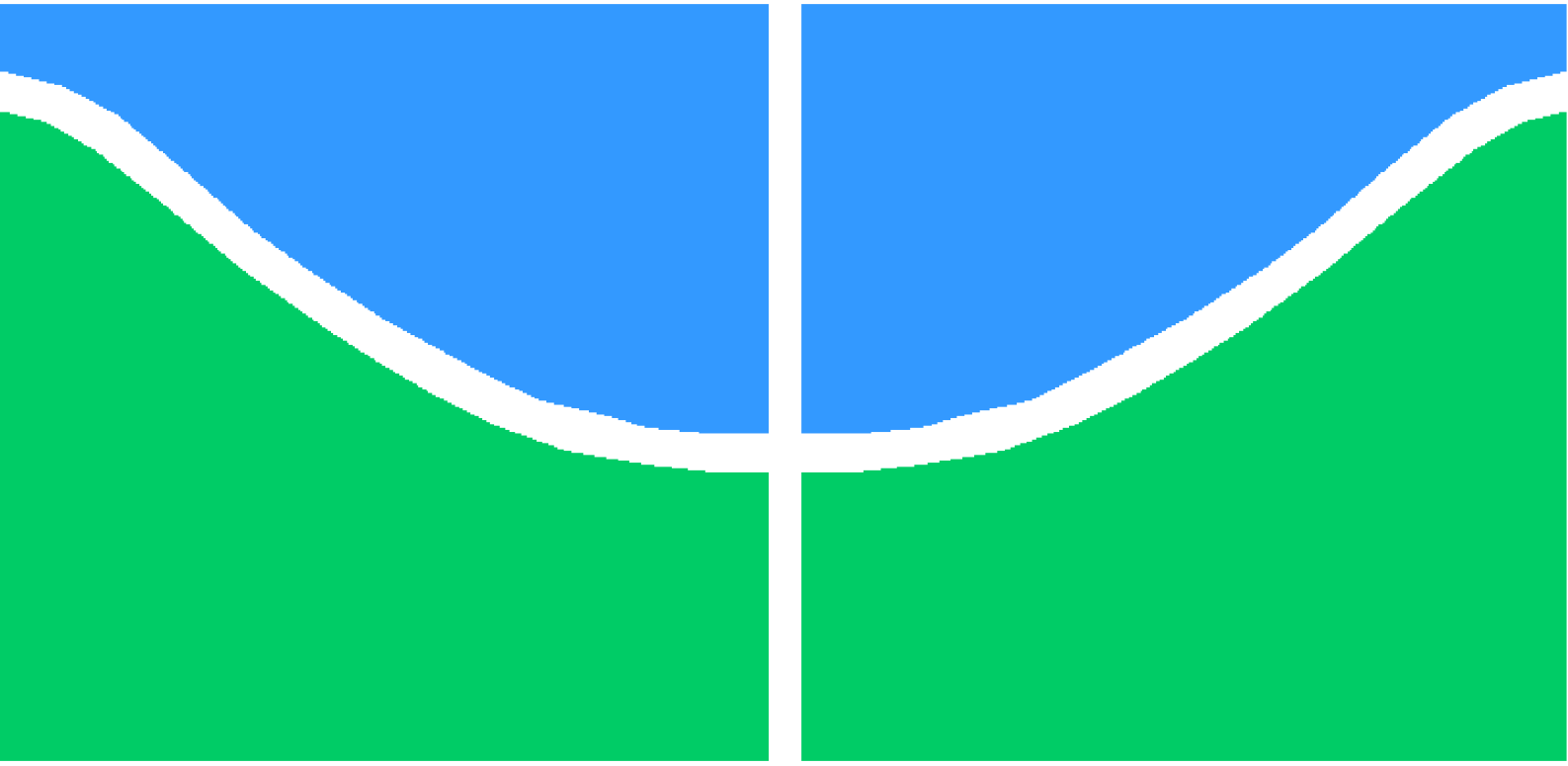

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Serviço de Seleção de Característcas para problemas de Áprendizado de Máquina

