FlexRank: Um Rankeador Lexicográfico Rápido

Lucas de Souza Rodrigues

SERVIÇO DE PÓS-GRADUAÇÃO DA UFMS

Data de Depósito:

Assinatura:_____

Lucas de Souza Rodrigues

Orientador: *Prof. Dr. Edson Takashi Matsubara*
Co-orientador: *Prof. Dr. Bruno Magalhães Nogueira*

Tese apresentada a Universidade Federal de Mato Grosso do Sul como parte dos requisitos necessários à obtenção para do título em Mestre em Ciências de Computação.

UFMS - Campo Grande
Julho/2016

Dedicatória

À minha esposa,
Eglem,

À minha família.

Agradecimentos

Toda conquista é fruto de muito esforço e dedicação! Neste tempo tive a oportunidade de ampliar meus conhecimento e horizontes na área da computação. Foi um aprendizado constante no qual, tive o prazer em conhecer professores comprometidos em seus ensinamentos. Em especial agradeço a Deus por ter me sustentado até aqui, pois: "*Bem-aventurado o homem que acha sabedoria, e o homem que adquire conhecimento...*" (Provérbios 3:13).

Agradeço a realização deste sonho, primeiramente a minha esposa, que me auxiliou em todo esse processo, foi uma companheira inestimável. Sua ajuda nos pequenos detalhes com certeza, foram essenciais para essa concretização e mesmo em minhas dificuldades, foi quem esteve ao meu lado me encorajando a superar cada obstáculo. Ao meu pai, Luis Carlos ou Giovani (nome de guerra), um grande militar reconhecido por sua dedicação em tudo que faz, um pai exemplar, que sempre abriu mão de tudo para dar o melhor aos seus filhos. Minha mãe, Laiz, um mãe dedicada, exemplo de mulher, que sempre foi a base da nossa família, trazendo amor e harmonia para nós. A minhas irmãs, Priscila e Fernanda, pela ajuda durante este período, vocês são especiais e sem dúvida as melhores irmãs. Aos meus sogros, Edno e Elizabeth, por não medirem esforços durante esse tempo de viagens e estudos.

Aos meus amigos de quitinete, Cleison e Fabio, pelos momentos de descontração e parceria, sem vocês esta etapa seria muito mais difícil. Os colegas do LIA, Arthur, Marcos, Kymberlim, Shi, Lucas, Adolfo, Eliseu, Bruno, Yuri, Adauto e Edison, obrigado pelo apoio nos estudos, em especial, pela Daiane e Eduardo que sempre me auxiliaram nos experimentos e resultados. A FACOM, por ceder toda sua infraestrutura para o desenvolvimento deste trabalho. A UFGD, pela concessão do afastamento para que fosse possível a realização deste mestrado.

Agradeço também, ao professor Edson Takashi, uma pessoa dedicada à pesquisa, com uma visão impressionante sobre cada aspecto que a ciência pode proporcionar. Sua percepção em lidar com problemas complexos de uma

maneira simples, fez com que este trabalho fosse desafiador e ao mesmo tempo prazeroso. Muito obrigado pelos dias e noites de estudos, pesquisa, correção e apoio durante essa fase. Ao professor Bruno, pela coorientação e ajuda durante os experimentos realizados neste trabalho.

Enfim, a todos que direta ou indiretamente me ajudaram neste trabalho meus sinceros agradecimentos.

Resumo

O uso de Aprendizado de Máquina (AM), tem sido amplamente utilizado em problemas reais nos últimos anos. Este trabalho propõe o uso de técnicas em AM para problemas com dados textuais, com abordagem em algoritmos baseados em regras lexicográficas e legitimamente *rankeadores*. Com a popularização dos dados em meio digitais, torna-se interessante aplicar técnicas de AM para melhor organizar as informações contidas neste vasto campo de bases textuais. O aprendizado supervisionado, uma área de AM, com uso de algoritmos de *rankeamento* é uma alternativa viável para ambientes que possuem poucos dados rotulados. Logo, para alcançar os desafios deste trabalho é proposto o algoritmo FLEXRANK que tem o objetivo de *rankear* conjuntos textuais massivos. Para realizar tal feito FLEXRANK conta com uma estratégia simples que utiliza apenas atributos relevantes e por conseguinte realiza lexicograficamente a ordenação dos exemplos em um conjunto de dados. Deste modo, inicialmente são apresentados os tipos de algoritmos de AM, medidas de avaliação em algoritmos de classificação, *rankeamento* e abordagem dos algoritmos LEXRANK e FLEXRANK proposto neste trabalho. Trabalhos que possuem correlação de ranking de textos, especialmente aqueles que atuam em mineração de textos, são abordados neste estudo. Destaca-se também estudos anteriores com foco a balizar os experimentos e resultados acalçados ao longo deste trabalho. FLEXRANK foi avaliado empiricamente sobre uma série de conjuntos de dados em comparação com os algoritmos SVM, Árvores de Decisão, Naive Bayes, KNN e LEXRANK. Os resultados demonstram que para os problemas de classificação de textos massivos, FLEXRANK tem resultados comparáveis, por meio de Curva ROC AUC, a SVM e mais rápido do que Árvores de Decisão para classificar novos exemplos.

Palavras-chave: aprendizado supervisionado, *ranking*, *rankeamento* lexicográfico, seleção de atributos, classificação textual

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AM Aprendizado de Máquina

ANS Aprendizado Não Supervionado

AS Aprendizado Supervisionado

ASS Aprendizado Semissupervisionado

AUC *Area Under ROC Curve*

CV *CrossValidation*

DT Árvore de Decisão

FN Falso Negativo

FP Falso Positivo

IA Inteligência Artificial

KNN K-Vizinhos mais Próximos

LIA Laboratório de Inteligência Artificial

LR *Likelihood Ratio*

MAP *Maximum a Posteriori*

MT Mineração de Texto

NB *Naive Bayes*

ODDS *Odds Ratio*

ROC *Receiver Operating Characteristic*

SVM *Máquina de Vetores de Suporte*

TFP Taxa de Falso Positivo

TVP Taxa de Verdadeiro Positivo

VN Verdadeiro Negativo

VP Verdadeiro Positivo

Lista de Algoritmos

CAPÍTULO

1

Introdução

Uma das principais funções da mente humana é a tomada de decisão. A decisão normalmente analisa um conjunto de características (atributos) que podem ser ordenados segundo um critério de preferência (importância). A ordem destes critérios de preferências auxiliam o processo de tomada de decisão. Raramente algoritmos de aprendizado máquina fazem uso direto destes critérios. O presente trabalho mostra uma maneira interessante da aplicação dos *modelos de preferência lexicográfica* que utilizam raciocínio lexicográfico (Yaman et al., 2011).

Para ilustrar o funcionamento de modelos de preferência lexicográfica, suponha o seguinte problema de decisão: Dado um conjunto de possíveis destinos $D \in$ {Salvador, Recife, Rio de Janeiro, Porto de Galinhas, Balneário Camburiú, Fernando de Noronha}, onde o objetivo é ordenar os destinos por ordem de preferência. Diversos critérios podem ser definidos como custo da viagem menor que o orçamento (*orçamento*), voos que podem ser pagos com milhas aéreas (*milhas*) e temperatura maior que 30 graus (*quente*) no período da viagem. O principal critério de preferência para a decisão é o *orçamento*, o segundo é *milhas* e o terceiro é *quente*. Os três critérios são binários e o *valor de preferência* é zero (0). Assim, o problema pode ser mapeado conforme a Tabela 1.1, na qual o valor zero (0) representa o valor de preferência. No atributo *orçamento*, zero indica custo de viagem menor que o orçamento e um (1) indica custo de viagem maior que o orçamento. No atributo *milhas*, zero representa destino com vôos que podem ser pagos com milhas e um representa caso contrário. No atributo *quente*, zero representa temperatura maior que 30 graus e um caso contrário.

Tabela 1.1: Destinos de viagem por preferência.

Destino	*orcamento*	*milhas*	*temperatura*
Balneário Camburiú	1	1	0
Fernando de Noronha	1	1	0
Porto de Galinhas	0	0	1
Recife	0	1	0
Rio de Janeiro	1	0	1
Salvador	0	0	0
Destinos ordenados lexicograficamente			
Salvador	0	0	0
Porto de Galinhas	0	0	1
Recife	0	1	0
Rio de Janeiro	1	0	1
Balneário Camburiú	1	1	0
Fernando de Noronha	1	1	0

Ao ordenar lexicograficamente utilizando zeros e uns, a ordem produzida representa a ordem de preferência pelo algoritmo de aprendizado de *rankings*. Os empates, no ranking lexicográfico, fazem a ordem lexicográfica utilizar critérios secundários, refinando a ordenação e produzindo *rankings* de melhor qualidade.

Logo, a utilização de modelos de preferências no *rankeamento* auxiliam o processo de ordenação dos exemplos, como também reproduzem uma decisão próxima a de um ser humano na escolha dos destinos de viagem conforme representação da Tabela 1.1.

Definido o funcionamento de modelos de preferência lexicográfica, considere utilizar este modelo em situações que precisam ordenar um fluxo grande de exemplos e que necessitam de um baixo tempo de *rankeamento*. A categorização de páginas web sob a perspectiva de um filtro que realize o bloqueio ou liberação de páginas é um problema interessante de ser abordado. Proxy, Figura 1.1, é uma ferramenta que possui opções para canalizar o tráfego de conteúdo da internet, as quais destacam-se: realizar caching de páginas web com alto índice de acessos, otimizar a velocidade de acesso dentro de uma rede interna e restringir o acesso a usuários que não possuem permissão a tal conteúdo da web (Tanenbaum, 2003).

Neste processo de decisão o uso indireto de um rankeador torna-se relevante para eleger páginas de maior preferência e de menor preferência a uma determinada categoria de assuntos (ex: política, violência, esporte, drogas) (Matsubara, 2008). Caracterizado por um processo em que tem como entrada

Figura 1.1: Proxy

o conteúdo textual da página requisitada e retorna um rótulo para página com base na preferência lexicográfica. Por fim, se a página solicitada for permitida é feita a liberação, caso contrário é bloqueado seu acesso.

Logo, problemas com relação de ordem baseados em preferências, são equivalentes a modelos de preferências lexicográficas (Schmitt and Martignon, 2006) e (Flach and Matsubara, 2007), na qual o conjunto de variáveis de entrada do problema, representam o critério de preferência utilizado para ordenar exemplos. Modelos de preferência lexicográficas possuem diversas aplicações, entre as quais vale ressaltar a ordenação de grandes volumes de dados textuais de acordo com alguma preferência.

No trabalho de Rigo (2013) realizado no Laboratório de Inteligência Artificial (LIA) da FACOM, são descritos métodos para classificação de páginas de internet. O projeto realizado tinha como objetivo fazer a moderação de acesso de grandes volumes de páginas de internet. A moderação poderia ser utilizada para bloquear o acesso de páginas indevidas. No âmbito institucional, pode-se desejar bloquear acesso a redes sociais, no âmbito familiar, pode-se desejar bloquear acessos de páginas com violência, temas homofóbicos e pornografia. O trabalho de Rigo (2013) utilizou aprendizado de máquina semissupervisionado ativo para a indução de classificadores. Entretanto, a abordagem utilizada não era factível de uso em um ambiente real de grande volume de dados devido ao tempo de classificação proibitivo à aplicação.

O presente trabalho oferece uma possível solução para resolver o problema de tempo de classificação utilizando aprendizado de *rankings*.

1.1 Objetivos e Hipóteses

O objetivo deste trabalho é desenvolver um algoritmo de *ranking* lexicográfico, com a função de ordenar conteúdos textuais (ex: páginas de internet, bases textuais) que tenha como principal característica um baixo tempo de ordenação.

A hipótese deste trabalho, sugere ser possível adaptar o algoritmo LexRank (Matsubara, 2008), utilizando somente atributos (critérios de preferên-

cia) relevantes à ordem de preferência. Logo, espera-se que os algoritmos investigados neste trabalho, sejam suficientes para compor o núcleo de um sistema de classificação de conteúdos textuais (ex: bases textuais, páginas de internet, e-mails, fóruns) em problemas de dados massivos.

1.2 Organização

O trabalho a seguir é dividido em: Capítulo 2 que trata assuntos elementares sobre o Aprendizado de Máquina e seus conceitos. No Capítulo 3 são descritos detalhadamente os algoritmos LexRank Matsubara (2008) e FlexRank, que fazem jus a construção de algoritmos estritamente rankeadores com abordagem lexicográfica. No Capítulo 4 é abordada a análise experimental dos algoritmos LexRank e FlexRank, etapas de treinamento e teste, metodologia e resultados. Por fim, no Capítulo 5, é apresentada a conclusão do trabalho e quais suas principais contribuições, como também encaminhamentos para possíveis trabalhos futuros.

CAPÍTULO

2

Fundamentação Teórica

Neste capítulo são abordados tópicos referentes ao Aprendizado de Máquina, seus conceitos, aplicações e formalização dos principais métodos. São expostos, também, os tipos de categorias do Aprendizado de Máquina, Algoritmos de Aprendizado Supervisionado, Métodos de Avaliação e Trabalhos Relacionados a este estudo.

2.1 Notações

A representação de Zhu (2005), define uma coleção de exemplos (ou atributos) $X = (x_1, x_2, ..., x_{n_{ex}})$, na qual $x_{n_{ex}}$ representa o número de exemplos. Para problemas de classificação, existem ainda os valores do atributo classe, definido por $Y = (y_1, y_2, ..., y_{n_{ex}})$, na qual $y_{n_{ex}}$ representa o número de classes. A junção entre exemplo $x_{n_{ex}}$ e sua respectiva classe $y_{n_{ex}} \in \{Cl_1, .., Cl_{n_{ex}}\}$ é descrita pelo conjunto $\{(x_1, y_1), ..., (x_{n_{ex}}, y_{n_{ex}})\}$. Logo, seja $x \in X$ e $y \in Y$, sendo representados por valores contínuos, discretos ou booleanos.

Esta notação de Zhu (2005) pode ser utilizada entre os algoritmos de AM para representação de conceitos e modelos. Por exemplo para um problema de aprendizado supervisionado, a função $h : X \rightarrow Y$ é induzida, com objetivo de predizer uma classe y_i a um novo exemplo x_i.

Zhu (2005), define também a notação de dimensionalidade de atributos, representado por um vetor de atributos de $x_{n_{atr}}$ dimensões $x_i = (x_{i1}, ..., x_{in_{atr}})$ com $i = (1, ..., n_{ex})$, no qual cada dimensão é representada por um atributo do exemplo.

Na Tabela 2.1 é apresentada uma representação para o formato atributo-valor com n_{ex} exemplos x_i . Logo, o formato x_{ij} descreve o valor do atributo j

para exemplo x_i. Os valores para a classe y_i, que dependendo do caso pode existir ou não (aprendizado supervisionado ou não), descreve o valor do atributo classe pertencente ao conjunto Y .

Tabela 2.1: Exemplos no formato atributo valor.

	x_{i1}	x_{i2}	$\cdots$	$x_{in_{atr}}$	Y_{classe}
x_1	x_{11}	x_{12}	$\vdots$	$x_{1n_{atr}}$	y_1
x_2	x_{21}	x_{22}	$\vdots$	$x_{2n_{atr}}$	y_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_{n_{ex}}$	$x_{n_{ex}1}$	$x_{n_{ex}2}$	$\vdots$	$x_{n_{ex}n_{atr}}$	$y_{n_{ex}}$

2.2 Aprendizado de Máquina

Para descrever Aprendizado de Máquina (AM), é importante compreender o termo "*aprendizado*" em sua essência, para que inferências feitas no campo da computação tenham relação direta com a definição proposta na literatura. Aprendizado é descrito na literatura como "*efeito ou consequência de aprender*" (Dicio, 2015). Aprender, por sua vez, tem como resultado de ação a aquisição de conhecimento. Segundo estudos de Skinner (1972), aprendizagem também concentra-se na capacidade de estimular ou reprimir comportamentos, desejáveis ou indesejáveis.

No entanto, estas definições descrevem de forma sintetizada que o resultado obtido pela aprendizagem é a geração de conhecimento e descrição de comportamentos (Michaelis, 2015). Porém, quando analisadas sob a perspectiva da computação, torna-se limitado o conceito, uma vez que máquinas não possuem raciocínio próprio sobre o conhecimento adquirido.

No AM, uma área da Inteligência Artificial, o conceito para aprendizado possui um contexto diferente, pois faz uso de algoritmos, métodos e técnicas que permitem ao computador aprender e tomar suas próprias decisões. Logo, a associação realizada por um computador para aferir seu aprendizado está intimamente ligada às alterações de comportamento, ou seja, ajustes realizados nos métodos de aprendizagem que obtêm melhora de performance (Ian H. Witten, 2011).

Nesse contexto, o AM engloba diversas áreas da ciência para realização de suas ações, como o uso da matemática, estatística, biologia, complexidade computacional, filosofia e análise cognitiva (Mitchell, 1997).

Em AM, inicialmente é realizada uma coleta de dados e definidos os dados para uma linguagem de descrição. Logo em seguida, uma hipótese é induzida

utilizando instâncias. Após estes passos, a hipótese pode ser recomendada a um problema. No entanto, muitas vezes é necessário refazer estas etapas ajustando comportamentos para um melhor desempenho.

Para exemplificar uma aplicação em AM, considere o desenvolvimento de um sistema em execução sobre a qualidade do dia para viajar *(Clima)*. Cada linha na Tabela 2.2, representa um dia e suas condições climáticas. O objetivo é induzir uma hipótese que classifique o dia em "*Bom*" ou "*Ruim*" para viajar.

Tabela 2.2: Clima - Sugestão de Viagem (Mitchell, 1997).

Exemplo	Aparência	Temperatura	Umidade	Ventando	Viajar
X_1	sol	25	72	sim	bom
X_2	sol	28	91	sim	ruim
X_3	sol	22	70	não	bom
X_4	sol	23	95	não	ruim
X_5	sol	30	85	não	ruim
X_6	nublado	23	90	sim	bom
X_7	nublado	29	78	não	bom
X_8	nublado	19	65	sim	ruim
X_9	nublado	26	75	não	bom
X_{10}	nublado	20	87	sim	bom
X_{11}	chuva	22	95	não	bom
X_{12}	chuva	19	70	sim	ruim
X_{13}	chuva	23	80	sim	ruim
X_{14}	chuva	25	81	não	bom
X_{15}	chuva	21	80	não	bom

Contudo, para que um ciclo de aprendizagem seja executado de maneira efetiva é necessário estabelecer alguns critérios que podem nortear o sistema de aprendizado. Conforme descreve Mitchell (1997), há 3 elementos básicos necessários ao AM:

- Uma tarefa T ;
- Uma medida de performance P ;
- Uma experiência E.

"Logo, diz-se que um programa de computador ***aprende*** *melhorando a experiência* ***E*** *em relação a uma classe de tarefas* ***T*** *mensurado pela performance* ***P****. Em outras palavras o programa aperfeiçoou a execução de sua tarefa"* (Mitchell, 1997)

Para o exemplo *(Clima)* os elementos descritos acima são representados da seguinte forma:

- Uma tarefa T = Previsão do Tempo;
- Uma medida de performance P = Número de dias que foram preditos como um bom dia para viajar e que realmente o eram;

- Uma experiência E = Conjunto de dias passados em que uma pessoa classificou como um bom dia para viajar ou não;

Assim os elementos básicos que representam o AM são descritos por um conjunto de algoritmos, que são divididos dentro do Aprendizado Supervisionado, Não Supervisionado e Semissupervisionado, abordados nas seções posteriores.

2.3 Aprendizado Não Supervisionado

O ANS pertence a uma classe de problemas na qual conjuntos de exemplos não possuem rótulo, representados apenas por exemplos $X = (x_1, x_2, ..., x_{n_{ex}})$. O objetivo dos algoritmos pertencentes a este paradigma é a obtenção de modelos que descrevem os dados (modelos descritivos). Este tipo de problema é muito comum em sistemas que envolvem grande quantidade de dados que não possuem informação externa sobre os exemplos coletados.

A característica fundamental deste aprendizado está na observação dos exemplos não rotulados. O algoritmo procura agrupar exemplos utilizando uma medida de similaridade. Porém é necessário ressaltar que o agrupamento é apenas uma das técnicas utilizadas em aprendizado não supervisionado.

O aprendizado não supervisionado, tem o objetivo de obter informações que caracterizem certa regularidade. Entre as tarefas mais frequentes destacam-se:

- **Agrupamento** - Reunir exemplos semelhantes em agrupamentos, utilizando uma medida de similaridade;
- **Detecção de Singularidade** - Verificar exemplos que possuem diferenças dos demais no conjunto de dados;
- **Redução de Dimensionalidade** - Reduzir o número de atributos sem perder as características dos dados;
- **Regras de Associação** - Encontrar exemplos que ocorrem em comum dentro de um conjunto de dados.

2.4 Aprendizado Supervisionado

A característica principal do AMS é a utilização de um conjunto de exemplos rotulados para indução de uma hipótese (classificador). Logo, a finalidade do classificador é predizer a classe ou categoria de um determinado exemplo.

Para o exemplo citado na Tabela 2.2, a classe é definida por dois valores, $Y = \{Ruim, Bom\}$. Estes atributos normalmente são ajustados a uma linguagem

computacional, assim rótulos nominais transformam-se em valores numéricos (binários, naturais ou reais). Por exemplo, utilizando rótulos binários para o *Clima*, tem-se o resultado: *Bom* = 1 e *Ruim* = 0. Para problemas que possuem *n* rótulos, é necessário relacionar um conjunto numérico de modo que satisfaça a relação $Y_{n_{cl}} = \{1,2,3,...,n_{cl}\}$, na qual n_{cl} é o número de classes (rótulos).

Problemas que utilizam aprendizado supervisionado, são divididos em problemas de **classificação** ou **regressão**.

Classificação - Expressa valores discretos para os atributos relacionados à classe *Y*. Na Tabela 2.2 é apresentada uma coleção de exemplos $X_n = \{1,2,3,...,n_{ex}\}$ que representa a condição para que uma pessoa possa viajar em virtude do tempo. Sua classe *Y* é representada por dois valores Y = *{Ruim, Bom}*, que posteriormente são codificados para linguagem descritiva, por exemplo, $Y = \{0,1\}$. Este formato pode ser compreendido por valores categóricos , que possuem $|Y_n| \geqslant 2$, por exemplo:

1. {Sim, Não}
2. {Positivo, Falso}
3. {Bom, Ruim, Ótimo}
4. {Amarelo, Azul, Verde, Vermelho}

Regressão - Problemas nas quais $Y \in \mathbb{R}$. Na Tabela 2.3 é descrito o *Preço de Imóveis* (tamanho, quantidade de quartos, andares e idade de construção), sua classe *Y* é o *Preço*. O algoritmo de aprendizado define uma função $h(e)$ que aproxime os valores preditos dos preços reais para imóveis no conjunto de dados.

Tabela 2.3: Preço de Imóveis.

Tamanho ($metros^2$)	Qtde Quartos	Qtde Andares	Idade	Preço (US$ mil)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

A indução de uma hipótese de classificação ou regressão é realizada durante o treinamento. As hipóteses induzidas neste processo de treinamento podem sofrer de dois problemas:

Sobre ajuste (*Overfit*) - É o caso onde a indução gera uma hipótese (classificador) super ajustada para o conjunto de treinamento, nestas situações o classificador decora o conjunto de treinamento.

