, 1995. Citado na página 14.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. *AI magazine*, v. 17, n. 3, p. 37, 1996. Citado 2 vezes nas páginas 8 e 12.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. Knowledge discovery and data mining: Towards a unifying framework. In: *KDD*. [S.l.: s.n.], 1996. v. 96, p. 82–88. Citado 2 vezes nas páginas 8 e 12.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge discovery in databases: An overview. *AI magazine*, v. 13, n. 3, p. 57, 1992. Citado na página 11.

- FUGLEDE, B.; TOPSOE, F. Jensen-shannon divergence and hilbert space embedding. In: IEEE. *International Symposium on Information Theory*. [S.l.], 2004. p. 31. Citado na página 46.
- GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, Elsevier, v. 32, n. 14, p. 2627–2636, 1998. Citado na página 15.
- GEURTS, P. Pattern extraction for time series classification. In: SPRINGER. *European Conference on Principles of Data Mining and Knowledge Discovery*. [S.l.], 2001. p. 115–127. Citado na página 12.
- HALL, M. et al. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, ACM, v. 11, n. 1, p. 10–18, 2009. Citado 2 vezes nas páginas 9 e 17.
- HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011. Citado na página 13.
- HOLMES, G.; DONKIN, A.; WITTEN, I. H. Weka: A machine learning workbench. In: IEEE. *Australian New Zealand Intelligent Information Systems Conference*. [S.l.], 1994. p. 357–361. Citado na página 17.
- HUANG, J.; LU, J.; LING, C. X. Comparing naive bayes, decision trees, and svm with auc and accuracy. In: IEEE. *International Conference on Data Mining*. [S.l.], 2003. p. 553–556. Citado na página 14.
- INDYK, P.; KOUDAS, N.; MUTHUKRISHNAN, S. Identifying representative trends in massive time series data sets using sketches. In: *VLDB*. [S.l.: s.n.], 2000. p. 363–372. Citado na página 12.
- JoKo. *XComp/XPack: A freeware PE32-imagefile packer/rebuilder*. 2007. Disponível em: <<http://www.soft-lab.de/joko/ExePack.htm>>. Citado na página 22.
- KELLER, J. M.; GRAY, M. R.; GIVENS, J. A. A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, IEEE, n. 4, p. 580–585, 1985. Citado na página 13.
- KLOVE, T. et al. Permutation arrays under the chebyshev distance. *IEEE Transactions on Information Theory*, IEEE, v. 56, n. 6, p. 2611–2617, 2010. Citado na página 44.
- KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: IJCAI. *International Joint Conferences on Artificial Intelligence*. [S.l.], 1995. v. 14, n. 2, p. 1137–1145. Citado na página 13.
- KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, v. 160, p. 3–24, 2007. Citado na página 16.
- LIU, B.; HSU, W.; MA, Y. Mining association rules with multiple minimum supports. In: ACM. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 1999. p. 337–341. Citado na página 12.
- MAIMON, O.; BROWARNIK, A. *Data mining and knowledge discovery handbook*. Springer, 2009. Citado na página 12.

- MATCODE Software. *MATCODE comPRESSor for executables*. 2007–2010. Disponível em: <<http://northfox.uw.hu/>>. Citado na página 22.
- MERIGÓ, J. M.; GIL-LAFUENTE, A. M. Using the owa operator in the minkowski distance. *International Journal of Computer Science*, v. 3, n. 3, p. 149–157, 2008. Citado na página 44.
- MURPHY, K. P. Naive bayes classifiers. *University of British Columbia*, 2006. Citado na página 14.
- Northfox. *Mew 11 SE*. 2004. Disponível em: <<http://northfox.uw.hu/>>. Citado na página 22.
- OBERHUMER, M.; MOLNAR, L.; REISER, J. *The Ultimate Packer for eXecutables*. 1996–2010. Disponível em: <<http://upx.sourceforge.net>>. Citado na página 22.
- Oreans Technologies. *Themida - Advanced Windows Software Protection System*. 2009. Disponível em: <<https://www.oreans.com/themida.php>>. Citado na página 23.
- PARK, D. S. K. J. B.; RUIZ, R.; MONTES, A. Ferramenta para reconhecimento de executáveis empacotados. In: *International Conference On Forensic Computer Science (ICoFOCS)*. [S.l.: s.n.], 2011. p. 173–181. Citado 7 vezes nas páginas 8, 9, 18, 20, 22, 23 e 37.
- PARK, K.; RUIZ, R.; MONTES, A. BinStat Tool for Recognition of Packed Executables. *The International Journal of Forensic Computer Science*, v. 6, n. 1, p. 44–58, 2011. ISSN 18099807. Disponível em: <<http://www.ijofcs.org/abstract-v06n1-pp03.html>>. Citado 3 vezes nas páginas 8, 23 e 28.
- PETERSON, L. E. K-nearest neighbor. *Scholarpedia*, v. 4, n. 2, p. 1883, 2009. Citado na página 13.
- PIATETSKY-SHAPIO, G. Knowledge discovery in personal data vs. privacy: A mini-symposium. *Intelligent Systems*, IEEE, n. 2, p. 46–47, 1995. Citado na página 11.
- QUINLAN, J. R. C4. 5: Programming for machine learning. *Morgan Kauffmann*, v. 38, p. 48, 1993. Citado 2 vezes nas páginas 19 e 37.
- QUINLAN, J. R. Data mining tools see5 and c5. 0. <http://www.rulequest.com/see5-info.html>, Rulequest Research, 2004. Citado 5 vezes nas páginas 9, 19, 20, 22 e 30.
- RAY, S. *6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python)*. 2015. <<https://www.analyticsvidhya.com/blog/2015/09/naive-bayes-explained/>>. (Accessed on 07/10/2017). Citado na página 14.
- RISH, I. An empirical study of the naive bayes classifier. In: IBM. *IJCAI 2001 workshop on empirical methods in artificial intelligence*. [S.l.], 2001. v. 3, n. 22, p. 41–46. Citado na página 14.
- ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 15.