Autor: Vinícius Franco da Cunha Arantes
Orientador: Professor Dr. Fabricio Ataides Braz

Brasília, DF
2015

Vinícius Franco da Cunha Arantes

Serviço de Seleção de Característcas para problemas de Aprendizado de Máquina

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Dr. Fabricio Ataides Braz

Brasília, DF

2015

Vinícius Franco da Cunha Arantes
Serviço de Seleção de Característcas para problemas de Aprendizado de Máquina/ Vinícius Franco da Cunha Arantes. – Brasília, DF, 2015-
52 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Dr. Fabricio Ataides Braz

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Aprendizado de Máquina. 2. Seleção de Características. I. Professor Dr. Fabricio Ataides Braz. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Serviço de Seleção de Característcas para problemas de Aprendizado de Máquina

CDU 02:141:005.6

Vinícius Franco da Cunha Arantes

Serviço de Seleção de Característcas para problemas de Áprendizado de Máquina

Monografia submetida ao curso de graduação em (Engenharia de Software) da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em (Engenharia de Software).

Trabalho aprovado. Brasília, DF, 27 de novembro de 2015:

Professor Dr. Fabricio Ataides Braz
Orientador

Professor Dr. Luiz Augusto Laranjeira
Convidado 1

Professor Dr. Nilton Corria da Silva
Convidado 2

Brasília, DF
2015

Resumo

A área de Aprendizado de Máquina tem crescido muito e ganhado muitos adpetos, o que faz com que cada vez mais seja necessário aprender a lidar e tratar o gigante volume de dados gerados diariamente por diversos usuários espalhados pelo mundo. Tendo em vista toda essa massa de dados, é preciso saber selecionar quais deles são realmente relevantes. Para poder melhor selecionar esses dados existe o Processo de Seleção de Características. Este trabalho apresenta um estudo sobre como é esse processo e como esse processo auxilia os problemas de Aprendizado de Máquina, além de mostrar resultados que incentivam o uso dessas técnicas. Finalmente, esse trabalho propõe a implementação de um serviço para selecionar características em meio a uma base utilizando modelos de seleção de características.

Palavras-chaves: Seleção de características, Aprendizado de Máquina, Modelos de Seleção de Características.

Abstract

The Machine Learning area has been growing a lot and aquiring many adepts, which makes increasingly more necessary to learn how to deal and treat the giant volume of data daily generated by diverse users around the world. Having all this mass of data in mind, it's necessary to know how to select the really relevant ones. To be able to better select this data there is the Feature Selection Process. This work shows a study about what is this process and how this process helps the Machine Learning problems, it also shows results that incetivate the use of these techniques. Finally, this work proposes the implementation of a service to select features in a datase using the feature selection models.

Key-words: Feature Selection, Machine Learning, Feature Selection Model.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Motivação

Na sociedade atual, cada vez mais milhares de dados são gerados diariamente e há uma dificuldade em se aproveitar ao máximo o conteúdo desses dados, uma vez que é impossível analisar-los individualmente para poder identificar o seu real valor. Segundo o Business Intelligence Observatory (2013), cerca de 90% dos dados de hoje foram gerados nos últimos dois anos, tendo como fontes documentos de textos, e-mails, streaming de vídeos e áudio, transações comerciais, telecomunicações e diversas outras. Estimas-se que valor investido para trabalhar esses dados tende a superar a faixa de 27 bilhões em 2015.

A fim de analisar esses dados, técnicas foram desenvolvidas e ferramentas foram criadas, tais meios visam facilitar a análise dessa grande massa de dados e extraindo valor dela. Um questionário levantado pela Bloomberg Businessweek (2011) mostra que 97% das companhias com receitas acima de 100 milhões de dólares usam algum tipo de análise de negócio, e que essa análise se baseia em grandes volumes de dados e seu valor agregado à companhia.