Sub ajuste (*Underfit*) - Neste caso o algoritmo não consegue absorver conhecimento adequadamente, devido ao viés (*bias*) do indutor. Isso ocorre devido a uma condição na qual a indução não consegue gerar uma hipótese satisfatória para novos problemas.

2.5 Aprendizado Semissupervisionado

O Aprendizado Semissupervisionado (ASS) destaca-se em diversos cenários nos quais existe uma grande quantidade de dados não rotulados e um número de exemplos rotulados pequeno ou quase nulo (Zhu, 2005).

Considere um problema em que um sistema gerenciador de e-mails é capaz de classificar mensagens enviadas pela internet para diversas caixas de e-mails de seus usuários. Inicialmente, o sistema recebe diariamente centenas de e-mails e deve classificar cada e-mail baseado em seu conteúdo, com o objetivo de evitar a inclusão de spams na caixa de entrada de seus usuário. Para a realização desta tarefa, o sistema de classificação conta apenas com alguns exemplos já rotulados, ou seja, seu conjunto de treinamento é pequeno quando comparado ao número de novos e-mails que chegam todo dia para classificação.

Para problemas com poucos exemplos rotulados, a aplicação do aprendizado supervisionado, pode deixar a desejar. Assim, a representação para o aprendizado semissupervisionado é definida por um conjunto de $x_{n_{ex}}$ exemplos, na qual é dividido em duas partes: os pontos já rotulados $X_l = (x_1, ..., x_{l_{ex}})$, com seus respectivos rótulos $Y_l = (y_1, ..., y_{l_{cl}})$, e para pontos ainda não rotulados $X_u = (x_{l+1}, ..., x_{l+u_{ex}})$.

Realizar a indução do classificador e filtrar novos exemplos para cenários assim (sistema gerenciador de e-mails), requer técnicas derivadas de conceitos tradicionais na área de AM. Algumas delas são técnicas *bayesianas*, métodos *kernel* e uso de agrupamentos com utilização de conhecimento prévio *(background knowledge)*. Essas técnicas já são utilizadas no aprendizado supervisionado e não supervisionado com aprimoramentos para tratar problemas em um conjunto de dados com poucos exemplos rotulados (Zhu, 2005). No entanto, esses aprimoramentos são realizados por que classificadores induzidos por um pequeno conjunto de exemplos rotulados, geralmente, não apresentam uma boa performance na classificação.

De acordo com Matsubara (2004), a classificação no ASS tem por objetivo rotular mais exemplos com intuito de incrementar o conjunto de exemplos rotulados a um algoritmo de AM supervisionado, assim é possível induzir melhores hipóteses. Logo, é viável realizar a transição de métodos e técnicas na resolução de problemas no AM, para refinar o processo de classificação.

Outro aspecto interessante no ASS é que ele pode ser usado também na geração de agrupamentos, (Basu et al., 2002). Na classificação, sua principal tarefa é rotular novos exemplos de forma segura, para que estes sejam usados posteriormente no conjunto de treinamento para uma classificação mais precisa. Já no agrupamento semissupervisionado, o restrito conjunto de exemplos rotulados ajuda na formação dos agrupamentos, como uma informação adicional que auxilia no processo do cálculo das centróides e aglomeração dos exemplos semelhantes. Alguns exemplos de algoritmos que se destacam neste método de aprendizagem são: COP-*k*-MEANS (Wagstaff et al., 2001), SEEDED-*k*-MEANS e CONSTRAINED-*k*-MEANS (Basu et al., 2002)).

2.6 Algoritmos de Aprendizado de Máquina Supervisionado

Esta seção tem por objetivo descrever alguns algoritmos de AMS, com o propósito de alinhar algoritmos tradicionais da área de AM, juntamente aos algoritmos propostos neste trabalho: LEXRANK (Flach and Matsubara, 2007) e FLEXRANK, para utilização dos mesmos na análise experimental e comparação estatística entre ambos.

2.6.1 K-Vizinhos mais Próximos

O Algoritmo *K-Vizinhos mais Próximos*, conhecido como KNN, foi introduzido por Cover and Hart (1967), é um algoritmo que faz a classificação de elementos ainda desconhecidos, com auxílio das categorias dos K elementos vizinhos mais próximos que já foram rotulados ao corpus de treinamento. Para determinar o rótulo de um novo exemplo X, o algoritmo KNN faz o cálculo dos *K-vizinhos mais Próximos* ao novo exemplo. Assim, faz a classificação dos novos exemplos com a classe Y que aparece com maior frequência entre seus *K-vizinhos*.

Porém, a determinação dos K primeiros elementos que possuem maior similaridade com o novo exemplo X, tem um custo considerável dado o volume dos n elementos a serem comparados dentro do conjunto de treinamento.

Segundo Webb (2003), "KNN *é um método simples para estimação de densidade*". Esta afirmação é estabelecida, pois KNN determina a densidade com base na similaridade dos exemplos locais na etapa de treinamento, que por sua vez são preditos nos exemplos desconhecidos da vizinhança local durante a classificação.

Por fim, para o classificador KNN realizar o cálculo dos possíveis *K-vizinhos*, calcula-se a distância euclidiana que pode ser representada pela Equação 2.1,

no qual x_i e x_j, representam duas instâncias:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{2.1}$$

Sua sequência de passos é definida pelos seguintes critérios:

Etapa de Treinamento

1. Armazenar em memória o conjunto de exemplos de treinamento.

Etapa de Teste

1. Calcular a distância de um novo exemplo em relação aos exemplos do conjunto de treinamento;
2. Identificar os *K-vizinhos* mais próximos ao novo exemplo;
3. Rotular o novo exemplo com a classe majoritária entre os *K-vizinhos* mais próximos.

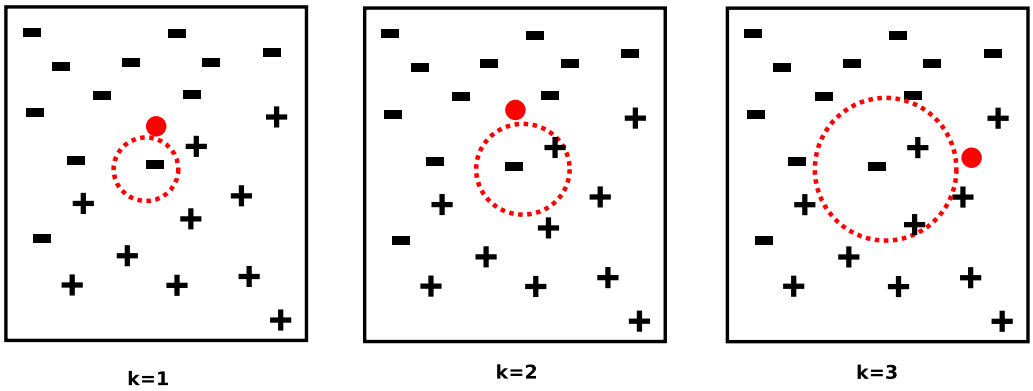

Figura 2.1: Algoritmo *K-Vizinhos Mais Próximos*, com variação dos *K-Vizinhos*.

KNN é um algoritmo que possui fácil implementação, e o parâmetro de ajuste em sua construção mais importante é a variação do número dos K melhores vizinhos, conforme ilustração da Figura 2.1 que descreve o raio de atuação para predição conforme o número de *K-vizinhos* é incrementado. Na ilustração ao predizer um exemplo, caracterizado por um círculo vermelho, para $K = 1$ a classe majoritária será "negativa", para $K = 2$ caso a predição seja pareada o classificador analisa qual melhor hipótese para o novo exemplo e por fim com $K = 3$ o exemplo é classificado como "positivo". No entanto, é necessário um ajuste de parâmetro coeso para obter bons resultados ao conjunto de exemplos sem rótulo, pois se K for pequeno, também será sensível a ruído. Logo, se K for grande, a vizinhança pode incluir pontos de outras classes (Mitchell, 1997).

2.6.2 Naive Bayes

Naive Bayes (NB) é um algoritmo baseado em métodos Bayesianos, (McCallum et al., 1998). São métodos com um tipo de inferência estatística que oferecem uma abordagem probabilística. O aprendizado utiliza inferência estatística, e permite gerar uma probabilidade para hipótese assumida pelo classificador. Porém, conforme Mitchell (1997), algoritmos de AM priorizam a *melhor* probabilidade de um conjunto de exemplos de treinamento, assim a hipótese mais viável em aprendizado Bayesiano é também dita como a *melhor* hipótese.

Para a classificação de novos exemplos, o algoritmo NB faz uso do *Teorema de Bayes* (Barnard and Bayes, 1958), na qual a informação é representada na forma de probabilidade condicional e que possui a seguinte premissa: "Qual a probabilidade de um evento ocorrer dada uma condição? Ou seja, qual a probabilidade de um evento (y), sabendo qual será o resultado de um evento (x)".

Para compreender melhor a condição do *Teorema de Bayes*, é necessário entender os dois tipos de probabilidade: **priori** e **posteriori**. Probabilidade priori, está relacionada a probabilidade de um evento acontecer sem conhecimento prévio de outro evento. Um exemplo para esta propriedade é: a probabilidade de jogar uma moeda e obter o lado $P(cara) = 1/2$. Já a probabilidade posteriori é definida como condicional, ou seja, quando um evento relevante é considerado no processo probabilístico. Logo, para o exemplo da moeda quando jogada n vezes, temos a posteriori que a distribuição dos valores obtidos nas jogadas estão dispostas no previsto a priori. Para este tipo de problema o *Teorema de Bayes* estabelece a Equação 2.2, na qual, $P(y)$ e $P(x)$: são definidas como probabilidades a **priori** de y e x. E, $P(y|x)$: é definida como probabilidade a **posteriori** de x condicional a y:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \tag{2.2}$$

Com base nesta premissa, ao utilizar o *Teorema de Bayes* na etapa de treinamento em NB, obtêm-se a máxima a posteriori (maximum a posteriori ou somente MAP) $vMAP$ dos valores dos atributos relacionados ao conjunto de exemplos, (Matsubara, 2008). Assim, a equação para o cálculo $vMAP$, pode ser descrita utilizando a condição do *Teorema de Bayes*. Onde, $P(y_v)$ é estimada com base na frequência de cada valor $y_v \in Y$, em que aparece nos exemplos de treinamento, da seguinte forma:

$$vMAP = \underset{y_v \in Y}{\arg\max} P(y_v|x_i) = \underset{y_v \in Y}{\arg\max} \frac{P(x_i|y_v).P(y_v)}{P(x_i)} \tag{2.3}$$

Por fim, para que Naive Bayes possa determinar qual a melhor classe de um exemplo, é considerado o produto das probabilidades de cada atributo individualmente, levando-se em conta que os valores dos atributos são condicionalmente independentes:

$$P(x_{i_1}, x_{i_2} ... X_{in_{at}} | y_v) = P(x_{i_1} | y_v) p(x_{i_2} | y_v) ... P(x_{in_{at}} | y_v) = \prod_{j=1}^{n_{at}} p(x_{ij} | y_v) \tag{2.4}$$

Logo, é possível concluir que a hipótese gerada pelo classificador NB, pode ser descrita na Equação 2.5. A qual, omite-se da equação o valor de $P(x_i)$, pois ao realizar o produto das probabilidades de cada atributo, o valor de $P(x_i)$ está presente no cálculo de todos os y_v.

$$v\text{NB} = \arg\max_{y_v \in Y} P(y_v) = \prod_{j=1}^{n_{at}} p(x_{ij} | y_v) \tag{2.5}$$

NB faz uso das frequências analisadas nos exemplos de treinamento para estimar as probabilidades de $P(y_v)$ e $P(x_{ij}|y_v)$, que por sua vez correspondem a hipótese assumida na etapa de treinamento pelo classificador. Por definição Matsubara (2008), NB é caracterizado como um algoritmo de desempenho bom e classificação rápida, pois as hipóteses com dependência entre os atributos nem sempre são consideradas tornando seu espaço de hipótese restrito aos valores obtidos entre os termos $P(y_v)$ e $P(x_i|y_v)$.

2.6.3 Árvore de Decisão

A Árvore de Decisão (AD), utiliza a estratégia dividir-para-conquistar. No algoritmo, os dados são divididos em subconjuntos menores a fim de obter entre as partições exemplos que separam melhor as classes presentes no conjunto de dados. É um processo que ocorre por meio da recursividade em cada subconjunto, com a seleção dos atributos que dividem satisfatoriamente as classes em cada nível da árvore.

O primeiro passo para a construção da AD é definir o atributo que representa o nó da árvore. Esse atributo deve ser selecionado de modo a dividir o conjunto de dados em partições. Cada partição possui todos ou a maioria dos exemplos da mesma classe, como visto na Figura 2.2.

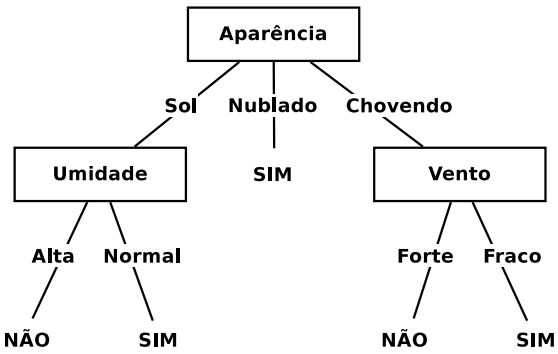

Figura 2.2: Previsão do Tempo, adaptado de: (Mitchell, 1997).

Para medir o grau de "pureza" das partições geradas é realizado o cálculo de entropia de Shannon (Shannon, 2001), conhecido também como cálculo de medida de impuridade. Sua equação para resolução de problemas binários, pode ser descrita pela Equação 2.6. Onde, p_+ é descrita como a fração de exemplos positivos em S e $p_- = 1 - p_+$ com a fração de exemplos negativos.:

$$Entropia(S) = -p_+ log_2 p_+ - p_- log_2 p_- \tag{2.6}$$

A Equação 2.6 comporta-se conforme a Figura 2.3. Suponha utilizar o exemplo da moeda visto na Seção 2.6.2, a probabilidade de obter em uma jogada um dos lados {cara | coroa} é de 50% para cada um dos lados. Logo, p_+ atinge seu valor máximo em 0.5, e seu valor mínimo quando p_+ é igual a 0 ou 1 para um conjunto de exemplos.

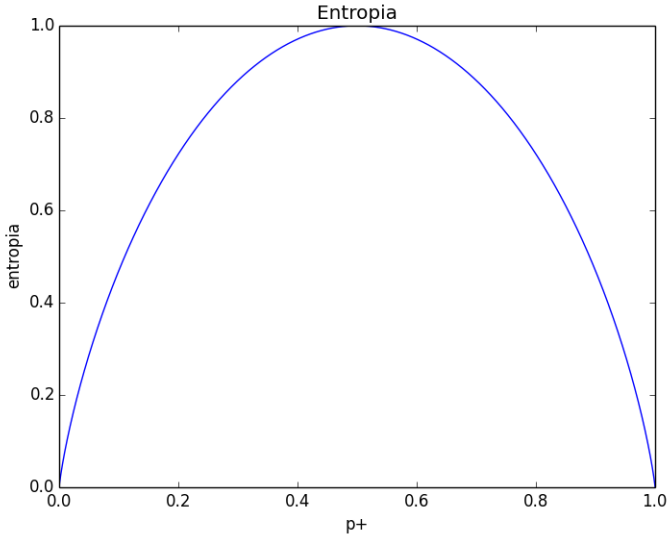

Figura 2.3: Cálculo de Entropia, (Shannon, 2001).

A entropia pode ser considerada uma medida de incerteza. Porém, em casos na qual o conjunto de exemplos é considerado "puro" pertencentes a somente uma classe, a incerteza calculada na entropia é zero. Todavia, realizado o cálculo da entropia é necessário analisar qual o ganho de informação

obtido em cada subconjunto na formação da AD, definido pela Equação 2.7, onde S representa o conjunto de exemplos; v um valor do atributo; $atributo_v$ o subconjunto de S em que o valor desse $atributo$ é v:

$$GanhoInfo(S, atributo) = Entropia(S) - \sum_{v \in Dom(atributo)} \frac{|atributo|}{|S|} \times Entropia(atributo_v) \tag{2.7}$$

Através do ganho de informação, obtém-se o atributo relevante para fazer o particionamento da árvore. Este processo ocorre com a seleção do atributo que possui o maior ganho de informação e assim sucessivamente os demais nós são criados até que a árvore seja construída por completo.

No entanto, é importante ressaltar que não necessariamente Árvores de Decisão sempre usam o cálculo de entropia de Shannon (Shannon, 2001) como parâmetro de ajuste em seu algoritmo classificador. A literatura prevê inúmeras medidas de impurezas para a construção do classificador baseado em árvores de decisão, como por exemplo: DKM (Kearns and Mansour, 1996), índice de Gini (Breiman et al., 1984), AUC (Ferri et al., 2002), Razões de Chances "ODDS RATIO" (Bland and Altman, 2000), dentre outros.

As possíveis vantagens em se utilizar o algoritmo de AD, é a representação gerada pela árvore o que torna o algoritmo versátil para problemas como a mineração de dados. Outro ponto, é a simplicidade na representação da hipótese, o qual não visa avaliar todos os atributos de um exemplo (Mitchell, 1997). Assim, é possível que durante a construção do modelo algumas ramificações que não possuam representação significativa na classificação sejam podadas, tornando o processo de classificação mais rápido.

2.6.4 Máquina de Vetores de Suporte

O último algoritmo de aprendizado de máquina visto neste capítulo é a Máquina de Vetores de Suporte, conhecido também como SVM. É um classificador criado por (Vapnik, 1998) e alguns colaboradores. Em sua implementação padrão, o algoritmo apenas compara um novo exemplo a duas possíveis classes, o que torna um classificador binário.

A ideia do algoritmo consiste em divisão por meio de limites de decisão, que representem uma separação ótima entre classes do conjunto de dados por meio da minimização dos erros (Vapnik, 1998). Sua formulação para um conjunto de dados qualquer é representada por uma série de amostras $D = (x_n, y_n)$ que simbolizam uma coleção de exemplos X_n com suas respectivas classes Y_n, conforme notação descrita na Seção 2.1, na qual $X_n \in \mathbb{R}^M$ representam uma série de pontos vetoriais e $Y_n \in \{-1, 1\}$ as suas respectivas classes.

Para uma melhor compreensão, a Figura 2.4 apresenta a separação ótima

entre as classes por meio de um hiperplano (H), que é utilizado para delimitar a orientação do plano, a fim de maximizar a margem entre as bordas (H_1 e H_2) e o ponto mais próximo entre as classes do problema.

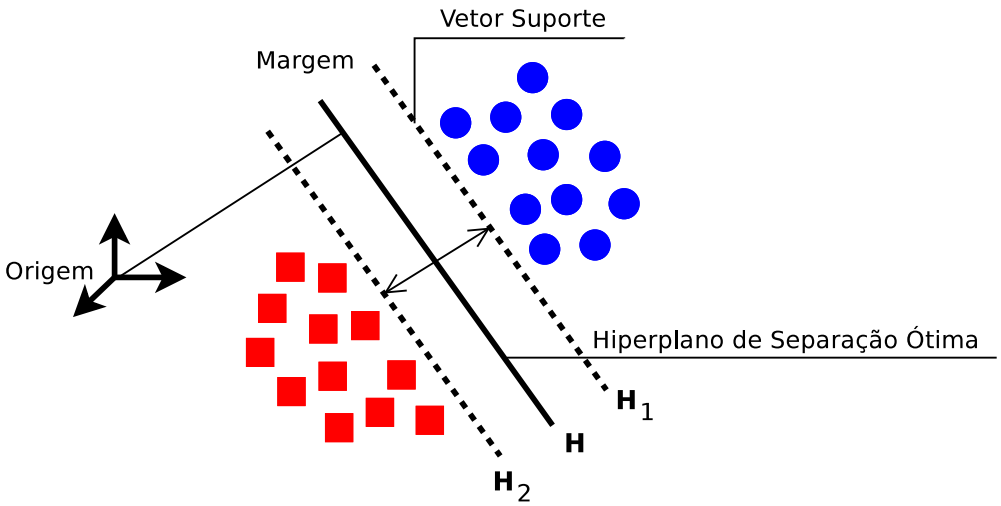

Figura 2.4: Esquema de classificação por meio do SVM, adaptado de (Huang et al., 2002) e (Melgani and Bruzzone, 2004).

Frequentemente é utilizado com as seguintes funções kernel: (i) linear, (ii) quadrática, (iii) polinomial e (iv) função de base radial. SVM é considerado um algoritmo de *alto* desempenho quando utilizado com os parâmetros adequados obtidos por conjuntos de avaliação.

2.7 Medidas de Avaliação dos Algoritmos

Nesta seção são abordadas as medidas de avaliação para que desenvolvedores e pesquisadores da área possam avaliar os modelos e algoritmos de AM.

2.7.1 Matriz de Confusão

Antes de avaliar as medidas de avaliação, é necessário explanar sobre o instrumento utilizado para realizar o cálculo obtido por cada avaliação. A Matriz de Confusão, é representada por uma tabela que armazena os valores de cada instância obtida durante a etapa de classificação do conjunto de dados. Cada coluna desta matriz representa as instâncias previstas em uma classe do conjunto de dados, enquanto que as linhas descrevem as instâncias reais de cada classe do conjunto.

É uma método eficaz para análise, quando existem apenas duas classes em um conjunto, sendo possível generalizar problemas multi-classe com a redução de classes (ex: um contra todos). A Matriz de Confusão é definida por duas linhas e duas colunas com as seguintes situações:

(VP)	verdadeiro positivo	: exemplo positivo	predito como positivo
(FP)	falso positivo	: exemplo negativo	predito como positivo
(VN)	verdadeiro negativo	: exemplo negativo	predito como negativo
(FN)	falso negativo	: exemplo positivo	predito como negativo

As seguintes situações são relacionadas na Tabela 2.4 e dispostas entre os exemplos presentes em um conjunto de dados. Onde,VP, FP, VN e FN representam, respectivamente, o número de exemplos verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos; *Pos* o total de exemplos positivos; *Neg* o total de exemplos negativos; *PPos* o total de exemplos preditos como positivos; *PNeg* o total de exemplos como negativos; *Total* o número total de exemplos no conjunto de exemplos.