- SHARMA, A. *How to create a perfect Decision Tree*. 2015. <<https://www.edureka.co/blog/decision-trees/>>. (Accessed on 06/29/2017). Citado na página 17.
- SHERWOOD, T. et al. Automatically characterizing large scale program behavior. In: ACM. *ACM SIGARCH Computer Architecture News*. [S.l.], 2002. v. 30, n. 5, p. 45–57. Citado na página 44.
- TABISH, S. M.; SHAFIQ, M. Z.; FAROOQ, M. Malware detection using statistical analysis of byte-level file content. In: ACM. *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*. [S.l.], 2009. p. 23–31. Citado 2 vezes nas páginas 8 e 18.
- WAIKATO, M. L. G. at the University of. *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. 2011. <http://www.cs.waikato.ac.nz/~ml/weka/gui_explorer.html>. (Accessed on 06/29/2017). Citado na página 18.
- WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016. Citado na página 17.
- WITTEN, I. H. et al. Weka: Practical machine learning tools and techniques with java implementations. University of Waikato, 1999. Citado na página 17.
- WU, X. et al. Top 10 algorithms in data mining. *Knowledge and information systems*, Springer, v. 14, n. 1, p. 1–37, 2008. Citado na página 9.
- YAN, W.; ZHANG, Z.; ANSARI, N. Revealing packed malware. *ieee seCurity & PrivaCy*, IEEE, v. 6, n. 5, 2008. Citado 2 vezes nas páginas 8 e 18.

Apêndices

APÊNDICE A – Cálculos Estatísticos

- Índice de Simpson

$$S_{b_i} = \frac{\sum_{j=0}^n k_f (k_f - 1)}{N(N - 1)} \quad (\text{A.1})$$

Fonte: (COVER; THOMAS, 2012)

- Distância de Canberra

$$d_C(X, Y) = \sum_{i=1}^n \frac{|x_i - y_i|}{|x_i| + |y_i|} \quad (\text{A.2})$$

Fonte: (BUGATTI, 2008)

- Distância de Ordem de Minkowski

$$d_{\text{wm}}(A, B) = \left(\sum_{i=1}^n w_i |a_i - b_i|^\lambda \right)^{1/\lambda} \quad (\text{A.3})$$

Fonte: (MERIGÓ; GIL-LAFUENTE, 2008)

- Distância de Manhattan

$$\text{ManhattanDist}(a, b) = \sum_{i=1}^D |a_i - b_i| \quad (\text{A.4})$$

Fonte: (SHERWOOD et al., 2002)

- Distância de Chebyshev

$$d_{\max}(\pi, \sigma) = \max \{ |\pi_j - \sigma_j| \mid 1 \leq j \leq n \} \quad (\text{A.5})$$

Fonte: (KLOVE et al., 2010)

- Distância de Bray Curtis

$$BC_{b_i} = \frac{\sum_{f=0}^n |X_f - X_{f+1}|}{\sum_{f=0}^n (X_f + X_{f+1})} \quad (\text{A.6})$$

Fonte: (COVER; THOMAS, 2012)

- Separação Angular

$$AS_{b_i} = \frac{\sum_{f=0}^n X_f \cdot X_{f+1}}{\left(\sum_{f=0}^n X_f^2 \cdot \sum_{f=0}^n X_{f+1}^2 \right)^{1/2}} \quad (\text{A.7})$$

Fonte: (COVER; THOMAS, 2012)

- Coeficiente de Correlação

$$CC_{b_i} = \frac{\sum_{j=0}^n (X_f - \bar{X}_{b_i}) \cdot (X_{f+1} - \bar{X}_{b_i})}{\left(\sum_{f=0}^n (X_f - \bar{X}_{b_i})^2 \cdot \sum_{f=0}^n (X_{f+1} - \bar{X}_{b_i})^2 \right)^{1/2}} \quad (\text{A.8})$$

Fonte: (COVER; THOMAS, 2012)

- Entropia

$$E(R) = - \sum_{v \in \Delta_n} t(r_v) \log_2 t(r_v) \quad (\text{A.9})$$

Fonte: (COVER; THOMAS, 2012)

- Divergência de Kullback – Leibler

$$KL_{b_i}(X_f \| X_{f+1}) = \sum_{f=0}^n X_f \log \frac{X_f}{X_{f+1}} \quad (\text{A.10})$$

Fonte: (COVER; THOMAS, 2012)

- Divergência de Jensen-Shannon

$$\text{JSD}(P\|Q) = \frac{1}{2}\text{D}(P\|M) + \frac{1}{2}\text{D}(Q\|M) \quad (\text{A.11})$$

Fonte: (FUGLEDE; TOPSOE, 2004)

com

$$M = \frac{1}{2}(P + Q) \quad (\text{A.12})$$

Fonte: (FUGLEDE; TOPSOE, 2004)

- Divergência de Itakura – Saito

$$BF_{b_i}(X_f, X_{f+1}) = \sum_{f=0}^n \left(\frac{X_f}{X_{f+1}} - \log \left(\frac{X_f}{X_{f+1}} \right) - 1 \right) \quad (\text{A.13})$$

Fonte: (COVER; THOMAS, 2012)

- Variação Total

$$\delta_{b_i}(X_f, X_{f+1}) = \frac{1}{2} \sum_f |X_f - X_{f+1}| \quad (\text{A.14})$$

Fonte: (COVER; THOMAS, 2012)