Para realizar essas análises, várias empresas tem feito investimentos na área de Aprendizado de Máquina, em inglês *Machine Learning (ML)*. A área de Aprendizado de Máquina é considerada um ramo da Inteligência Artificial, sendo uma área especializada no estudo e construção de sistemas que sejam capazes de aprender de forma automatizada a partir de dados (BRINK; RICHARDS, 2014). Dentro da área de Aprendizado de Máquina existem vários passos para se construir um modelo, e um desses passos é o Processo de Seleção de Características em que um dado potencialmente útil para realizar predições (MITCHELL, 1997).

Problemas de Aprendizado de Máquina costumam gerar muitos dados que aparentam ser úteis, mas que nem sempre definem com exatidão o problema, ou que ajudem a alcançar o objetivo traçado na construção do modelo. De acordo com Guyon (2003), muitos problemas de Aprendizado de Máquina chegam a variar entre cinco mil a cinquenta mil características que, inicialmente, são consideradas importantes ou relevantes ao problema. Esse grande número de características que são levantadas inicialmente demandam muito tempo para serem processadas, levando a necessidade de reduzir esse número para auxiliar na sua análise. O elevado número ainda acarreta no possível aparecimento de ruídos durante a análise dos dados. A Seleção de Características visa auxiliar esse problema motivado pelas técnicas para lidar com esses dados. Com essa abordagem é possível escolher os melhores subconjuntos de características que conseguem alcançar os objetivos

propostos pelo modelo.

1.2 Objetivos

Esse trabalho teve como objetivo principal a implementação de um serviço que fosse capaz de analisar um conjunto de características e extrair o melhor subconjunto possível analisando-o em meio a três modelos de seleção de características.

1.3 Objetivos Específicos

Para poder alcançar o objetivo geral, esse trabalho foi divido em etapas. Os objetivos específicos do trabalho são:

1. Especificar o processo de seleção de características
2. Pesquisar modelos de seleção de características a serem utilizados
3. Implementar os modelos de seleção de características
4. Realizar testes nos modelos
5. Específicar arquitetura do serviço
6. Construir protótipo do serviço
7. Realizar testes no protótipo do serviço
8. Coletar e relatar resultados

1.4 Justificativa

Como já foi pontuado, a sociedade necessita de métodos e ferramentas para conseguir processar a avalanche de informação produzida por segundo. Nesse sentido, o tema deste trabalho se mostra muito pertinente, uma vez que o processo de seleção de características pode viabilizar processamento de massas de dados extremamente extensas, evidenciando as variáveis mais relevantes para o problema alvo do estudo, reduzindo assim o esforço computacional.

Tão importante como a concepção de novos modelos ou a criação de ferramentas para o processamento de dados é a simplificação do uso dessas tecnologias por profissionais, cuja densidade em programação, integração, modelos estatísticos/matemáticos seja limitada. Essa facilidade é o que motiva a proposta de desenvolvimento de um serviço para o cientista de dados poder realizar o processo de seleção de características, sem muita dificuldade

1.5 Método de Pesquisa

1.5.1 Etapa 1: Especificar o processo de seleção de características

A primeira etapa deste trabalho consistiu em elucidar e específicar como funciona o processo para seleção de características. Para tal, feito foi necessário fazer uma pesquisa bibliográfica visando demonstrar o fluxo do processo de seleção e suas principais etapas.

1.5.2 Etapa 2: Pesquisar modelos de seleção de características a serem utilizados

Essa etapa consistiu em pesquisar os modelos existentes e como são compostos, levando em consideração o processo estudado na etapa anterior e se esses modelos estão consistentes ao processo de seleção de características em si. Após realizada as pesquisas foram escolhidos os modelos que são utilizados no serviço implementado.

1.5.3 Etapa 3: Implementar os modelos de seleção de características

Para cada um dos modelos selecionados, foi realizado a sua implementação através da biblioteca Weka (WITTEN, 2005). Essa escolha foi feita visando uma melhor compatibilidade entre os modelos.