Tabela 2.4: Matriz de Confusão

	preditos positivo	preditos negativo	
exemplos positivo	VP	FN	Pos
exemplos negativo	FP	VN	Neg
	PPos	PNeg	Total

2.7.2 PRECISION

A medida de avaliação PRECISION, Equação 2.8, descreve a taxa com que todos os exemplos avaliados como positivos são realmente positivos.

$$\text{PRECISION} = \frac{VP}{FP+VP} \tag{2.8}$$

2.7.3 RECALL *ou* TVP

Enquanto PRECISION avalia os exemplos preditos, RECALL avalia entre os exemplos positivos quantos realmente foram preditos como positivos. Logo, expressa a qualidade de classificação da classe positiva.

$$\text{RECALL} = \frac{VP}{FN+VP} \tag{2.9}$$

2.7.4 TFP

Expressa a qualidade de classificação da classe negativa, no entanto avalia a classe com uma taxa de erro. Assim quanto menor o valor, melhor será a classificação da classe negativa.

$$\text{TFP} = \frac{FP}{FP+VN} \tag{2.10}$$

2.7.5 F1-MEASURE

A medida de avaliação F1-MEASURE possui como estimativa a média harmônica obtida entre os valores RECALL e PRECISION e sua fórmula pode ser definida na Equação 2.11.

$$\text{F1-MEASURE} = \frac{2 * \text{RECALL} * \text{PRECISION}}{\text{RECALL} + \text{PRECISION}} \tag{2.11}$$

2.7.6 ACCURACY

ACCURACY é baseada na proporção das instâncias classificadas corretamente pelo volume total de instâncias do conjunto. Sua formulação é definida pela Equação 2.12.

$$\text{ACCURACY} = \frac{TP+TN}{TP+TN+FN+FP} \tag{2.12}$$

2.7.7 *Análise* ROC

O gráfico ROC (*Receiver Operating Characteristic*), Figura 2.5, é um espaço bidimensional com os eixos X e Y representados respectivamente por TFP (Taxa de Falso Positivos) e TVP (Taxa de Verdadeiro Positivos). A linha diagonal representa um classificador aleatório. Os pontos inseridos no gráfico descrevem a curva ROC de um classificador, uma técnica para a visualização e seleção de classificadores baseado no seu desempenho, é uma ferramenta que permite estudar a variação da sensibilidade e especificidade de um classificador. O Céu ROC, representado pelos pontos próximos a (0,1), descrevem uma classificação próxima da ideal na qual todos exemplos são preditos corretamente. Em contrapartida, o Inferno ROC representado pelos pontos próximos a (1,0), estabelece a condição inversa ao Céu ROC sendo registrado os piores resultados para um classificador.

Para testes comparativos entre classificadores no gráfico ROC, a medida utilizada para validação estatística é a AUC (*Area Under ROC Curve*) área abaixo da curva ROC. Logo, quanto maior a curva dentro do espaço ROC,

maior será o valor de AUC (Area Under ROC Curve).

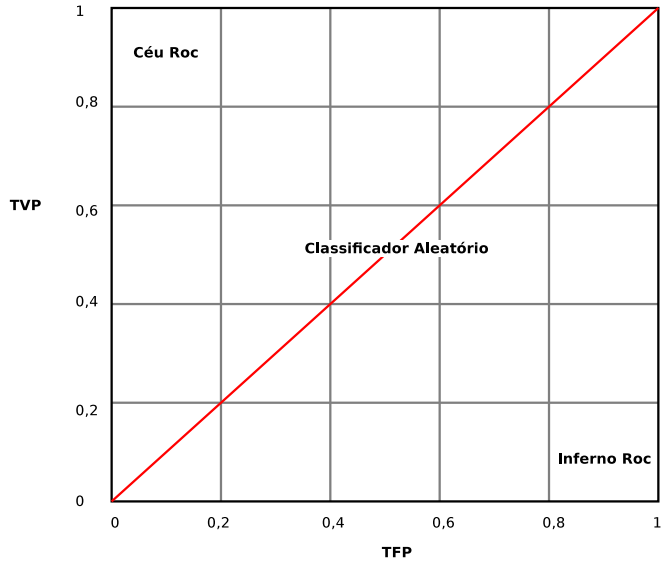

Figura 2.5: Gráfico ROC adaptado de Flach (2003).

A grandeza escalar de AUC abrange os limites ($\lim = 0 \leftrightarrow 1$), sendo 1 o maior valor obtido por um classificador aleatório. A Figura 2.6, apresenta a construção da curva ROC para o dataset diabetes (Asuncion and Newman, 2007), utilizando um classificador aleatório para n folds, em que consiste dividir o conjunto total em n subconjuntos de mesmo tamanho, na qual um subconjunto é utilizado para testes e os $n-1$ subconjuntos são usados na etapa de treino.

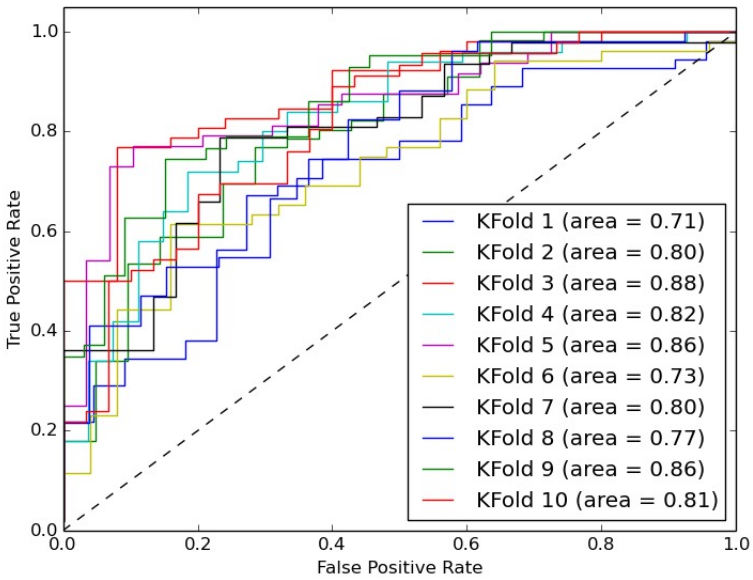

Figura 2.6: Gráfico ROC *dataset diabete* (Asuncion and Newman, 2007).

2.8 Aplicações dos Algoritmos

Um levantamento foi realizado entre trabalhos que possuem similaridade com o tema proposto. O objetivo é quantificar elementos teóricos e experimentos que possam estabelecer conexão com conceitos abordados neste capítulo.

2.8.1 Mineração de Dados Textuais

Em geral, o primeiro passo para a classificação de um exemplo é o tratamento do conjunto de dados. Em uma visão mais detalhada, para que um sistema de AM consiga realizar a classificação em um sistema de internet por exemplo, é necessário tratar páginas como exemplos que serão submetidos por um classificador. Páginas normalmente contêm figuras, vídeos, músicas, tags, links e texto. Porém, para problemas descritos neste trabalho, serão considerados apenas atributos textuais, pela grande quantidade de dados disponíveis. Logo é imprescindível o tratamento destes atributos para um formato compreensível aos algoritmos de AM.

A Mineração de Textos é um conjunto de técnicas para gestão do conhecimento, capaz de reconhecer padrões e definir regularidades em documentos escritos em linguagem natural. É um método constituído por etapas que descrevem ganho de conhecimento: (i) Identificação do problema; (ii) Pré-processamento; (iii) Extração Padrões; (iv) Pós-processamento; e (v) Utilização do Conhecimento, que são etapas fundamentadas por Fayyad et al. (1996) .

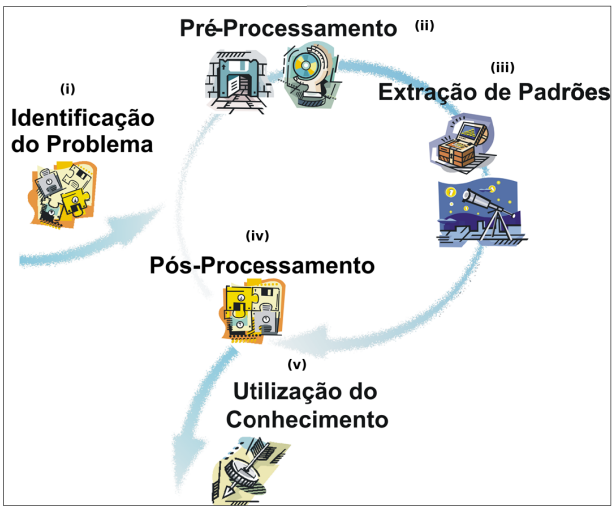

Figura 2.7: Etapas do processo de Mineração de Textos. Adaptado de: (Rezende et al., 2003)

A primeira etapa (i) identificação do problema, consiste em analisar o problema e definir uma estratégia para resolve-lo, neste caso se estabelece o tipo de aprendizado e qual classificador ideal para o problema. No processo de (ii) pré-processamento uma das preocupações consiste em ajustar e tratar as

propriedades textuais, para garantir a aplicação de métodos de Aprendizado de Máquina. Um desses ajustes é a remoção de caracteres especias, números, exclusão de exemplos duplicados e palavras comuns que não interferem no processo de classificação de texto. A extração de padrões (iii) visa obter a extração de conhecimento válido com base no método de aprendizagem. O pós-processamento (iv) consiste em organizar as informações obtidas na extração de padrões para que o classificador seja apto a realizar futuras predições com base na (v) utilização do conhecimento gerado.

Dada a importância dessas etapas, os classificadores abordados neste trabalho são construídos para receber os dados já tratados pelo pré-processamento, com objetivo de realizar a classificação a partir de atributos relevantes ao problema.

2.8.2 Classificação em Tempo Real

O trabalho de Miltsakaki and Troutt (2008) descreve o uso da classificação em tempo real para obter melhores resultados em busca de conteúdos literários. Sua intenção é relacionar uma série de conteúdos disponíveis na web com base nas buscas digitadas pelos usuários. A classificação é dada levando-se em conta três parâmetros: (i) palavra chave da pesquisa, (ii) classificação temática, (iii) análise da dificuldade de leitura. Para isso, o método descrito aqui conta com um conjunto de treinamento rotulado manualmente para predizer qual classificação temática do assunto, bem como a dificuldade de leitura.

O objetivo final é que adolescentes e adultos que possuem baixo ou alto nível de leitura, consigam encontrar um material de leitura adequado para a sua idade. Em seu experimento foi utilizado os algoritmos NB, MIRA *(Margin Infused Relaxed Algorithm)* e MAXENT *(Maximum Entropy)*, seu ponto de partida para realização dos itens (i, ii, iii) foi a utilização de fórmulas para medir a legibilidade e dificuldade em textos: LIX, RIX (Anderson, 1983) e Coleman-Liau (Coleman and Liau, 1975). Seus resultados foram relacionados quanto ao número total de sentenças, o número total de palavras, o número total de palavras longas (sete ou mais caracteres) e número total de letras no texto.

Um aplicação similar a esta é passível de ser utilizada em um sistema de filtragem de páginas web, uma vez que a classificação é dada em tempo real e possui baixo tempo de atualização, dado o ambiente de execução do problema.

O diferencial nos algoritmos NB, MIRA *(Margin Infused Relaxed Algorithm)*, MAXENT *(Maximum Entropy)* utilizados no trabalho de Miltsakaki and Troutt (2008), para que tenham uma resposta em tempo real é o uso das fórmulas para medição de legibilidade e dificuldade, na qual faz uso de seleção de atributos para otimizar o conjunto de dados e o processo de classificação de novos

exemplos.

2.8.3 Smart Saint

O estudo de Rigo (2013), tem por objetivo fazer a moderação de sites com uso do aprendizado semissupervisionado ativo. Sua proposta foi traçar alguns métodos comuns na literatura, juntamente com técnicas atuais para classificação de documentos. Foi desenvolvido um sistema chamado "Smart Saint", com uso de AM para facilitar a moderação de sites da internet, tornando a moderação menos custosa e mais eficiente.

Sua base de dados contou com nove conjunto de dados, todos relacionados a categorias importantes para moderação de sites. Os algoritmos utilizados em seu trabalho foram:

- *k-Nearest Neighbor* (KNN);
- *Support Vector Machine* (SVM);
- *Stochastic Gradient Descent* (SGD), (Bottou, 2010);
- *Logistic Regression* (LogReg), (Hosmer Jr and Lemeshow, 2004);
- *Naive Bayes* (NB).

O funcionamento do sistema "Smart Saint" se comporta como um proxy transparente, na qual o usuário solicita uma url, que é encaminhada ao sistema para verificação de acesso (*bloqueado ou liberado*), logo após, caso o acesso seja permitido a requisição é liberada ao usuário.

A Figura 2.8 detalha o funcionamento interno do sistema "Smart Saint", que pode ser compreendida da seguinte forma: Assim que o sistema proxy recebe uma requisição, ele encaminha o conteúdo HTML para averiguação no sistema "Smart Saint", que por sua vez faz o pré-processamento do conteúdo. (1) No estágio de Pré-Processamento, todo conteúdo HTML é convertido para um formato compreensível ao sistema classificador, chamado de *bag-of-words*. (2) O Classificador faz a análise do conteúdo, classificando-o como negativo ou positivo e, em seguida, envia o resultado ao Avaliador. Já os itens 4 e 5, referem-se respectivamente a avaliação do classificador, que mede o desempenho baseado em instâncias de classificação anteriores. Já o item 5, solicita ao agente externo um rótulo para uma requisição, que se encontra em um ponto de contenção no sistema "Smart Saint".

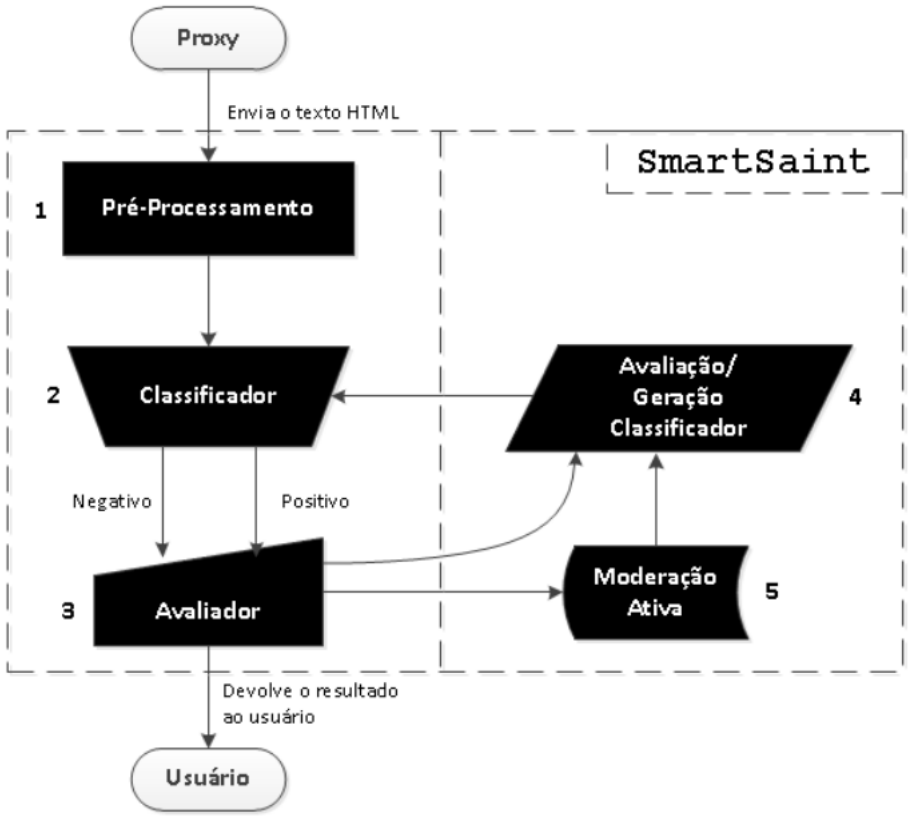

Figura 2.8: Visão Detalhada do Funcionamento do SmartSaint de: (Rigo, 2013)

2.9 Considerações Finais

Dado a descrição da base teórica e alguns trabalhos relacionados ao tema do projeto, propõe-se no capítulo seguinte um estudo para melhor relacionar todas a ideias descritas neste capítulo. O objetivo inicial é abordar os algoritmos LexRank e FlexRank, abordando suas construções, algoritmos, funções, otimizações e qual seus impactos em ambientes reais como no caso de um sistema de internet. Realizada todas estas etapas, o foco deste trabalho é retomar o estudo de (Rigo, 2013), porém com abordagem em um estudo teórico sobre os algoritmos de classificação, com intuito de aprimorar seus resultados com foco no "estado da arte" e por fim tratar tópicos ainda não cobertos em seu estudo.

CAPÍTULO

3

FlexRank: Fast Lexicograph Ranker

Neste capítulo é realizada uma introdução sobre ranking e seus conceitos. Em seguida, é detalhado o algoritmo base deste trabalho LEXRANK (Matsubara, 2008). Por fim, o algoritmo FLEXRANK é descrito, bem como suas adaptações para execução dos experimentos deste trabalho.

3.1 Ranking

Considere um grupo de atletas buscando vagas classificatórias para participar dos jogos Olímpicos. Durante este período, os melhores atletas são selecionados e recebem a classe *classificado* e os atletas que não atingiram pontuação suficiente para classificação recebem a classe *não-classificado*. Para a atribuição destas classes, os atletas são ordenados (ranqueados) segundo algum critério e muitas vezes um limiar neste ranking define a classe, por exemplo os melhores 100 atletas deste *ranking* são considerados classificados. No exemplo apresentado o *ranking* possui mais informação que a classificação e por isso a partir de *rankings* é possível obter uma classificação, mas o inverso não é possível. O *Ranking* ordena exemplos e a Classificação atribui valores nominais aos exemplos.

Para uma série de problemas esta relação de ordem é mais interessante que a classe. Recuperação de informações, sistemas de recomendação e análise de sentimentos são alguns exemplos. Suponha realizar uma busca na internet por uma determinado assunto. A busca deve classificar vários sites e retornar aqueles que melhor relacionam o assunto. Mais do que classificar, é interessante ordenar os sites por relevância, estabelecendo uma relação de ordem entre os sites selecionados (Matsubara, 2008).

Segundo Liu (2009), o aprendizado de *ranking* pode ser dividido em três tipos de abordagens: *pointwise*, *listwise* e *pairwise*. Na Tabela 3.1 são ilustradas as principais diferenças entre as abordagens. No primeiro tipo, *pointwise*, o algoritmo atribui um valor numérico denominado de *score* na qual quanto maior este valor, mais provável é a chance do exemplo ser positivo. No segundo, *listwise*, o algoritmo ordena uma lista de exemplos, seguindo a ordem dos exemplos mais positivos para os menos positivos. O último, *pairwise*, que é o principal foco deste trabalho, ao comparar dois exemplos ele retorna os valores nominais $\{>,=,<\}$. Quando ocorre a comparação dos pares o algoritmo retorna $>$ quando o primeiro exemplo é mais provável, $=$ igualmente provável, e $<$ menos provável ser positivo que o segundo (Matsubara, 2008).

Tabela 3.1: Tabela de comparação abordagens de ranking.

abordagem	entrada	saída	hipótese
pointwise	um exemplo	*score*	$\hat{s}: X \to \mathbb{R}$
listwise	uma lista de exemplos	exemplos ordenados	$\hat{l}: X^{n_{ex}} \to X^{n_{ex}}$
pairwise	um par de exemplos	$\{>,=,<\}$	$\hat{r}: X \times X \to \{>,=,<\}$

A hipótese aprendida de um classificador pode ser definida como $\hat{c}: X \to Y$ onde, como já descrito no Capítulo2, Y representa os possíveis valores do atributo classe. Observe que a hipótese da classificação, por definição é diferente dos problemas de *ranking*, que tem como saída informação que podem expressar uma relação de ordem. A conversão de um *ranking* em um classificador pode ser realizada definindo limiares no *ranking*, como no exemplo apresentado utilizando os 100 melhores atletas para definir entre as classes *classificado* e *não-classificado*.

Os algoritmos estabelecidos em AM podem fornecer um *score* de classificação NAIVE BAYES, REGRESSÃO LOGÍSTICA, REDES NEURAIS, ÁVORES DE DECISÃO, MÁQUINAS DE VETORES DE SUPORTE, VIZINHOS MAIS PRÓXIMOS, INDUÇÃO DE REGRAS (Russell and Norvig, 2003). Logo, são exemplos de algoritmos que permitem obter *score* em uma abordagem *pointwise*.

Entretanto, é possível desenvolver algoritmos *rankeadores* sem a necessidade de um *score*. Na seção a seguir é descrito o algoritmo LEXRANK, um algoritmo desenvolvido por Flach and Matsubara (2007) que possui esta propriedade.

3.2 LEXRANK

O algoritmo LEXRANK foi construído com base em conceitos de árvores de decisão e pelo algoritmo NAIVE BAYES. Sua característica principal está na

utilização da ordenação lexicográfica que pode ser representada por uma estrutura de árvore. Esta seção descreve passo a passo o algoritmo LEXRANK. A descrição do algoritmo foi extraída do trabalho proposto por (Flach and Matsubara, 2007).

Assim como em um dicionário, a primeira etapa realizada pelo algoritmo LEXRANK é a ordenação lexicográfica dos exemplos. Seu conceito assemelha-se a criação de árvores, além de ser um rankeador natural pois ordena seus elementos por definição. O algoritmo implementa a formulação de um *rankeador* que já possui uma ordem relativa em seu processo de implementação, sem a necessidade de obter um *score* pelo sistema classificador.

Exemplo 3.1. *Considere os seguintes exemplos com cinco atributos binários* $\boldsymbol{x}_1$ *= (0, 1, 0, 0, 0),* $\boldsymbol{x}_2$ *= (0, 0, 1, 0, 0),* $\boldsymbol{x}_3$ *= (0, 0, 0, 0, 1) e* $\boldsymbol{x}_4$ *= (0, 0, 1, 1, 1). Ao converter os exemplos em strings e ordená-los lexicograficamente, obtém-se a seguinte ordem:*

$\mathbf{x}_3$ = (0, 0, 0, 0, 1)
$\mathbf{x}_2$ = (0, 0, 1, 0, 0)
$\mathbf{x}_4$ = (0, 0, 1, 1, 1)
$\mathbf{x}_1$ = (0, 1, 0, 0, 0)

Matsubara (2008), em seu trabalho descreve que: *"em uma situação particular, essa ordenação lexicográfica pode representar os exemplos do mais positivo para o mais negativo, como em rankeadores".*

Com base nessa estrutura foi criada a implementação do algoritmo LEXRANK, abordada no trabalho de Flach and Matsubara (2007). O objetivo é ordenar os atributos seguindo uma ordem de preferência e depois realizar a ordenação dos exemplos.

Algoritmo

LEXRANK em sua proposta original atua apenas em conjuntos de exemplos binário e faz uso dos seguinte critérios:

Ordem de preferência de atributos: define uma ordem que impõe um critério que separa exemplos positivos e negativos. Considere ordenar atributos por ganho de informação de tal maneira que os primeiros atributos separam satisfatoriamente os exemplos positivos dos negativos.

Valor de preferência de um atributo: para atributos booleanos, é o valor que representa melhor uma classe positiva, que pode ser definido como zero ou um.

Definição 3.1. ***(Ranking Lexicográfico)*** *Dado um conjunto de atributos booleanos* $(A_1,...,A_{n_{at}})$*, de maneira que os índices apresentam uma ordem de preferência. Seja* v_{j+} *o valor de preferência do atributo* A_j*. Logo, o rankeador*

lexicográfico corresponde a uma ordem de preferência nos atributos e valores dos atributos definido como:

$$\hat{r}_{lex}(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) = \begin{cases} > \textit{se } A_k(\boldsymbol{x}_i) = v_{k+}; \\ < \textit{se } A_k(\boldsymbol{x}_i) \neq v_{k+}. \end{cases}$$

Onde k representa o menor índice de atributo tal que $\mathbf{x}_i$ e $\mathbf{x}_i + 1$ possuam diferentes valores (caso não exista uma índice que satisfaça essas condições, os dois exemplos estão empatados) (Matsubara, 2008).

LexRank é um *ranking* lexicográfico que faz uso do valor de preferência calculado pelas razões de verossimilhança Likelihood Ratio (LR). Este valor deve satisfazer a seguinte condição: $LR(A_j = v_{j+}) \geq 1$, para $LR(A_j) = \frac{p(A_j|+)}{p(A_j|-)}$. Logo, o valor descreve qual atributo é de maior v_{j+} ou de menor v_{j-} preferência. Por padrão, a ordem de preferência é decrescente com uso das razões de chances (Odds Ratio) $OR(A_j) = \frac{LR(A_j = v_{j+})}{LR(A_j = v_{j-})}$ no qual v_{j-} denota o valor de menor preferência.

Com base na definição é possível inferir que o valor de preferência para cada atributo estabelece o seguinte critério:

$$\textit{Valor de Preferência} \left(\frac{p(A_j|+)}{p(A_j|-)}\right) = \begin{cases} v_{j+} & \text{se } LR(A_j = v_{j+}) \geq 1; \\ v_{j-} & \text{se } LR(A_j = v_{j+}) < 1. \end{cases}$$

Intuitivamente o algoritmo LexRank pode ser visto como um *rankeador* do tipo *pairwise*. Dado um conjunto de atributos booleanos $(A_1, ..., A_{n_{at}})$, o algoritmo faz uma ordenação entre os atributos pela razão de verossimilhança Likelihood Ratio e aplicada a ordenação dos exemplos pela comparação aos pares. Esta comparação retorna o sinal $>$ quando o primeiro exemplo é mais provável, $=$ igualmente provável, e $<$ menos provável ser positivo que o segundo (Matsubara, 2008).

Dadas as definições anteriores, no Algoritmo 1 destaca-se, passo a passo, a etapa de treinamento de LexRank. Inicialmente são instanciadas as variáveis (OR) e (ValordePreferencia) e obtido através do laço de (para) a razão de verossimilhança Likelihood Ratio pelo uso dos valores da Matriz de Confusão. Em seguida é feita a comparação entre as razões $LR(A_j = 0)$ e $LR(A_j = 1)$ e definido o Valor de Preferência e o cálculo de Odds Ratio para o atributo j. Por fim, é realizada a ordenação dos atributos pelo valor obtido em (OR).

Para uma melhor compreensão, pode-se descrever as etapas executadas pelo Algoritmo 1 em:

1. Entrada: (x) é composto pelos atributos do conjunto de dados;
2. Saída: $LexRankAtributos$, $ValorDePreferencia$;

   $LexRankAtributos$: Atributos ordenados pelo valor do ODDS RATIO.

   $ValorDePreferencia$: Valor do atributo após cálculo do ODDS RATIO.
3. (Linhas 5 e 6) - Cálculo da Matriz de Confusão para cada atributo;
4. (Linhas 7 e 8) - Cálculo do LIKELIHOOD RATIO $(A_j = 0)$ e $(A_j = 1)$;
5. (Linha 9) - Determinar VALOR DE PREFERÊNCIA ;
6. (Linhas 10 à 14) - Calcular ODDS RATIO e atribuir Valor de Preferência;
7. (Linha 16) - Ordenar os atributos de acordo com os $OR(A_j)$;
8. (Linha 17) - Retornar os valores de $LexRankAtributos$ e $ValorDePreferencia$.

Algoritmo 1: Algoritmo de Treinamento LexRank

Entrada: Conjunto de Atributos x;
Saída: $LexRankAtributos$, $ValorDePreferencia$;

```
1  OR[ ] ;                    /* vetor da razão de chances (Odds Ratio) */
2  ValordePreferencia[ ] ;    /* vetor do valor de preferência */
3  início
       /* verificação de cada atributo                                  */
4      para j ← 1 até x faça
5          VP[j],FP[j],VN[j],FN[j] ← 1 ;        /* correção de laplace */
6          atualizar VP[j],FP[j],VN[j],FN[j] considerando o atributo j;
7          LR(A_j = 0) = VP[j]/FP[j];                  /* Likelihood (0) */
8          LR(A_j = 1) = FN[j]/VN[j];                  /* Likelihood (1) */
9          se LR(A_j = 0) >= LR(A_j = 1) então
10             ValordePreferencia[j] = 0;     /* Atributo Preferência (0) */
11             OR[j] = LR(A_j=0)/LR(A_j=1);          /* Oddis Ratio LR(0)/LR(1) */
12         senão
13             ValordePreferencia[j] = 1;     /* Atributo Preferência (1) */
14             OR[j] = LR(A_j=1)/LR(A_j=0);          /* Oddis Ratio LR(1)/LR(0) */
15     fim
16     LexRankAtributos ← índice de x ordenado de acordo com OR(A_j);
17     retorna LexRankAtributos,ValordePreferencia;
18 fim
```

O exemplo a seguir ilustra os passos da execução de LEXRANK.

Exemplo 3.2. *Considere o conjunto de exemplos descritos na Tabela 3.2*

Tabela 3.2: Conjunto de exemplos de treinamento.

A_α	A_β	A_Θ	Classe
0	1	0	+
0	1	1	-
1	1	0	+
0	0	1	-
1	1	1	-
0	0	1	+
0	1	0	+
1	0	0	-
0	0	0	+
1	1	1	-

1. A partir do conjunto de exemplos são calculadas as razões de verosimilhança $LR(A_j)$ com suavizador de *Laplace*, ou seja, consiste em somar 1 no final da contagem. Isso evita eventos com probabilidade zero, obtém-se:

$$LR(A_\alpha = 0) = \frac{5}{3}$$
$$LR(A_\alpha = 1) = \frac{2}{4}$$
$$LR(A_\beta = 0) = \frac{3}{3}$$
$$LR(A_\beta = 1) = \frac{4}{4}$$
$$LR(A_\Theta = 0) = \frac{5}{2}$$
$$LR(A_\Theta = 1) = \frac{2}{5}$$

2. Para cada um dos atributos com valores $A_j = 0$ ou $A_j = 1$, o valor de preferência para A_α,A_β,A_Θ é $v_{j+} = 0$ e $v_{j-} = 1$.

3. Calculando $OR(A_j)$:

$$OR(A_\alpha) = \frac{5/3}{2/4} = 3,332$$
$$OR(A_\beta) = \frac{3/3}{4/4} = 1,000$$
$$OR(A_\Theta) = \frac{5/2}{2/5} = 6,250$$

Tabela 3.3: Passos de treinamento do Algoritmo LEXRANK.

j	VP	FP	FN	VN	LR(0)	LR(1)	PREF	OR
A_α	5	2	3	4	5/3	2/4	0	3,332
A_β	3	4	3	4	3/3	4/4	0	1,000
A_Θ	5	2	2	5	5/2	2/5	0	6,250

4. Após o cálculo do ODDS RATIO detalhado na Tabela 3.3 com os valores de cada atributo (número do atributo, quantidade de verdadeiros positivos, falsos positivos, falsos negativos, verdadeiros negativos, razão de verossimilhança = 0 e 1, valor de preferência e ODDS RATIO, são ordenados os atributos em ordem decrescente de $OR(A_1, A_2$ e $A_3)$ e os exemplos lexicograficamente, é obtido o *ranking* ilustrado na Figura 3.1 e na Tabela 3.4.

5. Obtidos os valores de ODDS RATIO é possível estabelecer uma relação de ordem entre os atributos do maior para o menor valor de (OR).

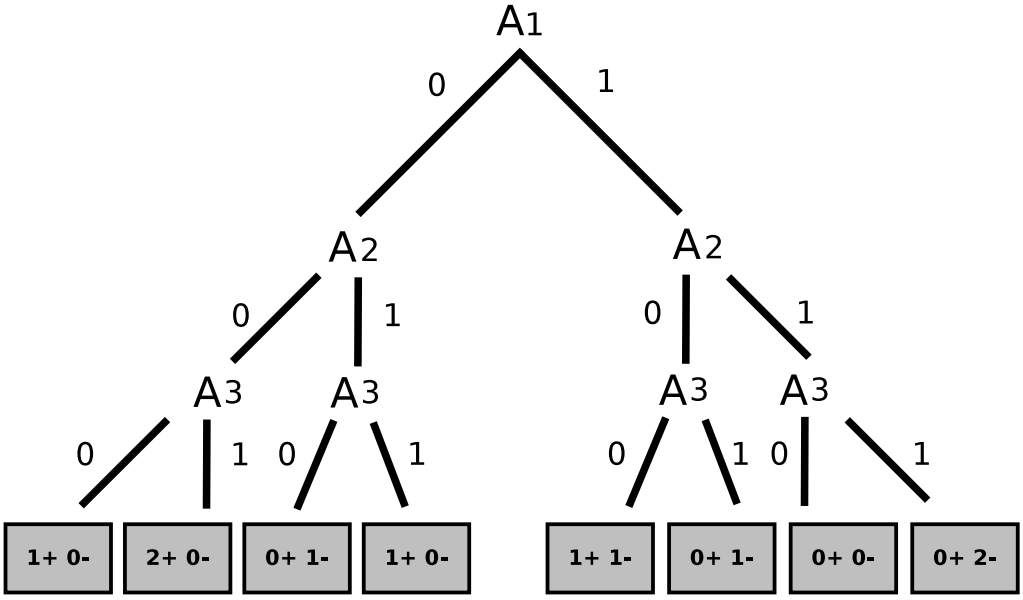

Figura 3.1: Exemplo 3.2 - Árvore de Decisão.

Tabela 3.4: Conjunto de exemplos ordenados lexicograficamente utilizando LexRank.

A_1	A_2	A_3	Classe
0	0	0	+
0	0	1	+
0	0	1	+
0	1	0	-
0	1	1	+
1	0	0	-
1	0	0	+
1	0	1	-
1	1	1	-
1	1	1	-

A complexidade do algoritmo LexRank é de $O(n_{ex})$ pelo cálculo de LR, acrescido da complexidade na inserção de $OR(A_j) = O(log(n_{at}))$ dos atributos ordenados na lista. Logo, a complexidade de LexRank é de $O(n_{at} \times n_{ex} + n_{at} \times log(n_{at}))$.

Fica evidente a simplicidade com que o algoritmo LexRank consegue ordenar os elementos de forma rápida. Em testes realizados entre atributos binários, LexRank obteve resultados significativos em relação aos algoritmos citados no Capítulo 2. Porém seu tempo de execução ficou aquém do esperado para os objetivos deste trabalho, pelo fato da implementação analisar toda coleção de atributos de um conjunto de dados.

Dada a limitação de complexidade imposta pelo LexRank, foi elaborado um novo algoritmo com base na descrição de LexRank. FlexRank é um algoritmo similar que realiza a redução dos atributos e diminui significativamente o tempo de execução do algoritmo na etapa de classificação de novos exemplos. A seção a seguir descreve o algoritmo FlexRank e quais adaptações foram necessárias para compor o novo algoritmo.

3.3 FlexRank

O algoritmo FlexRank foi construído utilizando a mesma estrutura do algoritmo LexRank. No entanto, a diferença em relação a LexRank é caracterizada pela seleção de atributos realizada na etapa de treinamento. Por meio de uma equação simplificada é possível otimizar o cálculo de AUC e selecionar apenas atributos relevantes no conjunto de dados. A seção a seguir detalha como este processo é realizado na descrição do algoritmo FlexRank.

3.3.1 Algoritmo

Considere o Exemplo 3.2 visto na Seção 3.2, realizando os cálculos estatísticos do algoritmo FLEXRANK, são obtidos os valores detalhados na Tabela 3.5. A qual possui os respectivos valores: número do atributo, o número de exemplos verdadeiros positivos, falsos positivos, falsos negativos, verdadeiros negativos, razão de verossimilhança (*LR*) = (0), razão de verossimilhança (*LR*) = (1), valor de preferência e razão de chances (ODDS RATIO). Com o objetivo de melhorar a compreensão do algoritmo, nesta etapa não é utilizado *laplace* na construção da Matriz de Confusão para o Exemplo 3.2.

Tabela 3.5: Passos de treinamento do Algoritmo FLEXRANK.

j	VP	FP	FN	VN	LR(0)	LR(1)	PREF	OR
A_2	4	1	2	3	4/2	1/3	0	6,00
A_3	2	3	2	3	2/2	3/3	0	1,00
A_1	4	1	1	4	4/1	1/4	0	16,00

Ao mapear os atributos pela razão de chances ODDS RATIO do Exemplo 3.2 em ordem decrescente obtém-se os atributos (A_1, A_2 e A_3). A partir da Tabela 3.5 é possível definir um *espaço de cobertura*, por meio de um algoritmo de classificação com base na ordem lexicográfica. A primeira propriedade é incluir atributos que aumentam AUC ROC. Assim atributos que não incrementam o valor de AUC podem ser descartados.

O plano bidimensional utilizando VP (eixo vertical) e FP (eixo horizontal) pode ser definido como um *espaço de cobertura R* (Fürnkranz and Flach, 2005). Logo, sua escala difere da utilizada na curva ROC, que utiliza TVP (Taxa de Verdadeiro Positivo) e TFP (Taxa de Falso Positivo) para resultados em AUC.

Considere o Exemplo 3.2, o atributo A_1 produz o seguinte ponto P_1 no *espaço de cobertura*: (VP = 6, FP = 4). Ao mapear este ponto no *espaço de cobertura* como o primeiro atributo mais relevante pela razão de chances (ODDS RATIO) do conjunto dados, cria-se uma divisão do conjunto em duas regiões R_+ e R_-, ilustrado na Figura 3.2. O comprimento e a largura do retângulo R_+ representam, respectivamente, o número de exemplos positivos e negativos em R_+, e o mesmo ocorre com R_-. Por fim, as duas regiões restantes, $R_{inferno}$ *e* R_{ceu}, são consideradas regiões "fixas", as quais não receberão pontos de divisão e elas permanecem intocadas pelos algoritmos LEXRANK e FLEXRANK.

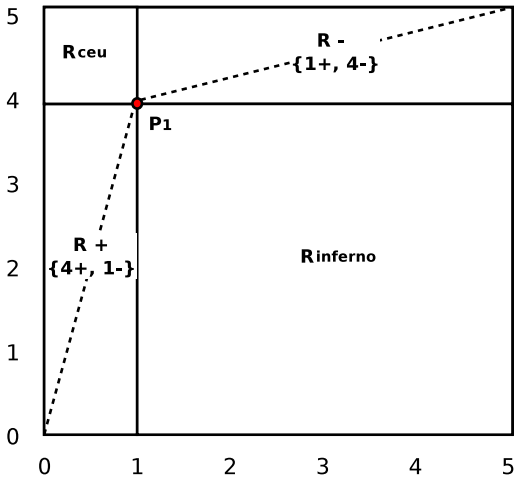

Figura 3.2: Exemplo 3.1 - Atributo A_1 *no espaço de cobertura*.

Ao incorporar o atributo A_2 são criados dois pontos (P_2' e P_2'') dentro dos espaços R_+ e R_-. A ordem e número de exemplos positivos e negativos representados pelas folhas da Figura 3.1 no *espaço de cobertura* são detalhados na Figura 3.3.

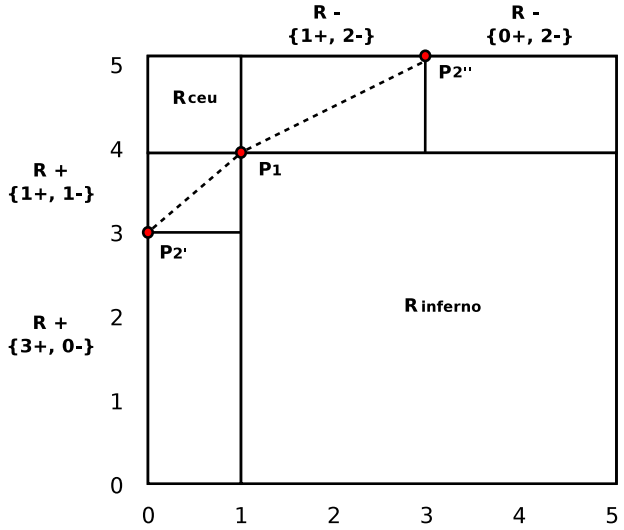

Figura 3.3: Exemplo 3.1 - Atributo A_1 e A_2 *no espaço de cobertura*.

Pela Figura 3.3 é possível observar que ao inserir mais um atributo no *espaço de cobertura*, a área abaixo da curva aumentou. Isso ocorre por que o segundo atributo torna-se relevante ao combinar os resultados entre o atributo A_1. No entanto, o atributo A_3 de ODDS RATIO não aumenta a área abaixo da curva no *espaço de cobertura*. Para esses casos o atributo deve ser desconsiderado, pois não é relevante o suficiente para compor o conjunto de atributos na classificação de novos exemplos.

A diferença no funcionamento do algoritmo FLEXRANK está associada à esta seleção de atributos. A estratégia adotada por FLEXRANK é obter o maior ganho de AUC sobre o *espaço de cobertura*, ou seja, quanto mais representativo

o atributo no *espaço de cobertura* melhor será o valor de AUC. Para comprovar esta afirmação, suponha analisar o Gráfico ROC representado na Figura 3.4.

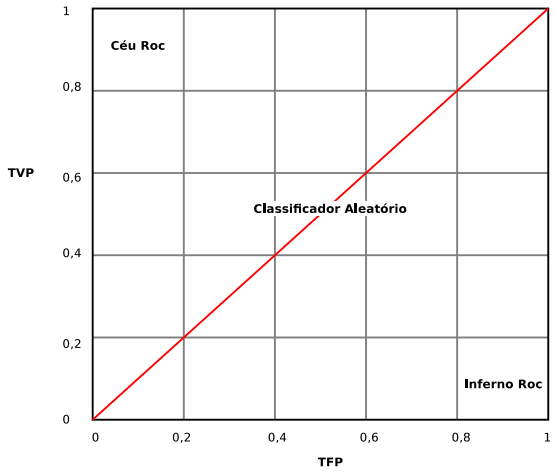

Figura 3.4: Gráfico ROC.

Pela definição detalhada na Seção 2.7.7, quanto maior a TVP (Taxa de Verdadeiros Positivos) e menor a TFP (Taxa de Falsos Positivos), maior será a curva ROC. Logo, ao traçar uma linha diagonal no plano bidimensional partindo do ponto (0,0) até (1,1) , todos os pontos (Pos, Neg) representados acima da diagonal tendem a estar mais próximos do Céu ROC representado pelo ponto (0,1), que descreve uma classificação perfeita onde todos exemplos são preditos corretamente.

A Definição 3.2 descreve uma equação que realiza o cálculo de ganho para cada atributo, partindo do pressuposto de que pontos acima da diagonal do plano bidimensional no Gráfico ROC são relevantes na seleção de atributos para o algoritmo FLEXRANK.

Definição 3.2. *Dado um espaço de cobertura R representado pelos eixos VP e FP com uma linha diagonal partindo do ponto (0,0) até (Pos, Neg), é definida a área acima da linha diagonal do espaço de cobertura R por:*

$$AUCINC = \frac{VP.Neg - Pos.FP}{2} \tag{3.1}$$

Na qual, *AUCINC* representa o valor de AUC ganho pelo atributo acima da linha diagonal sob a *curva de cobertura*.

Para comprovar tal afirmação, considere que dado um ponto (VP, FP) acima da linha diagonal, a área sob a *curva de cobertura* é:

$$\begin{gathered} Area(R_{Inferno}) + Area(R_+)/2 + Area(R_-)/2 \\ = \\ VP(Neg - FP) + (VP.FP)/2 + (Pos - VP)(Neg - FP)/2 \end{gathered} \tag{3.2}$$

E a área abaixo da linha diagonal é:

$$Area(R)/2 = (Pos.Neg)/2 \tag{3.3}$$

Subtraindo a área abaixo da linha diagonal da área de *curva de cobertura* obtém-se:

$$\frac{VP.Neg - Pos.FP}{2} \tag{3.4}$$

Pela afirmação da Definição 3.2 é possível analisar quais pontos estão acima da linha diagonal e portanto quais são os atributos relevantes sobre a curva ROC. Logo, um atributo ao ser submetido a Equação 3.4 e obter valor maior que 0, torna-se um atributo relevante. Caso contrário, o atributo não deve ser considerado válido para classificação de novos exemplos. A vantagem em se utilizar a Definição 3.2 é o simples fato de que o cálculo para seleção de atributos é simplificado, entretanto é possível também obter esses valores apenas pelo resultado de AUC.

3.3.2 LexRank x FlexRank

Para comprovar a diferença na execução dos algoritmos LexRank e FlexRank, considere analisar o exemplo da Tabela 3.6:

Tabela 3.6: Exemplo com 5 atributos binários.

A_α	A_β	A_Θ	A_Ω	A_Δ	Classe
0	1	0	0	0	+
0	1	1	1	1	-
1	1	0	0	0	+
0	0	1	1	0	-
1	1	1	1	0	-
0	0	1	0	1	+
0	1	0	0	1	+
1	0	0	0	1	-
0	0	0	1	0	+
1	1	1	0	0	-
0	1	0	0	1	+
1	0	1	0	0	-

Inicialmente o primeiro passo para ambos os algoritmos é realizar o cálculo de Likelihood Ratio e Odds Ratio. A Tabela 3.7, detalha os valores obtidos por cada atributo. Para cada um dos atributos com valores $A_j = 0$ ou $A_j = 1$, o valor de preferência para $A_\alpha, A_\beta, A_\Theta, A_\Omega, A_\Delta$ é $v_{j+} = 0$ e $v_{j-} = 1$.

Tabela 3.7: Passos de treinamento para os Algoritmos LEXRANK e FLEXRANK.

j	VP	FP	FN	VN	LR(0)	LR(1)	PREF	OR
A_α	5	1	2	4	5/2	1/4	0	10,000
A_β	2	4	3	3	2/3	4/3	0	0,499
A_Θ	5	1	1	5	5/1	1/5	0	25,000
A_Ω	5	1	3	3	5/3	1/3	0	5,003
A_Δ	3	3	4	2	3/4	3/2	0	0,500

Após obter os dados de treinamento o algoritmo LEXRANK ordenada os atributos mediante os resultados de ODDS RATIO em ordem decrescente de $OR(A_1,A_2,A_3,A_4,A_5)$ e posteriormente realiza a ordenação lexicográfica dos exemplos conforme detalha as Tabelas 3.8 e 3.9.

Tabela 3.8: Ordenação dos Atributos.

A_Θ	A_α	A_Ω	A_Δ	A_β	Classe
0	0	0	0	1	+
1	0	1	1	1	-
0	1	0	0	1	+
1	0	1	0	0	-
1	1	1	0	1	-
1	0	0	1	0	+
0	0	0	1	1	+
0	1	0	1	0	-
0	0	1	0	0	+
1	1	0	0	1	-
0	0	0	1	1	+
1	1	0	0	0	-

Tabela 3.9: Ordenação dos Exemplos.

A_1	A_2	A_3	A_4	A_5	Classe
0	0	0	0	1	+
0	0	0	1	1	+
0	0	0	1	1	+
0	0	1	0	0	+
0	1	0	0	1	+
0	1	0	1	0	-
1	0	0	1	0	+
1	0	1	0	0	-
1	0	1	1	1	-
1	1	0	0	0	-
1	1	0	0	1	-
1	1	1	0	1	-

A Figura 3.6 mostra a árvore de decisão criada pelo algoritmo LEXRANK depois de finalizado seu processo de ordenação. Com apenas 5 atributos a construção da árvore de decisão para LEXRANK torna-se relativamente grande.