1.5.4 Etapa 4: Realizar testes nos modelos

Os modelos implementados foram testados utilizando duas base de dados, a Madelon (MADELON, 2003) e a Titanic. Essas bases de dados foram pré-processadas para preencher as lacunas na base de dados. Os resultados foram obtidos e documentados.

1.5.5 Etapa 5: Específicar arquitetura do serviço

Nessa etapa foi feito a especificação de como o serviço proposto por esse trabalho funcionaria, descrevendo cada uma de suas fases e como se dá o seu fluxo.

1.5.6 Etapa 6: Construir protótipo do serviço

Após feito a especificação da arquitetura, foi feito um protótipo do serviço, com todas as funcionalidades levantadas nas etapas anteriores. Esse protótipo recebe uma base de cados e faz o processo de seleção utilizando o modelo selecionado, extraindo o melhor subconjunto de caraterísticas e mostrando a acurácia alcançada por esse subconjunto.

1.5.7 Etapa 7: Realizar testes no protótipo do serviço

Com a implementação do protótipo do serviço, foram feitos testes para garantir que as funcionalidades implementadas estão funcionando de acordo com o projeto. Foram utilizadas as mesmas bases e os resultados obtidos são iguais aos obtidos ao testar os modelos fora do serviço.

2 Desenvolvimento

Esse trabalho propõe o desenvolvimento de um serviço que seja capaz de receber uma base de dados, submete-la aos modelos propostos, e então entregar ao usuário uma análise de quais características foram escolhidas por cada um dos modelos e qual o ganho obtido em questão de acurácia de predição.

No capítulo introdutório deste trabalho foram estabelecidas oito etapas a serem realizadas. Logo abaixo temos um resumo do que foi feito em cada uma dessas etapas e qual o respectivo resultado.

2.1 Etapa 1: Especificar o processo de seleção de características

Essa etapa trouxe a fundamentação teórica necessária ao tema explorado por esse trabalho. Tal investigação permitiu esclarecimentos essenciais sobre práticas e modelos que subsidiaram a construção do serviço. Por se tratar de um conteúdo de suma importância para esse trabalho, e um tanto quanto extenso, foi decidido criar um capítulo para que fosse feito todo o levantamento bibliográfico referente ao processo de seleção de características (Capítulo 3). Neste capítulo foi descrito as diversas formas de combinação de técnicas a fim de demonstrar os vários modelos que se é possível construir.

2.1.1 Etapa 2: Pesquisar modelos de seleção de características a serem utilizados

Após realizada a pesquisa para o entendimento do processo de seleção, foi necessário realizar uma pesquisa para escolher quais modelos seriam utilizados, uma vez que existem vários, e que a sua escolha é de fundamental importância na execução desse trabalho.

Os modelos foram escolhidos após pesquisas e análises em relação a dificuldade de implementação, uma vez que a complexidade pode ser alta, além do tempo gasto para realizar os testes da etapa seguinte, que poderia demandar muito tempo. Os modelos selecionados foram: Relief-F (DASH, 1997), DTM (DASH, 1997), e LFS (GUTLEIN, 2009). Os modelos estão melhor descritos no Capítulo 4.

2.1.2 Etapa 3: Implementar os modelos de seleção de características

Para realizar a implementação foram realizadas pesquisas de ferramentas de mineração de dados. Durante as pesquisas foram encontradas ferramentas capazes de auxiliar

o projeto, sendo as que mais se destacam: Scikit-learn (COURNAPEAU, 2007), Weka (WITTEN, 2005) e Apache Spark (FOUNDATION, 2014). Dentre as ferramentas, a que mais se destacou a esse trabalho foi a Weka, pelo fato dela ter alguns dos métodos já implementados, e pelo fato dela ser uma biblioteca código aberto, o que facilitaria caso alguma mudança fosse necessária. A biblioteca também possui uma grande popularidade e comunidade, o que facilita na busca por auxilio ao seu uso.

A partir da seleção de uma biblioteca, foi então implementado cada um dos modelos escolhidos para serem utilizados no serviço.

2.1.3 Etapa 4: Realizar testes nos modelos

Conforme será mostrado no Capitulo 5, os modelos foram testados utilizando duas bases de dados: a base dos passageiros do Titanic e a base MADELON, utilizada em competições de aprendizado de máquina.

2.1.4 Etapa 5: Específicar arquitetura do serviço

Durante essa etapa foi pensado como deveria ser o funcionamento da aplicação, uma vez que a proposta desse trabalho era criar um serviço que pudesse submeter uma base de dados e extrair as melhores características de acordo com o modelo selecionado. Como os modelos já haviam sido implementados utilizando a biblioteca Weka, foi então necessário encontrar um conjunto de ferramentas que se adequasse à biblioteca, e que pudesse desenvolver uma aplicação web. Levando em consideração essas premissas foram escolhidos as ferramentas Ruby on Rails (HANSSON, 2008) e a linguagem JRuby (SIEGER, 2010). A combinação da linguagem JRuby com a ferramenta Ruby on Rails, permitiria utilizar o poder e agilidade de desenvolvimento Web da ferramenta Ruby on Rails, junto ao uso de bibliotecas implementadas em Java através do JRuby, o que se encaixaria perfeitamente no contexto desse trabalho.

Ao tentar realizar a comunicação das bibliotecas, alguns problemas foram encontrados, como por exemplo as divergências entre a estrutura da documentação do Weka com a sua estrutura de classes, bibliotecas Ruby que não eram compatíveis com JRuby e tratamento de exceções da biblioteca Weka. Esse problemas foram resolvidos ao longo do dessa etapa, e consolidados na Etapa 6, onde foi implementado o serviço.

2.1.5 Etapa 6: Construir protótipo do serviço

Após a devida integração entre as ferramentas e os componentes do sistema, foram criadas as entidades da aplicação, e a partir delas começou-se a modelar como seria o serviço. Seguindo o que foi proposto no diagrama da arquitetura e as fases que foram propostas, o sistema começou a tomar forma. Um ponto crucial da construção foi extrair

os modelos de seleção da ferramenta Weka, uma vez que acessar suas classes e montar os seus modelos foi trabalhoso, porém fez com que o serviço fizesse exatamente o que foi proposto. Tendo como resultado desta etapa um serviço que receberia uma base de dados, faria seu pre processamento, a executaria, e em seguida forneceria os resultados dessa execução.

2.1.6 Etapa 7: Realizar testes no protótipo do serviço

Para que fosse possível aferir a funcionalidade do serviço, as mesmas bases utilizadas na etapa 4, foram utilizadas para que fosse possível verificar se o desempenho seria mantido. Além desse teste, foram testados também bases de dados com estruturas diferentes da proposta na fase de Pré Processamento (6.3.1) e o sistema deveria recusá-las. Em ambos os casos o resultado foi positivo. O desempenho do sistema foi mantido em relação ao que foi feito anteriormente, e o sistema recusava as bases sem os pré requisitos de estrutura, mais detalhes podem ser observados no Capítulo **??**

3 Seleção de Características

3.1 Aprendizado de Máquina (*Machine Learning*)

A área de Aprendizado de Máquina é considerada um ramo da área de Inteligência Artificial, sendo uma área especializada no estudo e construção de sistemas que sejam capazes de aprender de forma automatizada a partir de dados (BRINK; RICHARDS, 2014). Desde que os computadores foram inventados, pesquisadores tentam procurar uma maneira de fazer com que eles possam aprender e melhorar sua performance através da experiência. Se fosse possível fazer com que os computadores pudessem aprender, diagnósticos médicos poderiam ser feitos, matérias de jornais que seriam entregues às pessoas de acordo com seus gostos, uma nova maneira de lidar com problemas e situações poderia ser criada (MITCHELL, 1997).