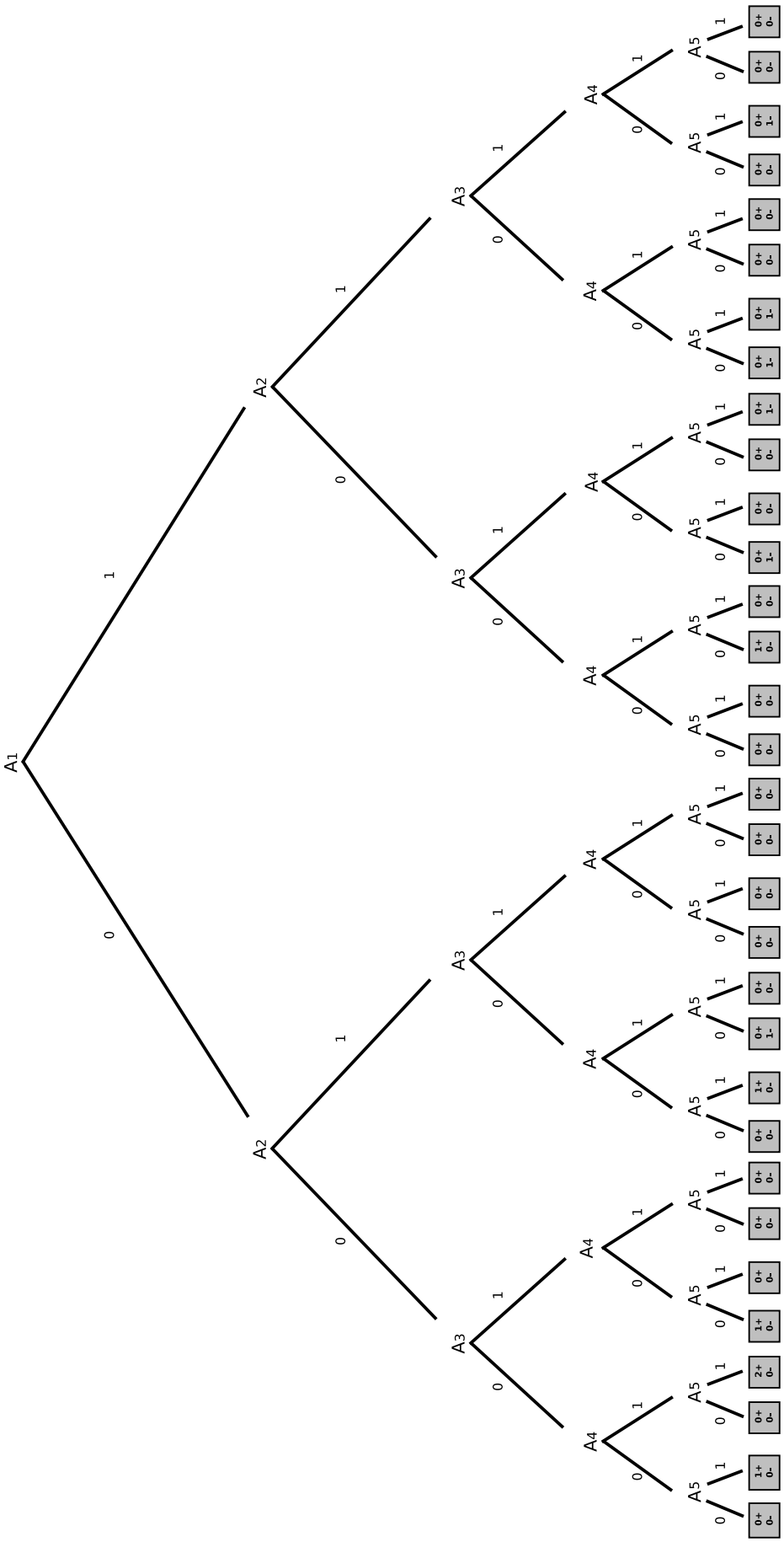

Figura 3.5: LexRank - Árvore de Decisão para exemplo da Tabela 3.6.

No entanto, o processo de execução do algoritmo FLEXRANK realiza a seleção de atributos pela Equação 3.4 após os cálculos obtidos pela Tabela 3.7. Neste caso, FLEXRANK aplica a equação atributo por atributo levando-se em conta a ordem estabelecida por ODDS RATIO. As Tabelas 3.10, 3.11 e 3.12 detalham o cálculo de ganho para cada atributo por meio da Equação 3.4 e a última coluna descreve o ganho normalizado, para que o valor obtido tenha variação entre de ($-0,5 \longleftrightarrow 0,5$), para isso é realizada a divisão do resultado pela quantidade de exemplos *Pos* vezes a quantidade de exemplos *Neg*. A partir do segundo atributo conforme detalhado na Figura 3.3 são criadas partições e para cada partição deve ser calculada a Equação 3.4.

Tabela 3.10: Cálculo de Ganho FLEXRANK- Atributo A_1.

j	VP	FP	*Pos*	*Neg*	Equação 3.4	Resultado	Ganho
$A_1 = A_\Theta$	5	1	6	6	$\frac{(5*6)-(6*1)}{2}$	12	0,333

Tabela 3.11: Cálculo de Ganho FLEXRANK- Atributo A_2.

j	VP	FP	*Pos*	*Neg*	Equação 3.4	Resultado	Ganho
$A_2 = A_\alpha$	4	1	4	2	$\frac{(4*2)-(4*1)}{2}$	2	0,100
$A_2 = A_\alpha$	1	0	3	3	$\frac{(1*3)-(3*0)}{2}$	1,5	

Tabela 3.12: Cálculo de Ganho FLEXRANK- Atributo A_3.

j	VP	FP	*Pos*	*Neg*	Equação 3.4	Resultado	Ganho
$A_3 = A_\Omega$	3	1	3	1	$\frac{(3*1)-(3*1)}{2}$	0	0,031
$A_3 = A_\Omega$	1	0	2	0	$\frac{(1*0)-(2*0)}{2}$	0	
$A_3 = A_\Omega$	1	0	1	2	$\frac{(1*2)-(1*0)}{2}$	1	
$A_3 = A_\Omega$	0	0	2	1	$\frac{(0*1)-(2*0)}{2}$	0	

Pelos dados das Tabelas 3.10, 3.11 e 3.12 apenas os atributos A_1,A_2 e A_3 são relevantes no processo de seleção, pois são atributos que possuem ganhos acima de $0,0$, ou seja, atributos que estão acima da linha diagonal conforme a Definição 3.2. Os ganhos para os atributos A_4 e A_5 não foram detalhados pois possuem valores negativos pela Equação 3.4. Realizado os cálculos é feita a ordenação dos atributos e exemplos conforme detalha as Tabelas 3.13 e 3.14.

Tabela 3.13: Ordenação dos Atributos.

A_Θ	A_α	A_Ω	Classe
0	0	0	+
1	0	1	-
0	1	0	+
1	0	1	-
1	1	1	-
1	0	0	+
0	0	0	+
0	1	0	-
0	0	1	+
1	1	0	-
0	0	0	+
1	1	0	-

Tabela 3.14: Ordenação dos Exemplos.

A_1	A_2	A_3	Classe
0	0	0	+
0	0	0	+
0	0	0	+
0	0	1	+
0	1	0	+
0	1	0	-
1	0	0	+
1	0	1	-
1	0	1	-
1	1	0	-
1	1	0	-
1	1	1	-

A Figura 3.6 mostra a árvore de decisão criada pelo algoritmo FLEXRANK depois de finalizado seu processo de ordenação. Quando comparado ao LEXRANK sua árvore é ligeiramente menor e o custo de tempo é reduzido tanto na etapa de ordenação dos exemplos como na geração da árvore de decisão.

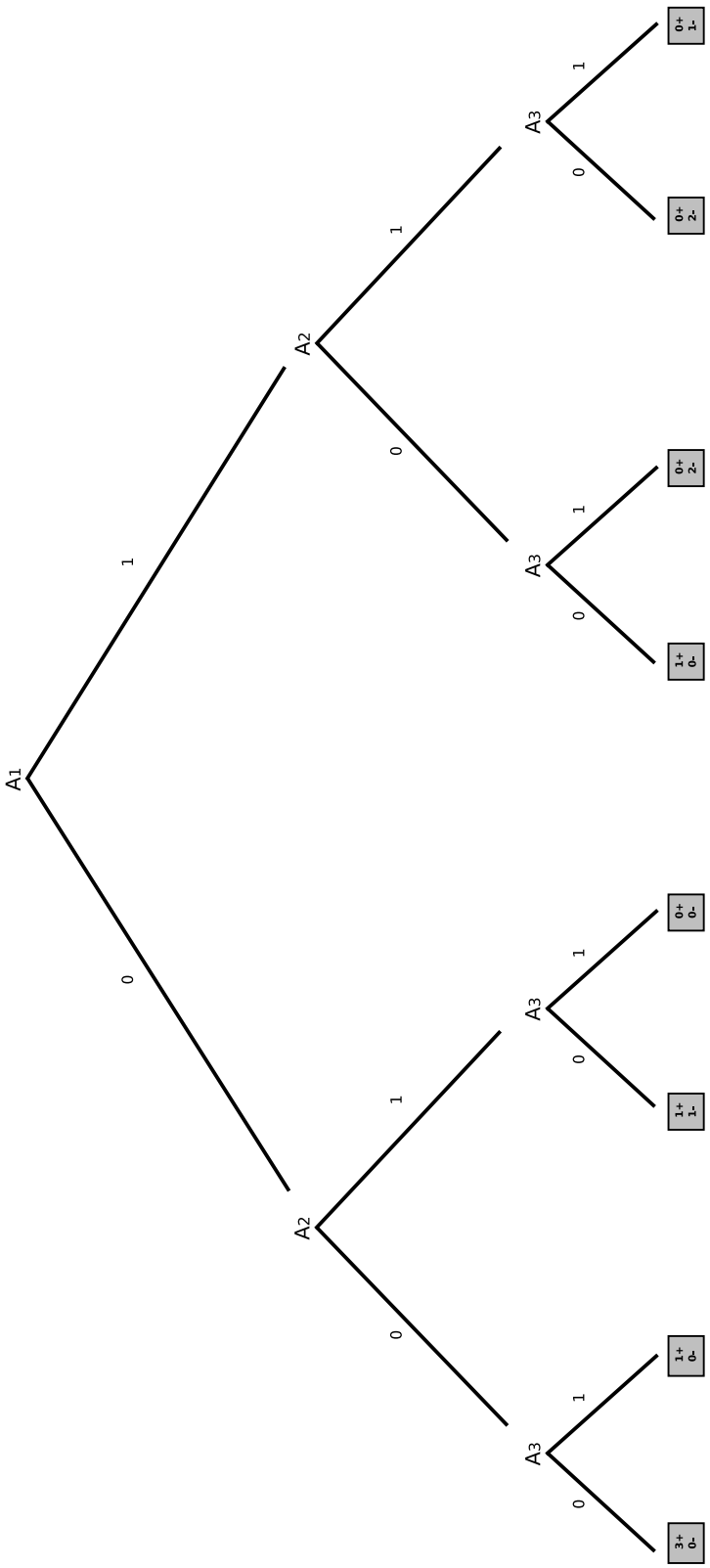

Figura 3.6: FLEXRANK - Árvore de Decisão para exemplo da Tabela 3.6.

3.3.3 Algoritmo

O Algoritmo 2 descreve a etapa de treinamento de FLEXRANK com uso da Definição 3.2 para realização da seleção de atributos, o que torna significativamente rápido o processo de classificação. Como FLEXRANK obtém os melhores atributos na etapa de seleção, espera-se que os valores de AUC obtidos por FLEXRANK em relação a LEXRANK nos experimentos da Seção 4 sejam melhores ou iguais a LEXRANK.

1. Entrada: (x) é composto pelos atributos do conjunto de dados;

2. Saída: $FlexRankAtributos$, $ValorDePreferencia$;

   $FlexRankAtributos$: Atributos ordenados pelo valor do ODDS RATIO e ganho AUC.

   $ValorDePreferencia$: Valor do atributo após cálculo do ODDS RATIO.

3. (Linhas 5) - Cálculo do LIKELIHOOD RATIO $(A_j = 0)$ e $(A_j = 1)$;

4. (Linhas 6) - Determinar VALOR DE PREFERÊNCIA;

5. (Linhas 7) - Calcular ODDS RATIO, caso $Pref = 0$;

6. (Linhas 8) - Atribuir VALOR DE PREFERÊNCIA = 0;

7. (Linhas 10) - Calcular ODDS RATIO, caso $Pref = 1$;

8. (Linhas 11) - Atribuir VALOR DE PREFERÊNCIA = 1;

9. (Linhas 12) - Armazenar todos os VALORES DE PREFERÊNCIA em vetor;

10. (Linhas 14) - Ordenar os atributos de acordo com o $OR(A_j)$;

11. (Linhas 17 e 18) - Verifica se o atributo incrementa AUC e adiciona ao vetor de atributos relevantes;

12. (Linha 20) - Retornar os valores de $FlexRankAtributos$ e $ValorDePreferencia$.

Algoritmo 2: Algoritmo de Treinamento FlexRank

Entrada: Conjunto de Atributos x;
Saída: $FlexRankAtributos$, $ValorDePreferencia$;

```
1  OR[ ] ;              /* vetor da razão de chances (Odds Ratio) */
2  ValordePreferencia[ ] ;          /* vetor do valor de preferência */
3  início
4      para j = 0 até x faça
5          Calcular LR(Aj);
6          Define Valor de preferência para cada atributo;
7          OR[j] = LR(Aj=0) / LR(Aj=1) ;
8          Pref = 0;
9          se LR(Aj = 1) > LR(Aj = 0) então
10             OR[j] = 1/OR[j];
11             Pref = 1;
12         ValorDePreferencia[j] = Pref;
13     fim
14     AttRank = índices de x em ordem decrescente de OR;
15     FlexRankAtributos = lista vazia;
16     para j = 0 até x faça
17         se AttRank[j] incrementa AUC então
18             FlexRankAtributos.adiciona(AttRank[j]);
19     fim
20     retorna (FlexRankAtributos, ValorDePreferencia);
21 fim
```

Realizada a descrição do algoritmo FLEXRANK, é necessário ainda utilizar estruturas eficientes na implementação do FLEXRANK. A Seção a seguir detalha quais estratégias foram abordadas para otimizar o algoritmo FLEXRANK em comparação aos algoritmos citados no Capítulo 2.

3.3.4 Otimizações da implementação do algoritmo FLEXRANK

Uma das etapas mais onerosas ao realizar o processo de treinamento em LEXRANK e FLEXRANK é a contagem dos valores correspondentes à Matriz de Confusão (VP, FP, FN, VN). Esse processo compreende analisar todos atributos e seus respectivos valores (0 ou 1) em cada exemplo, logo em seguida mapear os valores para criação da Matriz de Confusão.

Com objetivo de reduzir o tempo de cálculo para montagem da Matriz de Confusão, o conjunto de dados foi armazenado em matriz e posteriormente cada coluna representando um atributo teve seu complemento adicionado a matriz. A Tabela 3.15 detalha o complemento dos atributos ($\bar{A_1}$,$\bar{A_2}$,$\bar{A_3}$) e classe

($Cl\bar{a}sse$) do Exemplo 3.2. Foi um procedimento necessário para que fosse possível realizar o cálculo da Matriz de Confusão através do produto escalar entre matrizes, assim ao invés de calcular cada atributo separadamente o produto escalar retorna uma matriz resultante com os valores para montagem da Matriz de Confusão o que diminui significativamente o tempo de execução na etapa de treinamento.

Tabela 3.15: Conjunto de exemplos de treinamento acrescidos do seu complemento.

$\bar{A_2}$	A_2	$\bar{A_3}$	A_3	$\bar{A_1}$	A_1	$Cl\bar{a}sse$	$Classe$
1	0	0	1	1	0	-	+
1	0	0	1	0	1	+	-
0	1	0	1	1	0	-	+
1	0	1	0	0	1	+	-
0	1	0	1	0	1	+	-
1	0	1	0	0	1	-	+
1	0	0	1	1	0	-	+
0	1	1	0	1	0	+	-
1	0	1	0	1	0	-	+
0	1	0	1	1	1	+	-

Para uma melhor compreensão, suponha realizar o cálculo da Matriz de Confusão para o atributo A_2 do Exemplo 3.2. Ao utilizar o produto escalar, faz-se a multiplicação do atributo pelo valor da classe transposta correspondente, ambos acrecidos de seus complementos. Abaixo é ilustrado a multiplicação entre as duas matrizes:

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \text{x} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix}$$

Por fim o resultado obtido na matriz resultante é representado pelos valores da Matriz de Confusão pela seguinte condição:

$$\begin{bmatrix} 4 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} VP & FN \\ FP & VN \end{bmatrix}$$

Construída a Matriz de Confusão calcula-se ODDS RATIO. Com os valores já armazenados em uma matriz resultante aplica-se a divisão entre as linhas (0 e 1) da matriz $\frac{VP}{FP}$ e $\frac{FN}{VN}$ que representam respectivamente os valores de $LR(A_j = 0)$ e $LR(A_j = 1)$. Por fim, $LR(A_j = 0)$ e $LR(A_j = 1)$ são submetidos a divisão do maior pelo menor e obtido os valores de ODDS RATIO e de preferência do referido atributo.

Realizada a etapa de cálculos aplica-se a ordenação dos atributos na matriz por ODDS RATIO. Em seguida, realiza-se a seleção de atributos que utiliza a matriz de atributos e de Confusão para aplicar a Equação 3.1. A etapa de seleção de atributos é simplificada dado que os valores (VP, FP, FN, VN) e total de (Pos, Neg) já são contabilizados pelas operações entre matrizes na etapa inicial de LEXRANK e FLEXRANK.

Em linguagem de programação muitas das operações citadas nesta seção são realizadas por bibliotecas já incluídas também nos algoritmos citados no Capítulo 2. Em PYTHON (Bird et al., 2009) a operação entre matrizes é feita pela biblioteca NUMPY (Van Der Walt et al., 2011) e o produto escalar pela função SAFE_SPARSE_DOT [1]. Para o cálculo de ODDS RATIO é aplicado o uso de LIST COMPREHENSION [2] e sua ordenação pela função NP.ARGSORT [3], também utilizada na ordenação dos exemplos.

Com base nas definições acima o Algoritmo 3 descreve as otimizações na etapa de treinamento de FLEXRANK. Foram estratégias construídas para que os algoritmos LEXRANK e FLEXRANK fossem competitivos nos experimentos da Seção 4.2. Haja vista que todos os algoritmos citados no Capítulo 2 são referenciados pela biblioteca SCIKIT-LEARN (Pedregosa et al., 2011) com suas devidas otimizações.

Para uma melhor compreensão, pode-se descrever as etapas executadas pelo Algoritmo 3 em:

1. Entrada: (x) é composto pelo do conjunto de treino e limiar de ganho;

2. Saída: $FlexRankAtributos$, $ValorDePreferencia$;

   $FlexRankAtributos$: Atributos ordenados pelo valor do ODDS RATIO e ganho AUC.

[1] http://scikit-learn.org/stable/developers/utilities.html
[2] https://docs.python.org/2/tutorial/datastructures.html
[3] http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.argsort.html

ValorDePreferencia: Valor de preferência do atributo após cálculo do ODDS RATIO.

3. (Linhas 3) - *attr* recebe um atributo com todos os valores;

4. (Linhas 4) - Criação da Matriz de Confusão para cada atributo;

5. (Linhas 5 e 6) - Cálculo do LIKELIHOOD RATIO $(A_j = 0)$ e $(A_j = 1)$;

6. (Linha 7 à 11) - Calcular ODDS RATIO e atribuir Valor de Preferência;

7. (Linhas 12) - Caso o valor de preferência do atributo seja (1) inverter valores do atributo;

8. (Linha 14) - Ordenar os atributos de acordo com os $OR(A_j)$;

9. (Linha 15) - *Particao* recebe as instâncias do conjunto de treino;

10. (Linha 17) - Verifica se o total de *Pos* ou *Neg* é zero, em caso afirmativo a Equação 3.4 possui resultado (0);

11. (Linha 22 à 23) - Criação da Matriz de Confusão para cada atributo com base nas instâncias adicionas a *Particao*;

12. (Linha 26) - Cálculo de ganho para cada atributo;

13. (Linha 27) - Adiciona as instâncias que serão utilizadas pelo próximo atributo;

14. (Linha 29) - Verifica se ganho é maior que *limiar*;

15. (Linha 30) - *Particao* é atualiza com as instâncias da *lista_profundidade*;

16. (Linha 31) - Armazena atributos relevantes;

17. (Linha 33) - *FlexRankAtributos* recebe os atributos mais relevantes;

18. (Linha 34) - Retornar os valores de *FlexRankAtributos* e *ValorDePreferencia*.

Algoritmo 3: Algoritmo de Treinamento Otimizado FLEXRANK

Entrada: Conjunto de treino x, *limiar*

Saída: $FlexrankAtributos$ e $ValorDePreferencia$;

início
  para $j \leftarrow 1$ **até** *numero de atributos* **faça**
    $attr \leftarrow$ coluna j de x; /* vetor binário do atributo j */
    $\begin{bmatrix} VP & VN \\ FP & VN \end{bmatrix} = \begin{bmatrix} \overline{attr}^t \\ attr^t \end{bmatrix} \times \begin{bmatrix} \bar{y} & y \end{bmatrix}$ /* Criação Matriz de Confusão */
    $LR(A_j = 0) = \frac{VP}{FP}$; /* LIKELIHOOD RATIO (0) */
    $LR(A_j = 1) = \frac{FN}{VN}$; /* LIKELIHOOD RATIO (1) */
    $ValordePreferencia[j] = 0$; /* Valor de Preferência (0) */
    $OR = \frac{LR(A_j=0)}{LR(A_j=1)}$; /* ODDS RATIO (0) */
    se $LR(A_j = 0) < LR(A_j = 1)$ **então**
      $ValordePreferencia[j] = 1$; /* Valor de Preferência (1) */
      $OR = \frac{LR(A_j=1)}{LR(A_j=0)}$; /* ODDS RATIO (1) */
      $attr = \overline{attr}$; /* Inversão dos valores no atributo */
  fim
  $FlexRankAtributos \leftarrow x$ ordenado de acordo com $OR(A_j)$;
  $Particao \leftarrow$ instâncias do conjunto de treino;
  para $j \leftarrow 1$ **até** $FlexRankAtributos$ **faça**
    se $VP[j] + FN[j] = 0$ *ou* $FP[j] + VN[j] = 0$ **então**
      $AUCINC \leftarrow 0$; /* Cálculo de Ganho = 0 */
    senão
      $AUCINC \leftarrow 0$;
      para P *em* $Particao$ **faça**
        ; /* Criação da Matriz de Confusão para o atributo */
        $\begin{bmatrix} inst_+ \\ inst_- \end{bmatrix} = \begin{bmatrix} \overline{attr}^t & attr^t \end{bmatrix} \times [P]$
        $\begin{bmatrix} VP & FN \\ FP & VN \end{bmatrix} = \begin{bmatrix} inst_+^t \\ inst_-^t \end{bmatrix} \times \begin{bmatrix} \bar{y} & y \end{bmatrix}$;
        $Pos \leftarrow VP + FN$;
        $Neg \leftarrow VN + FP$;
        $AUCINC = AUCINC + \frac{VP.Neg - Pos.FP}{2}$; /* Cálculo do Ganho */
        adicionar $inst_+$ e $inst_-$ em $lista\_profundidade$;
      fim
      se $AUCINC > limiar$ **então**
        $Particao \leftarrow lista\_profundidade$;
        $AtributoRelevante[i] \leftarrow j$;
  fim
  $FlexRankAtributos \leftarrow AtributoRelevante[\,]$;
  retorna $FlexrankAtributos, ValorDePreferencia$;
fim

3.4 Considerações Finais

Realizada a descrição dos algoritmos LexRank e FlexRank é necessário aplicar etapas de experimentos em ambos algoritmos com os demais classificadores citados no Capítulo 2, com o intuito de avaliar o desempenho de ambos classificadores. O objetivo em utilizar os dois algoritmos citados nesta seção é a contribuição na área de AM com uso de algoritmos legitimamente *rankeadores* e que auxiliem problemas práticos onde a relação de ordem é mais interessante que a classe.

A próxima seção detalha os experimentos realizados em todos algoritmos citados no Capítulo 2 em comparação a LexRank e FlexRank em conjunto de dados textuais e não textuais. Aborda também quais as modificações foram realizadas nas implementações de LexRank e FlexRank para comparação igualitária com algoritmos desenvolvidos pela biblioteca Scikit-Learn e os ajustes de parâmetros necessários em todos os algoritmos para um desempenho satisfatório em ambos experimentos.

CAPÍTULO

4

Análise Experimental dos Algoritmos LexRank e FlexRank

Neste capítulo é reportada a fase experimental dos algoritmos LexRank e FlexRank, bem como quais foram os resultados de ambos em comparação aos demais algoritmos de classificação vistos no Capítulo 2. As seções a seguir especificam: as bases de dados utilizadas neste trabalho, etapas de treinamento e teste, experimentos realizados e por fim a discussão dos resultados obtidos.

4.1 Metodologia da Avaliação Experimental

Os Algoritmos LexRank (Matsubara, 2008) e FlexRank foram comparados com os algoritmos Naive Bayes (NB), Máquina de Vetores de Suporte (SVM), K-Vizinhos mais Próximos (KNN) e Árvore de Decisão (DT). Para que essa comparação fosse justa em termos de tempo de execução, os algoritmos LexRank e FlexRank foram implementados utilizando estruturas matriciais e cálculos estatísticos com complexidade de tempo similiar aos algoritmos implementados na biblioteca Scikit-Learn (Pedregosa et al., 2011). Nesta comparação, foram utilizadas diversas bases de dados, especificadas na próxima seção.

4.1.1 Bases de Dados Utilizadas

Foram selecionados 26 conjuntos de dados da UCI (Asuncion and Newman, 2007) e 7 conjuntos textuais das mais diversas áreas (ex: fóruns, medicina, economia, tecnologia) para realização dos experimentos, (Rossi et al.,

2013). São conjuntos que possuem atributos nominais, poucos valores ausentes (*missing values*) e em sua grande maioria apenas duas classes. Para conjuntos com mais de duas classes (multi-classe), foi definida apenas uma classe como positiva e as demais classes como negativa (*um versus todos*).

Na Tabela 4.1, são apresentados os conjuntos de dados utilizados com os seguintes campos: nome do conjunto (Conjunto), o número de atributos (#Atrib), o número de exemplos (#Exem) e número de partições de validação cruzada (CROSSVALIDATION- CV) [1] utilizado para cada conjunto. Os conjuntos com número de exemplos menor que de 250 são rotulados com símbolo "*". Os colchetes informam que o conjunto tem mais de duas classes (*multi-classe*) e o valor entre os colchetes representa a classe selecionada como positiva.

Os conjuntos de dados foram avaliados por todos classificadores deste trabalho. As mesmas partições foram usadas em todos classificadores, de maneira a possibilitar uma comparação igualitária entre os algoritmos.

Os algoritmos foram ajustados para trabalharem com as versões binárias de cada conjunto em avaliação. Com isto, ajustes de parâmetros foram realizados nos algoritmos: LEXRANK, FLEXRANK, NB, SVM, KNN, os quais serão especificados na Seção 4.1.3.

4.1.2 Pré-Processamento das Bases

Para configuração dos experimentos, os conjuntos de dados UCI (Asuncion and Newman, 2007) foram discretizados com uso da discretização não supervisionada de dez partições (*ten-bin*) para atributos contínuos. Este procedimento foi realizado dentro do ambiente WEKA (Witten and Frank, 2005), com uso dos filtros: DISCRETIZE (Irani, 1993) e NOMINALTOBINARY (Breiman et al., 1984). Para os conjuntos textuais, realizou-se a conversão dos dados para valores binários considerando a seguinte regra: atributos com valores iguais a 0 permanecem inalteráveis e atributos iguais ou maiores a 1, faz-se alteração para o valor 1. Por fim, os atributos com valores ausentes, foram removidos do conjunto de dados.

Em relação ao CV, conjuntos com menos de 250 exemplos foram avaliados com apenas 5 partições. Conjuntos com mais de 250 exemplos, tiveram seu valor de CV em 10 partições. Para conjuntos com mais de duas classes, foi selecionada a classe minoritária para ser representada como classe positiva e as demais convertidas em classe negativa. Todos conjuntos foram armazenados em matrizes com uso da biblioteca NUMPY (Van Der Walt et al., 2011), com a finalidade de facilitar operações matriciais, bem como a manipulação dos dados.

[1] http://scikit-learn.org/stable/modules/cross_validation.html

Tabela 4.1: Conjunto de exemplos da UCI e Textuais utilizados nos experimentos.

Conjunto UCI				
#	Conjunto	#Atrib	#Exem	#CV
1	anneal[3]	145	898	10
2	breast-cancer	49	277	10
3	breast-w	91	683	10
4	car[unacc]	22	1728	10
5	credit-a	98	653	10
6	credit-g	125	1000	10
7	dermatology[1]	139	358	10
8	diabetes	81	768	10
9	*glass	91	214	5
10	haberman	33	306	10
11	*hayes-roth[1]	41	160	5
12	heart-statlog	131	270	10
13	ionosphere	332	351	10
14	kr-vs-kp	41	3196	10
15	letter[A]	161	20000	10
16	liver-disorders	61	345	10
17	*lymph[met]	66	148	5
18	mfeat-pixel[one]	1649	2000	10
19	*molec-bio	229	106	5
20	monks-1	16	556	10
21	monks-2	16	601	10
22	monks-3	16	554	10
23	*postoper-pat[A]	22	87	5
24	sonar	601	208	10
25	splice[EI]	288	267	10
26	tic-tac-toe	28	3190	10
Conjunto Textual				
27	Hitech [Medical]	12941	2301	10
28	Review_Polarity	15697	2000	10
29	CSTR [I.A]	1725	299	10
30	SyskillWebert [BioMedical]	5476	345	10
31	IrishEconomicSentiment [posite]	8658	1660	10
32	Classic3 [cacm]	7748	7095	10
33	LATimes [metro]	6279	6279	10

4.1.3 Etapas de Treino e Teste

Na etapa de treino, alguns ajustes foram necessários para que os algoritmos em avaliação fossem compatíveis com as versões binárias dos conjuntos. A implementação utilizada para construção dos algoritmos de classificação, foi retirada da biblioteca SCIKIT-LEARN (Pedregosa et al., 2011) desenvolvido em PYTHON (Bird et al., 2009), com aplicação de validação cruzada e ajuste de parâmetro automatizado.

O ajuste de parâmetro automatizado, ocorreu pelo uso da função RANDOMIZEDSEARCHCV (Bergstra and Bengio, 2012), o qual implementa um método de ajuste e previsão, com objetivo de encontrar bons parâmetros para os algoritmos.

Na Tabela 4.2 são mostradas quais foram os parâmetros ajustados por cada algoritmo na etapa de teste. O parâmetro representa a variável submetida para ajuste e o valor um resultado discreto ou escala.FLEXRANK ajustou seu limiar de ganho pelo parâmetro (THRESHOLD) com variação entre 0 a 0.5, dada a métrica de avaliação AUC. O algoritmo SVM usou a função de base radial `rbf`. NB fez uso da função `BernoulliNB` com `alpha = 0,02` por *default*, usado quando atributos avaliados possuem valor binário. KNN utilizou a métrica `matching` com ajuste do parâmetro `n_neighbors`. Nos classificadores LEXRANK e DT, foram utilizados os parâmetros *default*.

Tabela 4.2: Ajustes de Parâmetros: FLEXRANK, SVM, NB, KNN.

Algoritmo	Parâmetro	Valor
FLEXRANK	THRESHOLD	`0 ⟷ 0,5`
SVM	C	`scipy.stats.expon(scale=100)`
	GAMMA	`scipy.stats.expon(scale=.1)`
	KERNEL	`rbf`
NB	ALPHA	`0,02`
KNN	N_NEIGHBORS	`np.arange(1,11)`
	METRIC	`matching`

A implementação, por padrão, faz validação cruzada interna de três partições para encontrar os melhores parâmetros. O método de ajuste de parâmetros RANDOMIZEDSEARCHCV é iterativo e para todos os algoritmos foram utilizados 10 iterações. A validação cruzada foi realizada pelo método K-FOLD com partições estratificadas, o que garante a mesma proporção de exemplos para cada classe do conjunto. Por fim, o *score* adotado pelo RANDOMIZEDSEARCHCV para definição dos melhores parâmetros foi o `(roc_auc)`.

Todos os algoritmos buscaram ajustar seus classificadores com base na medida de avaliação da análise ROC, que visa analisar os classificadores sobre uma perspectiva de desempenho em relação aos exemplos de cada partição com taxa de acertos para VP (Verdadeiro Positivo) e FP (Falso Positivo). Logo, para os testes realizados neste trabalho, quanto maior o valor de AUC, melhor será o desempenho do classificador. Há, também, uma medida de tempo para avaliar qual a duração de treino e teste de cada classificador. A biblioteca utilizada neste procedimento foi a TIMEIT (Bird et al., 2009), desenvolvida em PYTHON e seu tempo calculado em segundos.

Logo após a etapa de treinamento e com os classificadores aptos a realizarem predições, dá-se início à etapa de teste, a qual utiliza exemplos não avaliados no processo de treinamento. Nesta etapa, obtém-se o valor de AUC para cada partição, o tempo de execução e por fim calcula-se a média dos respectivos valores. Os parâmetros obtidos na etapa de treinamento foram

aplicados nas partições de teste e seus resultados submetidos à medida de avaliação da análise ROC. A biblioteca utilizada para avaliação ROC foi a SKLEARN.METRICS [2], sendo analisadas as classes reais do conjunto de dados e as classes preditas pelos classificadores.

Na Tabela 4.3 são apresentadas as bibliotecas do pacote de códigos SCIKIT-LEARN (Pedregosa et al., 2011) utilizados neste trabalho.

Tabela 4.3: Bibliotecas utilizadas em Sklearn.

Biblioteca	Função
`Cross_validation`	Usado na validação cruzada pelos conjuntos
`Metrics`	Cálculo da Curva ROC (AUC)
`RandomizedSearchCV`	Ajuste de Parâmetros
`SVM`	Código Support Vector Machine
`BernoulliNB`	Código Naive Bayes
`KNeighborsClassifier`	Código k-Nearest-Neighbors
`Tree`	Código Decision Tree
`Utils.extmath`	Usado nas operações matriciais com os algoritmo LEXRANK e FLEXRANK
`BaseEstimator`	Classe base para classificadores em Sklearn

Por fim, os valores obtidos na avaliação ROC foram tabulados para aferição e cálculos estatísticos. A etapa de teste será descrita mais detalhadamente nas Seções 4.2.1 e 4.2.2, juntamente com os resultados obtidos em cada experimento.

4.2 Avaliação Experimental

Esta seção descreve a avaliação experimental dos classificadores citados no Capítulos 2 e 3. Todos os experimentos foram desenvolvidos com base nas descrições das etapas de teste e treino apresentadas anteriormente.

4.2.1 Experimento 1: Execução dos Algoritmos com Conjuntos de Dados UCI

Por convenção, todos os conjuntos foram binarizados, dado que estudos anteriores não apontaram diferença estatística significativa (Matsubara, 2008) entre os conjuntos reais e os binarizados. O primeiro experimento foi realizar a classificação dos conjuntos UCI, apresentada na Tabela 4.4. Para este procedimento foram usados os algoritmos FLEXRANK, LEXRANK, NB, SVM, KNN e DT, os quais passaram por ajuste de parâmetro na etapa de treinamento e uso de validação cruzada, já descrito nas Seções 4.1.2 e 4.1.3.

[2] http://scikit-learn.org/stable/modules/classes.html#sklearn-metrics-metrics

Tabela 4.4: Média de AUC dos Algoritmos (LexRank, FlexRank, NB, SVM, DT, KNN) - Conjunto de Dados UCI.

#	Conjuto	LexRank	FlexRank	SVM	NB	DT	KNN
1	anneal[3]	0,986	0,950	**1,000**	0,978	0,988	0,998
2	breast-cancer	0,729	0,719	0,730	**0,731**	0,639	0,679
3	breast-w	0,991	0,982	**0,994**	0,987	0,923	0,990
4	car[unacc]	0,941	0,917	**1,000**	0,990	0,983	0,974
5	credit-a	**0,920**	0,908	0,900	0,908	0,810	0,894
6	credit-g	0,753	0,711	**0,787**	0,786	0,616	0,742
7	dermatology[1]	**1,000**	0,999	**1,000**	**1,000**	0,980	**1,000**
8	diabetes	**0,826**	0,808	0,804	0,810	0,645	0,742
9	*glass	0,960	0,917	0,973	**0,986**	0,917	0,952
10	haberman	0,625	0,611	0,619	**0,652**	0,574	0,591
11	*hayes-roth[1]	0,855	0,848	**0,918**	0,916	0,828	0,704
12	heart-statlog	0,833	0,819	0,888	**0,914**	0,783	0,879
13	ionosphere	0,955	0,896	**0,975**	0,930	0,873	0,934
14	kr-vs-kp	0,977	0,763	**0,999**	0,951	0,996	0,989
15	letter[A]	0,964	0,947	**1,000**	0,966	0,954	0,994
16	liver-disorders	0,625	0,592	**0,696**	0,683	0,613	0,668
17	*lymph[met]	0,863	0,799	0,913	**0,916**	0,824	0,880
18	mfeat-pixel[one]	0,996	**0,998**	0,599	0,967	0,958	0,997
19	*molec-bio	0,910	0,912	0,696	**0,958**	0,783	0,921
20	monks-1	0,683	0,729	**1,000**	0,711	0,973	**1,000**
21	monks-2	0,478	0,484	**1,000**	0,535	0,985	0,619
22	monks-3	0,983	0,973	0,986	0,984	0,975	**0,991**
23	*postoper-pat[A]	0,426	0,494	**0,556**	0,416	0,451	0,383
24	sonar	0,697	**0,918**	0,882	0,819	0,673	0,875
25	splice[EI]	0,991	0,995	**0,996**	0,994	0,949	0,969
26	tic-tac-toe	0,731	0,525	**1,000**	0,751	0,942	0,817
	AUC Médio	**0,834**	**0,816**	**0,881**	**0,855**	**0,832**	**0,853**

Com base nos resultados da Tabela 4.4, SVM obteve o maior valor de AUC Médio em comparação aos outros algoritmos. LexRank e FlexRank tiveram resultados semelhantes na maioria dos conjuntos e com valores próximos ao algoritmo NB, sendo a diferença de 0,021 (LexRank x NB) e 0,039 (FlexRank x NB) na média de AUC respectivamente. KNN e NB apesar da diferença de resultados em cada conjunto, obtiveram médias próximas de AUC entre os 26 conjuntos. DT, obteve a penúltima colocação no ranking de desempenho, ficando a frente apenas do algoritmo FlexRank e próximo a LexRank.

Os resultados da Tabela 4.4 são apresentados na Figura 4.1 em ordem crescente de atributos, entre os algoritmos LexRank, FlexRank, SVM e NB. O gráfico permite analisar pontos similares e diferentes entre SVM e os demais algoritmos, o qual demonstra a diferença de AUC entre os conjuntos pelos classificadores em avaliação.

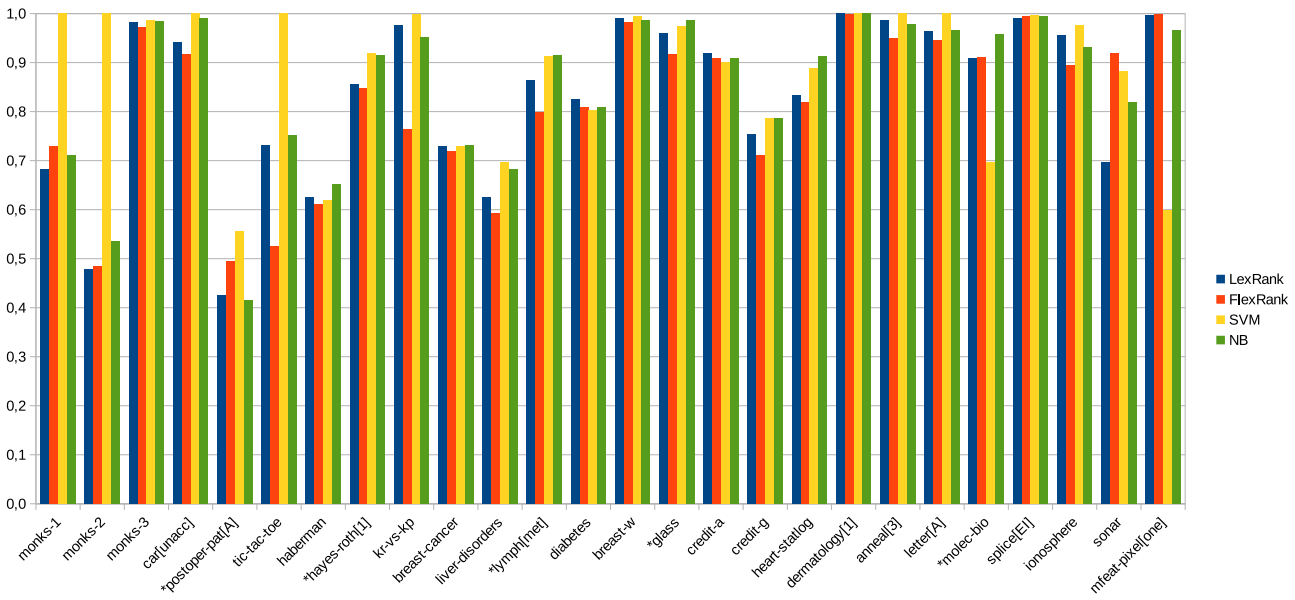

Figura 4.1: Análise das médias AUC entre os algoritmos: LexRank, FlexRank, SVM e NB.

Na Tabela 4.5, detalha-se o tempo de treino e teste gerados pelos algoritmos em análise. Sua medida padrão foi calculada em segundos e obtida pela média das partições de cada conjunto.

Destacam-se respectivamente como maior e menor resultado em tempo de treinamento os algoritmos NB e SVM, ambos possuem estratégias diferentes no treino, o que justifica o resultado apresentado. Devido à dimensionalidade dos dados e ajuste dos vetores de suporte, SVM tem uma diferença de tempo 4,205 segundos, se comparado com FlexRank, seu concorrente mais próximo em tempo de treinamento. As médias de LexRank e FlexRank nestes conjuntos estão acima do algoritmo SVM, porém inferiores a NB, KNN e DT. Tanto LexRank quanto FlexRank possuem estratégias iguais. Porém, o tempo de FlexRank é maior ao ter que ordenar lexigraficamente todos os atributos e realizar a seleção de atributos nesta etapa.

O algoritmo KNN, apesar de ser considerado um algoritmo *lazy*, na implementação do Scikit-Learn, durante o treinamento, são construídos estruturas auxiliares como *ball tree* e *KD tree* para diminuir o tempo de classificação. Deste modo, o tempo de treinamento do KNN é considerável.

No tempo de teste, o algoritmo DT leva vantagem sobre os algoritmos abordados neste trabalho. No entanto, destaca-se o tempo de FlexRank em relação aos demais algoritmos, logo atrás de NB. Apesar da tênue diferença de AUC entre os algoritmos LexRank e FlexRank em relação a SVM e NB, há indícios de que a execução e o tempo de classificação do algoritmo FlexRank têm uma ligeira vantagem entre os demais algoritmos, por ser legitimamente um algoritmo rankeador e realizar a classificação de novos exemplos com uma quantidade mínima de atributos.

Tabela 4.5: Média de Tempo dos Algoritmos (FLEXRANK, LEXRANK, NB, SVM, DT, KNN) - Conjunto de Dados UCI.