Por definição, Mitchell (1997) diz que um dado programa de computador é capaz de aprender com experiência E, com respeito a um grupo de tarefas T e índice de performance P, se a performance medida por P em T aumenta com E. Ou seja, se conforme o tempo passa, a performance aumenta, ele estará aprendendo.

Existem diversos exemplos de aplicações que aprendem com o tempo e possuem alto grau de acurácia quanto à predição dos possíveis resultados, como por exemplo os sistemas de detecção de spams dos provedores de Email, onde é analisado o texto presente nos emails, o seu email de origem, a frequência desses emails, entre outros dados. No caso de um detector de spam, entende-se pela acurácia do modelo a performance em porcento com que ele consegue classificar corretamente um email, sendo ele spam ou não. O detector de spam utilizaria de uma base de dados de emails para treinar o modelo, sendo essa composto por emails categorizados entre spam e não spam, essa seria a base de dados de treinamento onde ele irá aprender a reconhecer os padrões existentes nos emails, para posteriormente poder predizer e classificar os emails. Temos então que um modelo de Aprendizado de Máquia é composto por um algoritmo de aprendizado de máquina, uma base de dados para realizar o treinamento, e os dados de entrada para serem classficados (MITCHELL, 1997)

3.2 Processo de Seleção

Seleção de Características é um processo que seleciona um subconjunto de um conjunto original de características, a optimicidade do subconjunto é medida por um critéro de avaliação. (YU, 2005). As características são fundamentais para a solução de

um dado problema de Aprendizado de Máquina, e quão melhor for essa seleção, melhor será o resultado obtido assim como o tempo necessário para resolve-lo, em termos de acertividade e consumo de recurso computacional.

As características podem ser de dois tipos: numéricas ou categóricas. As numéricas são representadas por valores, como o próprio nome surgere, já a categórica visa dividir as características em diferentes categorias, não importando a sua ordem.

O processo de seleção de características geralmente segue quatro passos conforme ilustrado pela Figura 1, sendo eles: geração de subconjuntos (*subset generation*), avaliação do subconjunto (*subset evaluation*), critério de parada (*stopping criterion*), e validação de resultados (*result validation*). De acordo com Molina (2002) um subconjunto ideal é aquele que segue uma das seguintes abordagens: a) um subconjuto que possua um tamanho específico que otimize a forma de avaliação. b) um subconjunto de tamanho menor que satisfaça alguma restrição na forma de avaliação. c) o subconjunto com o melhor desempenho levando em consideração sua dimensão e o valor de sua medida de avaliação.

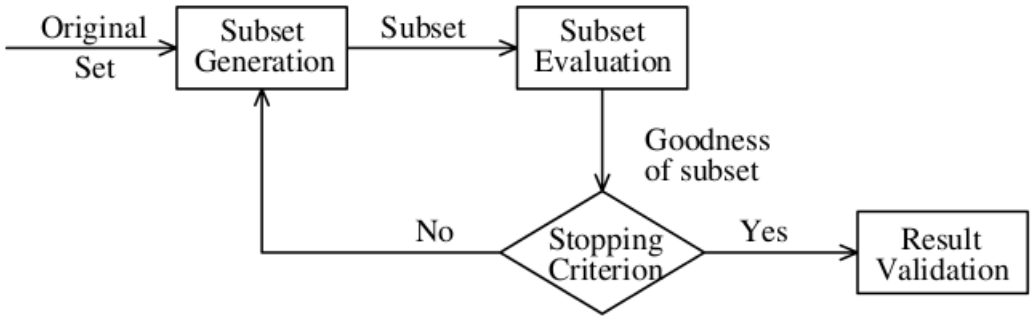

Figura 1 – Fluxograma de seleção de características. (YU, 2005)

3.3 Geração de Subconjuntos

O processo de seleção de características pode ser visto como um problema de busca em que cada um dos estados da busca representa um dos subconjuntos de características (LIU; MOTODA., 1998). Imagine um conjunto com três características. Se forem colocadas em um vetor, se posicionariam da seguinte maneira (A1, A2, A3). Tendo em vista que o valor 1 representa a presença da características no subconjunto a ser gerado, temos que em (1, 0, 0) apenas a característica A1 está presente nesse subconjunto. Variando todas as possibilidades partindo de um conjunto com todas as características e chegando a um conjunto com nenhuma. Tal processo está ilustrado na Figura 2.