#	Conjuto	LEXRANK		FLEXRANK		SVM		NB		DT		KNN	
		Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste
1	anneal[3]	0,0227	0,0270	0,0737	0,0020	0,6825	0,0102	0,0090	0,0012	**0,0039**	**0,0002**	0,0044	0,0248
2	breast-cancer	0,0026	0,0029	0,0117	0,0008	0,0342	0,0006	0,0017	0,0003	0,0013	**0,0001**	**0,0007**	0,0014
3	breast-w	0,0108	0,0129	0,0423	0,0018	0,1347	0,0024	0,0045	0,0007	0,0041	**0,0002**	**0,0024**	0,0081
4	car[unacc]	0,0074	0,0090	0,0262	0,0042	0,3398	0,0035	0,0034	0,0006	**0,0021**	**0,0002**	0,0038	0,0076
5	credit-a	0,0111	0,0133	0,0446	0,0012	0,2158	0,0035	0,0046	0,0007	0,0050	**0,0002**	**0,0023**	0,0113
6	credit-g	0,0234	0,0264	0,0716	0,0033	0,9039	0,0165	0,0084	0,0011	0,0116	**0,0002**	**0,0043**	0,0328
7	dermatology[1]	0,0086	0,0106	0,0359	0,0011	0,0591	0,0011	0,0039	0,0006	0,0017	**0,0001**	**0,0015**	0,0045
8	diabetes	0,0106	0,0129	0,0387	0,0027	0,2917	0,0045	0,0045	0,0007	0,0061	**0,0002**	**0,0024**	0,0119
9	*glass	0,0031	0,0079	0,0220	0,0015	0,0182	0,0007	0,0019	0,0005	0,0011	**0,0001**	**0,0006**	0,0021
10	haberman	0,0020	0,0023	0,0103	0,0007	0,0317	0,0006	0,0016	0,0003	0,0012	**0,0001**	**0,0007**	0,0011
11	*hayes-roth[1]	0,0013	0,0029	0,0083	0,0009	0,0077	0,0003	0,0014	0,0003	**0,0005**	**0,0001**	**0,0005**	0,0007
12	heart-statlog	0,0062	0,0072	0,0237	0,0007	0,0495	0,0010	0,0031	0,0005	0,0019	**0,0002**	**0,0011**	0,0027
13	ionosphere	0,0211	0,0251	0,0716	0,0008	0,2435	0,0046	0,0079	0,0011	0,0138	**0,0003**	**0,0028**	0,0095
14	kr-vs-kp	0,0261	0,0302	0,0636	0,0048	0,9198	0,0078	0,0102	0,0013	**0,0061**	**0,0002**	0,0168	0,1152
15	letter[A]	0,6829	0,7351	1,5264	0,0698	52,9296	0,5729	**0,2651**	0,0241	0,8260	**0,0015**	1,2193	17,1923
16	liver-disorders	0,0040	0,0047	0,0194	0,0012	0,0624	0,0011	0,0024	0,0004	0,0020	**0,0002**	**0,0010**	0,0022
17	*lymph[met]	0,0018	0,0043	0,0132	0,0008	0,0083	0,0005	0,0016	0,0004	0,0007	**0,0001**	**0,0004**	0,0010
18	mfeat-pixel[one]	0,7702	0,7358	2,0461	0,0053	37,7464	0,7843	0,2211	0,0238	**0,1635**	**0,0013**	0,1942	1,7116
19	*molec-bio	0,0047	0,0109	0,0502	0,0004	0,0152	0,0009	0,0026	0,0007	0,0017	**0,0002**	**0,0006**	0,0015
20	monks-1	0,0021	0,0024	0,0076	0,0015	0,6088	0,0015	0,0019	0,0004	**0,0009**	**0,0002**	**0,0009**	0,0016
21	monks-2	0,0023	0,0025	0,0088	0,0017	1,7075	0,0016	0,0020	0,0004	0,0011	**0,0002**	**0,0009**	0,0015
22	monks-3	0,0021	0,0023	0,0072	0,0016	0,1135	0,0009	0,0018	0,0004	**0,0008**	**0,0002**	**0,0008**	0,0017
23	*postoper-pat[A]	0,0006	0,0010	0,0044	0,0004	0,0037	0,0003	0,0012	0,0003	0,0005	**0,0001**	**0,0004**	0,0004
24	sonar	0,0232	0,0278	0,1517	0,0003	0,1554	0,0035	0,0092	0,0014	0,0116	**0,0003**	**0,0024**	0,0066
25	splice[EI]	0,1844	0,2118	0,5083	0,0103	15,3939	0,2278	**0,0618**	0,0075	0,0699	**0,0007**	0,0746	0,8288
26	tic-tac-toe	0,0054	0,0066	0,0161	0,0012	0,7959	0,0022	0,0030	0,0005	0,0022	**0,0002**	**0,0021**	0,0071
	Média Tempo	**0,0708**	**0,0744**	**0,1758**	**0,0044**	**4,3643**	**0,0636**	**0,0246**	**0,0027**	**0,0439**	**0,0003**	**0,0593**	**0,7689**

Tabela 4.6: Teste de Friedman (FLEXRANK, LEXRANK, NB, SVM, DT, KNN) - Conjunto de Dados UCI.

#	Conjuto	LEXRANK	FLEXRANK	SVM	NB	DT	KNN
1	anneal[3]	0,986 (4,0)	0,950 (6,0)	**1,000 (1,0)**	0,978 (5,0)	0,988 (3,0)	0,998 (2,0)
2	breast-cancer	0,729 (3,0)	0,719 (4,0)	0,730 (2,0)	**0,731 (1,0)**	0,639 (6,0)	0,679 (5,0)
3	breast-w	0,991 (2,0)	0,982 (5,0)	**0,994 (1,0)**	0,987 (4,0)	0,923 (6,0)	0,990 (3,0)
4	car[unacc]	0,941 (5,0)	0,917 (6,0)	**1,000 (1,0)**	0,990 (2,0)	0,983 (3,0)	0,974 (4,0)
5	credit-a	0,920 (1,0)	0,908 (3,0)	0,900 (4,0)	0,908 (2,0)	0,810 (6,0)	0,894 (5,0)
6	credit-g	0,753 (3,0)	0,711 (5,0)	**0,787 (1,0)**	0,786 (2,0)	0,616 (6,0)	0,742 (4,0)
7	dermatology[1]	1,000 (4,0)	0,999 (5,0)	**1,000 (1,5)**	**1,000 (1,5)**	0,980 (6,0)	1,000 (3,0)
8	diabetes	**0,826 (1,0)**	0,808 (3,0)	0,804 (4,0)	0,810 (2,0)	0,645 (6,0)	0,742 (5,0)
9	*glass	0,960 (3,0)	0,917 (6,0)	0,973 (2,0)	**0,986 (1,0)**	0,917 (5,0)	0,952 (4,0)
10	haberman	0,625 (2,0)	0,611 (4,0)	0,619 (3,0)	**0,652 (1,0)**	0,574 (6,0)	0,591 (5,0)
11	*hayes-roth[1]	0,855 (3,0)	0,848 (4,0)	**0,918 (1,0)**	0,916 (2,0)	0,828 (5,0)	0,704 (6,0)
12	heart-statlog	0,833 (4,0)	0,819 (5,0)	0,888 (2,0)	**0,914 (1,0)**	0,782 (6,0)	0,879 (3,0)
13	ionosphere	0,955 (2,0)	0,896 (5,0)	**0,975 (1,0)**	0,930 (4,0)	0,873 (6,0)	0,934 (3,0)
14	kr-vs-kp	0,977 (4,0)	0,763 (6,0)	**0,999 (1,0)**	0,951 (5,0)	0,996 (2,0)	0,989 (3,0)
15	letter[A]	0,964 (4,0)	0,947 (6,0)	**1,000 (1,0)**	0,966 (3,0)	0,954 (5,0)	0,994 (2,0)
16	liver-disorders	0,625 (4,0)	0,592 (6,0)	**0,696 (1,0)**	0,683 (2,0)	0,613 (5,0)	0,668 (3,0)
17	*lymph[met]	0,863 (4,0)	0,799 (6,0)	0,913 (2,0)	**0,916 (1,0)**	0,824 (5,0)	0,880 (3,0)
18	mfeat-pixel[one]	0,996 (3,0)	**0,998 (1,0)**	0,599 (6,0)	0,967 (4,0)	0,958 (5,0)	0,997 (2,0)
19	*molec-bio	0,910 (4,0)	0,912 (3,0)	0,696 (6,0)	**0,958 (1,0)**	0,783 (5,0)	0,921 (2,0)
20	monks-1	0,683 (6,0)	0,729 (4,0)	**1,000 (1,0)**	0,711 (5,0)	0,973 (3,0)	1,000 (2,0)
21	monks-2	0,478 (6,0)	0,484 (5,0)	**1,000 (1,0)**	0,535 (4,0)	0,985 (2,0)	0,619 (3,0)
22	monks-3	0,983 (4,0)	0,973 (6,0)	0,986 (2,0)	0,984 (3,0)	0,975 (5,0)	0,991 (1,0)
23	*postoper-pat[A]	0,426 (4,0)	0,494 (2,0)	**0,556 (1,0)**	0,416 (5,0)	0,451 (3,0)	0,383 (6,0)
24	sonar	0,697 (5,0)	**0,918 (1,0)**	0,882 (2,0)	0,819 (4,0)	0,673 (6,0)	0,875 (3,0)
25	splice[EI]	0,991 (4,0)	0,995 (2,0)	**0,996 (1,0)**	0,994 (3,0)	0,949 (6,0)	0,969 (5,0)
26	tic-tac-toe	0,731 (5,0)	0,525 (6,0)	**1,000 (1,0)**	0,751 (4,0)	0,942 (2,0)	0,817 (3,0)
	Ranking Médio	**3,615**	**4,423**	**1,942**	**2,788**	**4,769**	**3,462**

Realizado os cálculos de AUC sobre os conjuntos UCI, da-se início ao teste de Friedman (Demsar, 2006) mostrada na Tabela 4.6. O teste de Friedman é um teste não-paramétrico, que tem por objetivo comparar dados amostrais relacionados, quando dados são avaliados mais de uma vez em um mesmo experimento. Este tipo de análise não visa examinar os dados brutos numericamente, consiste apenas em avaliar os dados obtidos após a ordenação de cada conjunto separadamente. Após a ordenação de cada conjunto, realiza-se o teste de hipótese, que visa aferir a igualdade ou diferença estatística entre conjuntos em análise.

O cálculo da estatística F realizado no teste de Friedman, obteve o seguintes resultados: a estatística F é de 11,19 para os valores obtidos. O valor crítico utilizando a estatística F com 5 e 125 graus de liberdade a 95% é de 2,29. De acordo com o teste de Friedman, usando o valor da estatística F, é rejeitada a hipótese-nula de que todos os algoritmos se comportam da mesma forma, pois o valor de 11,19 sobrepõe o limite de 2,29.

Por fim, na Figura 4.2 é abordada de maneira estatística o *Diagrama de Diferença Crítica* entre os algoritmos pelo teste de Nemenyi post-hoc, reafirmando os resultados obtidos na Tabela 4.6. O valor crítico para a comparação da média dos rankings de dois algoritmos diferentes a 95% de liberdade é 1,48. As diferenças de média dos rankings acima desse valor são significativas. Logo, os resultados iniciais indicam que o algoritmo SVM possui diferença significativa entre os algoritmos LexRank, FlexRank, KNN e DT.

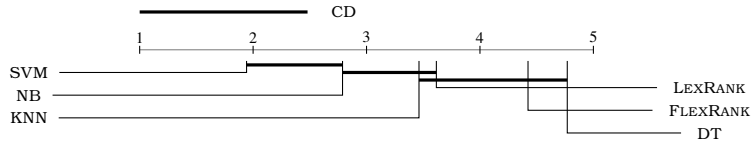

Figura 4.2: Diagrama de Diferença Crítica do AUC ROC. Linhas conectadas mostram que não há diferença significativa no percentil 95.

4.2.2 Experimento 2: Execução dos Algoritmos com Bases Textuais

Esta segunda etapa contou apenas com conjuntos textuais e foi necessária para validar o real objetivo deste trabalho. Conjuntos textuais são grandes o suficiente para averiguar os algoritmos de classificação e obter resultados precisos quanto ao tempo de execução, taxas de acerto e diferenças entre os métodos de cada classificador. São conjuntos que possuem em média 2000 a 16000 atributos, o que os torna conjuntos de dados esparsos e relativamente custosos na etapa de classificação.

Na Tabela 4.7 são apresentados as médias de AUC dos conjuntos relacionados aos algoritmos de classificação já citados no primeiro experimento

(Seção 4.2.1). No ranking de desempenho, esteve a frente, os algoritmos SVM e NB. No entanto, LexRank e FlexRank obtiveram resultados melhores se comparado aos valores obtidos nos conjuntos de dados UCI. Outro aspecto importante nos dados analisados é que FlexRank consegue em alguns conjuntos ter valores de AUC melhores que seu algoritmo antecessor, o LexRank. Esse fato ocorre porque em conjuntos esparsos a seleção de atributos feita por FlexRank, torna-se mais rápida na etapa de classificação.

Tabela 4.7: Média de AUC dos Algoritmos (LexRank, FlexRank, NB, SVM, DT, KNN) - Conjunto de Dados Textuais.

#	Conjuto	LexRank	FlexRank	SVM	NB	DT	KNN
27	Hitech [Medical]	0,813	**0,997**	0,888	0,831	0,657	0,600
28	Review_Polarity	0,708	0,809	**0,885**	0,854	0,642	0,620
29	CSTR [I.A]	0,848	0,782	0,849	**0,873**	0,658	0,723
30	SyskillWebert [BioMedical]	**0,985**	0,960	0,984	0,979	0,858	0,795
31	IrishEconomicSentiment [posite]	0,646	0,774	**0,797**	0,750	0,600	0,675
32	Classic3 [cacm]	0,976	**0,995**	0,992	0,979	0,916	0,576
33	LATimes [metro]	0,895	0,699	**0,942**	**0,942**	0,829	0,710
	AUC Médio	**0,839**	**0,860**	**0,905**	**0,887**	**0,737**	**0,671**

Os tempos de treino e teste visualizados na Tabela 4.8 correspondem as médias de cada conjunto e mostram valores relativamente diferentes do conjunto de dados UCI. No tempo de treino, destaca-se o classificador NB com o menor valor para realização de treino entre os 7 conjuntos avaliados e SVM o maior tempo de treino. Os tempos entre LexRank e FlexRank, diferem em ambas as etapas devido à seleção de atributos realizada pelo FlexRank. No entanto, dada a quantidade de atributos nos conjuntos textuais, FlexRank possui um tempo de treino plausível em relação aos algoritmos avaliados.

O tempo de teste obtido reafirma o conceito descrito no Capítulo 3, evidenciando o rápido processo de classificação do algoritmo FlexRank em conjuntos de dados relativamente grandes. O tempo de teste do algoritmo FlexRank foi o menor no segundo experimento, com SPEEDUP de 2,0 em relação ao seu concorrente mais próximo DT na média de tempo. Outro dado relevante é que o FlexRank e DT possuem tempos similares na etapa de treino e teste, no entanto com valores AUC diferentes. Esta diferença permite inferir que o método aplicado a FlexRank é mais efetivo em conjuntos de dados textuais.

O Teste de Friedman também foi aplicado neste experimento, a fim de analisar a diferença estatística entre os algoritmos. Na Tabela 4.9, é possível observar resultados obtidos pelos classificadores, com LexRank e FlexRank ocupando a terceira e quarta posição no ranking de desempenho. DT e KNN ficaram com os menores valores e SVM com o maior valor para os resultados deste experimento. O cálculo da estatística F realizado no teste de Friedman,

Tabela 4.8: Média de Tempo dos Algoritmos (FLEXRANK, LEXRANK, NB, SVM, DT, KNN) - Conjunto de Dados Textuais.

#	Conjuto	LEXRANK		FLEXRANK		SVM		NB		DT		Knn	
		Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste	Treino	Teste
27	Hitech [Medical]	8,766	6,600	20,653	**0,004**	430,029	7,906	**3,126**	0,324	22,014	0,019	3,376	17,328
28	Review_Polarity	9,098	7,279	22,672	**0,004**	429,066	8,946	3,186	0,335	9,599	0,018	**3,136**	15,946
29	CSTR [I.A]	0,105	0,123	0,567	**0,000**	0,843	0,017	0,030	0,004	0,074	0,000	**0,014**	0,034
30	SyskillWebert [BioMedical]	0,347	0,459	1,750	**0,001**	3,849	0,078	0,115	0,014	0,235	**0,001**	**0,049**	0,157
31	IrishEconomicSentiment [posite]	3,898	3,599	11,198	**0,002**	166,178	3,528	1,487	0,100	17,887	0,005	**1,133**	5,973
32	Classic3 [cacm]	14,808	12,569	36,569	**0,016**	983,016	19,338	**5,337**	0,360	19,952	0,017	6,662	84,034
33	LATimes [metro]	17,707	13,851	41,884	**0,015**	1460,682	28,959	**6,161**	0,422	35,308	0,021	8,747	96,483
	Média Tempo	**7,818**	**6,354**	**19,328**	**0,006**	**496,238**	**9,824**	**2,777**	**0,223**	**15,010**	**0,012**	**3,302**	**31,422**

obteve o valor de 11,04 para os valores obtidos. O valor crítico utilizando para estatística F com 5 e 30 graus de liberdade a 95% é de 2,53.

Por fim, na Figura 4.3 detalham-se os classificadores quanto ao *Diagrama de Diferença Crítica* pelo teste de Nemenyi post-hoc, não havendo diferença crítica entre os algoritmos SVM, NB, LexRank e FlexRank. O valor crítico para a comparação da média dos rankings de dois algoritmos diferentes a 95% de liberdade é 2,85. As diferenças de média dos rankings acima desse valor são significativas.

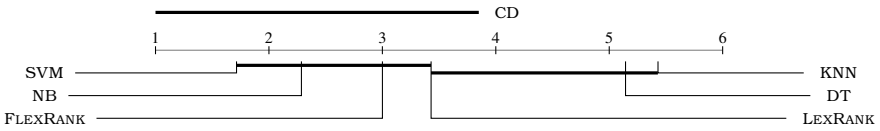

Figura 4.3: Diagrama de Diferença Crítica do AUC ROC. Linhas conectadas mostram que não há diferença significativa no percentil 95.

4.3 Discussão dos Resultados

Realizado os experimentos e com base nos valores alcançados, é possível descrever quais foram os aspectos positivos e negativos em relação aos objetivos deste trabalho. Esta seção visa detalhar de modo aprofundado todos os resultados atingidos pelos algoritmos LexRank e FlexRank em comparação aos algoritmos de classificação fundamentados no Capítulo 2.

Vale a pena utilizar algoritmos legitimamente lexicográficos e rankeadores no problema apresentado?

Para responder a essa questão, é necessário analisar os dois tipos de experimentos realizados neste trabalho. O primeiro critério que pode ser destacado é que algoritmos com características lexicográficas e puramente rankeadores, tendem a ter valores de AUC não muito eficientes para bases pequenas (Experimento 1), mesmo que na média final os valores de AUC entre os classificadores sejam próximos. Tal fato justifica-se pela pequena quantidade de exemplos e por que a relação de ordem para estes casos não afeta significativamente o valor de AUC. Para situações assim, talvez seja necessário utilizar classificadores com métodos diferentes na seleção de atributos para predição de novos exemplos.

Ao analisar conjuntos de dados textuais no segundo exemplo, os valores de AUC para LexRank e FlexRank são semelhantes aos algoritmos SVM e NB. Logo, para o problema proposto neste trabalho há indícios de que algoritmos legitimamente lexicográficos e rankeadores podem ser utilizados em conjuntos

Tabela 4.9: Teste de Friedman (FLEXRANK, LEXRANK, NB, SVM, DT, KNN) - Conjunto de Dados Textuais.

#	Conjuto	LEXRANK	FLEXRANK	SVM	NB	DT	KNN
27	Hitech	0,813 (4,0)	**0,997 (1,0)**	0,888 (2,0)	0,831 (3,0)	0,657 (5,0)	0,600 (6,0)
28	Review_Polarity	0,708 (4,0)	0,809 (3,0)	**0,885 (1,0)**	0,854 (2,0)	0,641 (5,0)	0,620 (6,0)
29	CSTR	0,848 (3,0)	0,782 (4,0)	0,849 (2,0)	**0,873 (1,0)**	0,658 (6,0)	0,723 (5,0)
30	SyskillWebert	**0,985 (1,0)**	0,960 (4,0)	0,984 (2,0)	0,979 (3,0)	0,858 (5,0)	0,795 (6,0)
31	IrishEconomicSentiment	0,646 (5,0)	0,774 (2,0)	**0,797 (1,0)**	0,750 (3,0)	0,600 (6,0)	0,675 (4,0)
32	Classic3	0,976 (4,0)	**0,995 (1,0)**	0,992 (2,0)	0,979 (3,0)	0,916 (5,0)	0,576 (6,0)
33	LATimes	0,895 (3,0)	0,699 (6,0)	0,942 (2,0)	**0,942 (1,0)**	0,829 (4,0)	0,710 (5,0)
	Ranking Médio	**3,429**	**3,000**	**1,714**	**2,286**	**5,143**	**5,429**

de dados massivos com eficiência equiparada aos algoritmos de classificação em AM.

No entanto, ao analisar o algoritmo FLEXRANK em ambos experimentos, é nítida a diferença nos valores de AUC entre o primeiro e segundo experimento, se comparado a SVM e NB. Ao combinar os métodos aplicados no algoritmo LEXRANK, juntamente com a seleção de atributos, FLEXRANK apresenta bons resultados para o valor de AUC. Deste modo sugere-se o uso do FLEXRANK para bases textuais por não ter diferença significativa com SVM em termos de AUC.

O algoritmo proposto neste trabalho apresenta tempo de classificação considerável para problemas em conjunto de dados massivos?

Problemas que necessitam de uma classificação rápida para compor a resolução de tarefas e atividades normalmente fazem uso de classificadores com baixo tempo de execução, seja na etapa de treino ou teste. Classificar conjuntos de dados massivos é uma tarefa custosa, principalmente quando exemplos são textuais e esparsos. Nos dois experimentos realizados neste trabalho, é possível destacar o algoritmo FLEXRANK como provável classificador a ser usado em problemas desta natureza. Apesar do tempo na etapa de treinamento não ser tão atrativo em relação aos algoritmos NB e DT, FLEXRANK possui um tempo relativamente baixo se comparado ao algoritmo que possui os melhores resultados deste trabalho, SVM.

Na Tabela 4.10 é detalhado o *speedup* de tempo dos algoritmos nos experimentos 1 e 2 deste trabalho na etapa de classificação (teste). Para os valores do primeiro experimento FLEXRANK fica atrás apenas do algoritmo DT com uma diferença miníma de -15,83. Ao trabalhar com conjuntos maiores no segundo experimento, FLEXRANK possui melhora quanto ao tempo de execução em relação aos algoritmo avaliados. Comparado ao SVM, FLEXRANK é quase duas mil vez mais rápido na etapa de classificação, o que permite inferir que para problemas que necessitam de classificação rápida, FLEXRANK torna-se uma boa escolha.

Tabela 4.10: Speedup de Tempo do Algoritmo FLEXRANK em relação aos algoritmos: LEXRANK, SVM, NB, DT, KNN. (Conjuntos UCI e Textual)

Speedup FLEXRANK					
Exp 1	FLEXRANK → LEXRANK	FLEXRANK → SVM	FLEXRANK → NB	FLEXRANK → DT	FLEXRANK → KNN
Speedup	15,99	13,67	0,58	-15,83	165,13
Exp 2	FLEXRANK → LEXRANK	FLEXRANK → SVM	FLEXRANK → NB	FLEXRANK → DT	FLEXRANK → KNN
Speedup	1055,58	1632,02	37,00	1,93	5219,81

Qual a porcentagem e relação de atributos selecionados entre os dois experimentos abordados neste trabalho?

Para efeito de comparação dos atributos selecionados, foi catalogada no Apêndice A através da Tabela A.1 pelo Experimento 4.2.1 e 4.2.2 respectivamente a relação de atributos selecionados por conjunto e qual porcentagem de atributos utilizada por FLEXRANK dado o total de atributos. O primeiro experimento na média geral obteve um total de 4572 atributos somado a todos os conjuntos, a qual foi necessário apenas 12,3% dos atributos para realizar a classificação dos exemplos no conjunto de dados UCI por FLEXRANK. Já no segundo experimento com um total de 58524 atributos foi necessário 0,057% dos atributos para que FLEXRANK realizasse a classificação dos exemplos em bases textuais. A Tabela A.1 é representada pelo número do conjunto de dados, nome do conjunto, validação cruzada utilizada, número de exemplos, total de atributos, porcentagem utilizada por FLEXRANK na classificação e quantidade média de atributos utilizada em cada conjunto.

Já a Tabela A.2 também detalhada no Apêndice A, informa quais foram os atributos selecionados em cada conjunto de dados. Esse detalhamento teve por objetivo mapear quais foram os atributos mais utilizados pelo classificador e que obtiveram maior relação com o conjunto de dados em avaliação. A Tabela é representada pelo número do conjunto de dados, nome do conjunto e quais os atributos foram selecionados. Outra fato interessante é analisar os atributos selecionados neste trabalho como referência a trabalhos futuros sobre a perspectiva da redução de atributos em conjuntos de dados UCI e textuais testados nos experimentos deste projeto.

CAPÍTULO

5

Conclusões

Problemas de classificação são ainda uma das principais tarefas em Aprendizado de Máquina. Entretanto, nos últimos anos, a tarefa de aprendizado de *rankings* (*learning to rank*) tem chamado a atenção da comunidade científica e muito tem sido feito nesta área. O presente trabalho fez um estudo mais detalhado sobre um destes algoritmos conhecido como LEXRANK. Com enfoque em bases textuais durante o desenvolvimento do trabalho buscou-se melhorias do LEXRANK. Apesar de intuitivamente ser evidente que os primeiros atributos mais a esquerda do ranking lexicográfico são os mais importantes, ao observar o comportamento de LEXRANK no espaço ROC, foi possível confirmar esta propriedade que motivou o desenvolvimento do FLEXRANK.

A partir desta observação este trabalho propõem o algoritmo FLEXRANK que faz uma busca gulosa utilizando razão de chances de razão de verossimilhança como heurística e aceita apenas os atributos que incrementam o AUC ROC, similarmente aos métodos de seleção de atributos híbridos como a árvore de decisão realiza para selecionar os atributos.

Foi realizada duas avaliações experimentais, as quais foram importantes para nortear os principais objetivos do projeto. A primeira resultou de conjuntos pequenos UCI e não textuais, mas foi essencial para ajustar os algoritmos e destacar diferenças de comportamento entre os classificadores no primeiro e segundo experimento. A segunda composta por conjuntos textuais visou colocar a prova o algoritmo FLEXRANK em relação aos demais algoritmos descritos neste trabalho.

O FLEXRANK tem como pontos fortes:

- Ganhos significativos em termos de velocidade de construção de ranking (tempo de teste) quanto ao seu predecessor LEXRANK da ordem de 1055,58 vezes em tempo de processamento, e 1,93 vezes frente as árvores de decisão, que são consideradas estruturas rápidas de classificação para bases de texto;
- AUC ROC comparável, sem diferença significativa (p-valor = 0.05), com algoritmos consagrados para dados textuais como o SVM e NB.

A avaliação experimental suporta estes argumentos utilizando dois experimentos: Experimento 4.2.1 e 4.2.2. Onde o primeiro verifica o comportamento do algoritmo para 26 bases da UCI de propósito geral e o principal resultado é a comparação do FLEXRANK com SVM; e o segundo utilizando 7 bases de texto onde FLEXRANK mostra o melhor resultado em tempo de classificação que todos os outros métodos testados.

Otimizações também demonstraram ser eficientes no cálculo de estatísticas que envolvem uma coleção de dados significativamente grande como o caso dos conjuntos textuais. O uso de produto escalar na operação dos cálculos e da biblioteca SCIKIT-LEARN auxiliou no processo de classificação e no tempo de execução de ambos algoritmos, em especial a LEXRANK e FLEXRANK.

Os resultados são motivadores dada a quantidade mínima de atributos necessários para realizar a classificação de novos exemplos pelo uso do Aprendizado Supervisionado. Excepcionalmente pelo ênfase dos classificadores legitimamente lexicográficos em comparação aos classificadores tradicionais.

5.1 Contribuições

Uma das contribuições deste trabalho foi a visualização de ranking lexicográficos sobre a perspectiva da análise ROC utilizando espaço de cobertura. Com esta análise é possível entender melhor o funcionamento de ranking lexicográfico e propor melhorias a esta classe de algoritmos.

Houve também a melhoria das propriedades extraídas do algoritmo LEXRANK, por meio do algoritmo FLEXRANK o que o torna interessante para aplicações que requerem classificação rápida e valores de AUC comparáveis com SVMS.

5.2 Trabalhos Futuros

Futuramente é interessante aplicar o algoritmo FLEXRANK a problemas reais, na qual o conjunto de dados sofra modificações constantemente (apren-

dizado *online*), como o caso da rotulação de páginas web. A criação de uma ferramente que filtre dados online e realize o acesso ou bloqueio de conteúdo (*proxy*) seria uma aplicação interessante para avaliar o algoritmo FLEXRANK.

Realizar estudos mais aprofundados sobre o *espaço de cobertura* e obter novas propriedades para seleção de atributos, são metas possíveis para um trabalho futuro. A investigação de novos métodos na seleção de atributos, pode otimizar ainda mais os algoritmos abordados neste trabalho: LEXRANK e FLEXRANK.

Outras melhorias que devem ser consideradas em um próximo trabalho é a adequação de LEXRANK e FLEXRANK a bases não binárias e multiclasse. Assim, é possível abordar os algoritmos citados neste trabalho a uma série de outros problemas sem ter que realizar adaptações primárias para análise do conjunto de dados que não sejam necessariamente binários.

APÊNDICE

A

Conjunto de Dados UCI e Textuais

Tabela A.2: Seleção de Atributos

Conjunto de Dados UCI		
	Conjunto de Dados	**Selecionados**
1	anneal[3]	surface-quality=? surface-quality=D hardness=(42.5-51) surface-quality=F hardness=(76.5-inf) family=TN
2	breast-cancer	node-caps deg-malig=1 deg-malig=3 breast tumor-size=10-14 tumor-size=30-34 tumor-size=35-39 breast-quad=central inv-nodes=0-2 inv-nodes=3-5 inv-nodes=6-8 inv-nodes=9-11
3	breast-w	Marginal_Adhesion=(9.1-inf) Clump_Thickness=(9.1-inf) Cell_Size_Uniformity=(-inf-1.9) Normal_Nucleoli=(9.1-inf) Bare_Nuclei=(-inf-1.9) Cell_Size_Uniformity=(9.1-inf)

		Cell_Shape_Uniformity=(-inf-1.9) Bare_Nuclei=(9.1-inf) Cell_Shape_Uniformity=(9.1-inf)
4	car[unacc]	maint=vhigh persons=2 persons=4 persons=more safety=low safety=high
5	credit-a	A11=(6.7-13.4) A11=(13.4-20.1) A6=x A15=(-inf-10000) A9 A10 A11=(-inf-6.7)
6	credit-g	checking_status=<0 checking_status=0<=X<200 checking_status=no checking duration=(-inf-10.8) savings_status=<100 duration=(44.8-51.6) savings_status=>=1000 savings_status=no known savings age=(-inf-24.6) credit_history=no credits/all paid credit_history=all paid credit_history=critical/other existing credit other_parties=guarantor purpose=used car purpose=radio/tv age=(47-52.6) credit_amount=(11154.4-12971.8) foreign_worker savings_status=500<=X<1000 credit_amount=(2067.4-3884.8)
7	dermatology[1]	scalp_involvement=0 clubbing_of_the_rete_ridges=0 clubbing_of_the_rete_ridges=2 elongation_of_the_rete_ridges=0 thinning_of_the_suprapapillary_epidermis=0 spongiosis=0 thinning_of_the_suprapapillary_epidermis=2 munro_microabcess=0 munro_microabcess=1
8	diabetes	preg=(-inf-1.7) preg=(6.8-8.5) preg=(10.2-11.9) age=(-inf-27)

		plas=(139.3-159.2)
		plas=(79.6-99.5)
		age=(51-57)
		plas=(59.7-79.6)
		insu=(507.6-592.2)
		insu=(761.4-inf)
		plas=(159.2-179.1)
		plas=(179.1-inf)
		mass=(13.42-20.13)
		mass=(20.13-26.84)
		mass=(46.97-53.68)
		pedi=(-inf-0.3122)
		age=(45-51)
		pedi=(2.1858-inf)
		skin=(9.9-19.8)
9	*glass	RI=(-inf-1.513428)
		Fe=(-inf-0.051)
		Al=(1.895-2.216)
		Al=(2.537-2.858)
		Ba=(-inf-0.315)
		Ba=(0.315-0.63)
		Ba=(1.575-1.89)
		Na=(14.055-14.72)
		Na=(14.72-15.385)
		Mg=(-inf-0.449)
		Mg=(1.347-1.796)
		Mg=(3.143-3.592)
		Mg=(3.592-4.041)
		Al=(0.611-0.932)
10	haberman	Age_of_patient_at_time_of_operation=(-inf-35.3)
		Age_of_patient_at_time_of_operation=(35.3-40.6)
		Age_of_patient_at_time_of_operation=(40.6-45.9)
		Patients_year_of_operation=59
		Patients_year_of_operation=60
		Patients_year_of_operation=61
		Patients_year_of_operation=65
		Patients_year_of_operation=67
		Patients_year_of_operation=69
		Number_of_positive_axillary_nodes_detected=(-inf-5.2)
		Number_of_positive_axillary_nodes_detected=(5.2-10.4)
		Number_of_positive_axillary_nodes_detected=(10.4-15.6)
11	*hayes-roth[1]	marital_status=(1.9-2.2)
		marital_status=(3.7-inf)
		hobby=(2.8-inf)
		age=(-inf-1.3)
		age=(3.7-inf)
		educational_level=(-inf-1.3)
		educational_level=(1.9-2.2)
		educational_level=(3.7-inf)

		marital_status=(-inf-1.3)
12	heart-statlog	thal=(6.6-inf) slope=(-inf-1.2) maximum_heart_rate_achieved=(97.2-110.3) maximum_heart_rate_achieved=(123.4-136.5) maximum_heart_rate_achieved=(136.5-149.6) number_of_major_vessels=(-inf-0.3) exercise_induced_angina=(-inf-0.1) chest=(2.8-3.1) sex=(0.9-inf) chest=(1.9-2.2) number_of_major_vessels=(2.7-inf) thal=(-inf-3.4) exercise_induced_angina=(0.9-inf) oldpeak=(-inf-0.62) sex=(-inf-0.1) chest=(3.7-inf) oldpeak=(2.48-3.1) oldpeak=(3.1-3.72)
13	ionosphere	a01=(-inf-0.1) a18=(-inf–0.8) a05=(-0.2-0) a12=(-inf–0.8) a01=(0.9-inf) a16=(-inf–0.8) a03=(-0.2-0) a13=(-0.2-0) a06=(-inf–0.8) a07=(-0.2-0) a33=(-0.2-0) a08=(-inf–0.8)
14	kr-vs-kp	wkna8 rimmx bxqsq
15	letter[A]	y-bar=(9-10.5) y-bar=(10.5-12) x2ybr=(-inf-1.5) xegvy=(4.5-6) xegvy=(7.5-9) xegvy=(9-10.5) x2ybr=(9-10.5) x2ybr=(10.5-12) xybar=(12-13.5) xy2br=(3-4.5) x-bar=(1.5-3) y2bar=(6-7.5) y2bar=(7.5-9) y2bar=(9-10.5) x-bar=(3-4.5)

		y-bar=(1.5-3)
		width=(-inf-1.5)
16	liver-disorders	mcv=(84-87.8)
		mcv=(87.8-91.6)
		mcv=(91.6-95.4)
		mcv=(99.2-inf)
		alkphos=(34.5-46)
		alkphos=(46-57.5)
		alkphos=(57.5-69)
		alkphos=(92-103.5)
		alkphos=(103.5-115)
		sgpt=(-inf-19.1)
		sgpt=(19.1-34.2)
		sgpt=(34.2-49.3)
		sgpt=(49.3-64.4)
		sgot=(-inf-12.7)
		sgot=(12.7-20.4)
		sgot=(20.4-28.1)
		sgot=(43.5-51.2)
		sgot=(66.6-74.3)
		gammagt=(-inf-34.2)
		gammagt=(34.2-63.4)
		gammagt=(151-180.2)
		drinks=(2-4)
17	*lymph[met]	changes_in_node=lac_margin
		changes_in_node=lac_central
		early_uptake_in
		no_of_nodes_in=(-inf-1.7)
		changes_in_stru=stripped
		lym_nodes_enlar=(3.7-inf)
		regeneration_of
		no_of_nodes_in=(4.5-5.2)
		no_of_nodes_in=(5.9-6.6)
		no_of_nodes_in=(6.6-7.3)
18	mfeat-pixel[one]	att129=6
		att98=6
		att154=0
		att113=0
		att128=6
		att113=6
		att112=6
		att128=0
		att139=0
		att114=6
		att138=0
19	*molec-bio	p-34=a
		p-34=g
		p-33=g
		p-36=a

		p-36=t
		p-35=t
20	monks-1	attr1=1
		attr1=2
		attr1=3
		attr2=1
		attr2=2
		attr3
		attr4=1
		attr4=2
		attr4=3
		attr5=1
		attr5=2
		attr5=3
		attr5=4
		attr6
21	monks-2	attr1=1
		attr1=3
		attr2=1
		attr2=2
		attr4=1
		attr4=2
		attr4=3
		attr5=1
		attr5=2
		attr5=3
		attr5=4
		attr6
22	monks-3	attr2=1
		attr2=2
		attr2=3
		attr3
		attr4=1
		attr4=2
		attr5=1
		attr5=2
		attr5=3
		attr5=4
		attr6
23	*postoper-pat[A]	L-CORE=mid
		L-SURF=high
		L-SURF=low
		L-SURF=mid
		L-BP=high
		L-BP=mid
		SURF-STBL
		CORE-STBL=unstable
		BP-STBL=mod-stable
		BP-STBL=stable

		BP-STBL=unstable
		COMFORT=05
		COMFORT=07
		COMFORT=10
24	sonar	attribute_36=(0.504-0.6032)
		attribute_43=(-inf-0.07733)
		attribute_44=(0.46572-0.54334)
		attribute_12=(0.50128-0.56952)
		attribute_11=(-inf-0.09943)
		attribute_26=(0.18289-0.27368)
25	splice[EI]	attribute_31=G
		attribute_31=C
		attribute_31=T
		attribute_31=A
		attribute_32=T
		attribute_32=C
		attribute_32=G
		attribute_32=A
		attribute_33=C
		attribute_33=T
		attribute_35=G
26	tic-tac-toe	bottom-right-square=o
		middle-middle-square=o
		middle-middle-square=x
	Conjunto de Dados Textual	
27	Hitech [Medical]	baur
		martinez
		packard
		wordperfect
		fieger
		cree
		hewlett
		inton
		merck
		brassi
		flicking
		manslaught
		trace
28	Review_Polarity	camil
		turkei
		finest
		hyam
		har
		outstand
		underwood
		vulner
		ludicr
		wonderfulli
		plod

		predat
		seamless
		wast
		ideolog
		worst
		idiot
		mcconaughei
		undevelop
		inept
29	CSTR [I.A]	word
		corpu
		dialogu
		speech
30	SyskillWebert [BioMedical]	Hospit
		med
		butt
		song
		goat
		medic
		medicin
		informat
		band
		molecular
		sheep
31	IrishEconomicSentiment [posite]	fuel
		stabilis
		dollar
		crude
		garda
		oil
		revis
		rose
		parent
		australia
32	Classic3 [cacm]	thick
		flat
		patient
		pressur
		book
		mach
		wing
		superson
		tunnel
		algorithm
		veloc
		wall
		laminar
		temperatur

		algol
33	LATimes [metro]	supervisori
		briefcas
		deutsche
		playpen
		found
		rebroadcast
		scorch
		coache
		councilmen
		sherlock

Tabela A.1: Seleção de Atributos Conjunto de Dados

Conjuntos de Dados UCI						
#	Dataset	CV	Ex	Atributos		
				Total	%	Mín
1	anneal[3]	10	898	145	3,4	5
2	breast-cancer	10	277	49	12,9	6,3
3	breast-w	10	683	91	6,2	5,6
4	car[unacc]	10	1728	22	22,7	5
5	credit-a	10	653	98	3,8	3,7
6	credit-g	10	1000	125	5,3	6,6
7	dermatology[1]	10	358	139	4,5	6,3
8	diabetes	10	768	81	10,7	8,7
9	*glass	5	214	91	7,9	7,2
10	haberman	10	306	33	13,9	4,6
11	*hayes-roth[1]	5	160	41	20,0	8,2
12	heart-statlog	10	270	131	5,1	6,7
13	ionosphere	10	351	332	1,6	5,2
14	kr-vs-kp	10	3196	41	7,3	3
15	letter[A]	10	20000	161	6,2	10
16	liver-disorders	10	345	61	17,7	10,8
17	*lymph[met]	5	148	66	6,4	4,2
18	mfeat-pixel[one]	10	2000	1649	0,4	6,2
19	*molec-bio	5	106	229	1,5	3,4
20	monks-1	10	556	16	39,4	6,3
21	monks-2	10	601	16	43,1	6,9
22	monks-3	10	554	16	49,4	7,9
23	*postoper-pat[A]	5	87	22	19,1	4,2
24	sonar	10	208	601	0,4	2,3
25	splice[EI]	10	267	288	3,4	9,9
26	tic-tac-toe	10	3190	28	7,1	2
	Média		**38924**	**4572**	**12,3**	**6,01**
Cojnunto de Dados Textuais						
27	Hitech [Medical]	10	2301	12941	0,027	3,5
28	Review_Polarity	10	2000	15697	0,017	2,7
29	CSTR [I.A]	10	299	1725	0,116	2
30	SyskillWebert [BioMedical]	10	345	5476	0,077	4,2
31	IrishEconomicSentiment [posite]	10	1660	8658	0,023	2
32	Classic3 [cacm]	10	7095	7748	0,075	5,8
33	LATimes [metro]	10	6279	6279	0,067	4,2
	Média		**19979**	**58524**	**0,057**	**3,49**

Referências Bibliográficas

Anderson, J. (1983). Lix and rix: Variations on a little-known readability index. *Journal of Reading*, 26(6):490–496. Citado na página 22.

Asuncion, A. e Newman, D. J. (2007). UCI machine learning repository. Citado nas páginas xiii, 20, 49, e 50.

Barnard, G. A. e Bayes, T. (1958). Studies in the history of probability and statistics: Ix. thomas bayes's essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3/4):293–315. Citado na página 13.

Basu, S., Banerjee, A., e Mooney, R. J. (2002). Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, páginas 27–34. Morgan Kaufmann Publishers Inc. Citado na página 11.

Bergstra, J. e Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305. Citado na página 51.

Bird, S., Klein, E., e Loper, E. (2009). *Natural language processing with Python*. "O'Reilly Media, Inc.". Citado nas páginas 45, 51, e 52.

Bland, J. M. e Altman, D. G. (2000). The odds ratio. *Bmj*, 320(7247):1468. Citado na página 16.

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, páginas 177–186. Springer. Citado na página 23.

Breiman, L., Friedman, J., Stone, C. J., e Olshen, R. A. (1984). *Classification and regression trees*. CRC press. Citado nas páginas 16 e 50.

Coleman, M. e Liau, T. (1975). A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283. Citado na página 22.

Cover, T. M. e Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27. Citado na página 11.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30. Citado na página 58.

Dicio (2015). Aprendizagem - dicionário online de português. http://www.dicio.com.br/aprendizagem/. (Acesso em 30 Mar 2015). Citado na página 6.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., et al. (1996). Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, páginas 82–88. Citado na página 21.

Ferri, C., Flach, P., e Hernández-Orallo, J. (2002). Learning decision trees using the area under the roc curve. In *ICML*, volume 2, páginas 139–146. Citado na página 16.

Flach, P. e Matsubara, E. (2007). On classification, ranking, and probability estimation. In de Raedt, L., Dietterich, T., Getoor, L., Kersting, K., e Muggleton, S. H., editors, *Probabilistic, Logical and Relational Learning - A Further Synthesis*, number 07161 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. Citado nas páginas 3, 11, 26, e 27.

Flach, P. A. (2003). The geometry of roc space: understanding machine learning metrics through roc isometrics. In *ICML*, páginas 194–201. Citado nas páginas xiii e 20.

Fürnkranz, J. e Flach, P. A. (2005). Roc ?n?rule learning?towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77. Citado na página 33.

Hosmer Jr, D. W. e Lemeshow, S. (2004). *Applied logistic regression*. John Wiley & Sons. Citado na página 23.

Huang, C., Davis, L., e Townshend, J. (2002). An assessment of support vector machines for land cover classification. *International Journal of remote sensing*, 23(4):725–749. Citado nas páginas xiii e 17.

Ian H. Witten, E. F. (2011). *Data mining: practical machine learning tools and techniques, Volume 31*. Elseviers, New York, NY, USA. Citado na página 6.

Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. Relatório técnico, Proceealings of 13th International Joint Conference on Artificial Intelligence. Citado na página 50.

Kearns, M. e Mansour, Y. (1996). On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, páginas 459–468. ACM. Citado na página 16.

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331. Citado na página 26.

Matsubara, E. T. (2004). *O algoritmo de aprendizado semi-supervisionado co-training e sua aplicão na rotulação de documentos*. Dissertação de Mestrado, ICMC-USP. Citado na página 10.

Matsubara, E. T. (2008). *Relações entre Ranking, Anáise ROC e Calibração em Aprendizado de Máquina*. Dissertação de Doutorado, ICMC-USP. Citado nas páginas 3, 4, 13, 14, 25, 26, 27, 28, 49, e 53.

McCallum, A., Nigam, K., et al. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, páginas 41–48. Citeseer. Citado na página 13.

Melgani, F. e Bruzzone, L. (2004). Classification of hyperspectral remote sensing images with support vector machines. *Geoscience and Remote Sensing, IEEE Transactions on*, 42(8):1778–1790. Citado nas páginas xiii e 17.

Michaelis, D. (2015). Dicionário português online: Moderno dicionário da língua portuguesa - michaelis - uol. http://michaelis.uol.com.br/moderno/portugues/index.php?lingua=portugues-portugues&palavra=aprendizagem. (Acesso em 30 Mar 2015). Citado na página 6.

Miltsakaki, E. e Troutt, A. (2008). Real-time web text classification and analysis of reading difficulty. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, EANL '08, páginas 89–97, Stroudsburg, PA, USA. Association for Computational Linguistics. Citado na página 22.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition. Citado nas páginas xiii, xv, 6, 7, 12, 13, 15, e 16.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. Citado nas páginas 45, 49, 51, e 53.

Rezende, S. O., Pugliesi, J., Melanda, E., e Paula, M. d. (2003). Mineração de dados. *Sistemas inteligentes: fundamentos e aplicações*, 1:307–335. Citado nas páginas xiii e 21.

Rigo, F. V. (2013). *Moderação de Sítios Utilizando Aprendizado Semissupervisionado Ativo*. Dissertação de Mestrado, UFMS. Citado nas páginas xiii, 3, 23, e 24.

Rossi, R. G., Marcacini, R. M., e Rezende, S. O. (2013). Benchmarking text collections for classification and clustering tasks. *Institute of Mathematics and Computer Sciences, University of Sao Paulo*. Citado na página 49.

Russell, S. J. e Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition. Citado na página 26.

Schmitt, M. e Martignon, L. (2006). On the complexity of learning lexicographic strategies. *The Journal of Machine Learning Research*, 7:55–83. Citado na página 3.

Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55. Citado nas páginas xiii, 15, e 16.

Skinner, B. F. (1972). *Tecnologia do ensino*. Herder, Ed. da universidade São Paulo. Citado na página 6.

Van Der Walt, S., Colbert, S. C., e Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30. Citado nas páginas 45 e 50.

Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience. Citado na página 16.

Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al. (2001). Constrained k-means clustering with background knowledge. In *ICML*, volume 1, páginas 577–584. Citado na página 11.

Webb, A. R. (2003). *Statistical pattern recognition*. John Wiley & Sons. Citado na página 11.

Witten, I. H. e Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann. Citado na página 50.

Yaman, F., Walsh, T. J., Littman, M. L., et al. (2011). Learning lexicographic preference models. In *Preference learning*, páginas 251–272. Springer. Citado na página 1.

Zhu, X. (2005). Semi-supervised learning literature survey. Relatório Técnico 1530, Computer Sciences, University of Wisconsin-Madison. Citado nas páginas 5 e 10.