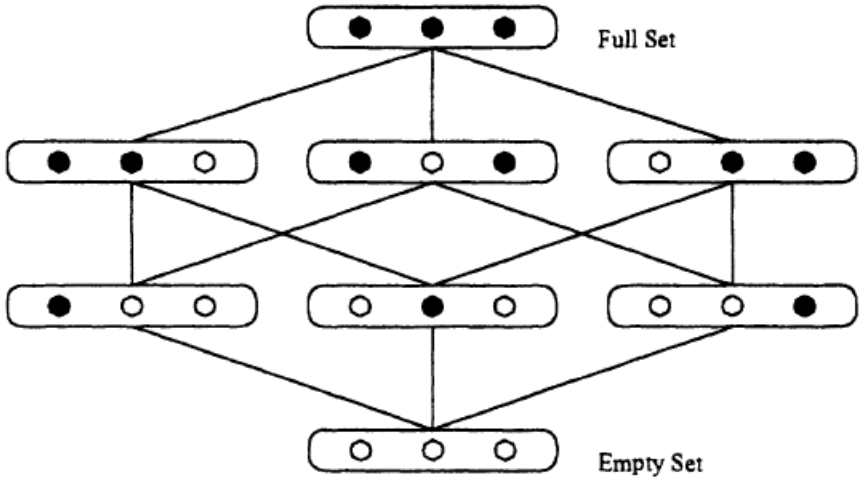

Figura 2 – Busca por remoção de características (SBG). (LIU; MOTODA., 1998)

Um dos passos para realizar o processo de seleção de características é escolher a estratégia de busca. A estratégia de busca é a forma como será feito a busca dos subconjuntos a serem avaliados. Existem três maneiras de realizar a busca: busca completa, busca sequencial e busca randomica.

Busca completa - baseia-se em analisar todos os subconjuntos possíveis. Essa busca pode ser exaustiva ou não-exaustiva. A exaustiva buscará analisar todas as combinações, sendo assim, ela se torna um problema da ordem $O(2^N)$. A não-exaustiva busca analisar o máximo de estados possiveis dentro de um determinado limite, buscando evitar comparações desnecessárias para a geração dos subconjuntos (YU, 2005). A busca completa sempre achará o melhor subconjunto, uma vez que ela varre todas as possibilidades, mesmo utilizando técnicas não-exaustivas (DASH, 1997).

Busca sequencial - baseia-se em remover ou adicionar características conforme é executado. Esse tipo de busca pode acarretar em perda de alguns subconjuntos ótimos, porém é mais rapido e exige menos recursos computacionais, sendo um problema da ordem de $O(N^2)$. (DASH, 1997) Existem três formas de realizar esse tipo busca: *sequential forward generation (SFG)*, *sequential backward generation (SBG)*, *bi-drectional generation*. A primeira consiste em começar com um conjunto vazio e ir adicionando características a ele. Já a *backward* consiste em usar um conjunto completo e ir removendo características sucessivamente. A terceira forma consiste em adicionar e remover simultaneamente, ela é composta por duas buscas simultaneas que geralmente partem do meio do conjunto inicial, com o objetivo de encontrar o melhor subconjunto possível. (YU, 2005).

Busca randomica - esta segue duas frentes. A primeira utiliza busca sequencial e adiciona as características de forma randomica. A outra frente gera o próximo subconjunto de maneira completamente randomica, ou seja, gera todo o subconjunto de características aleatórias. Tende a ser um problema da ordem de $O(N^2)$ (YU, 2005).

3.4 Avaliação do Subconjunto

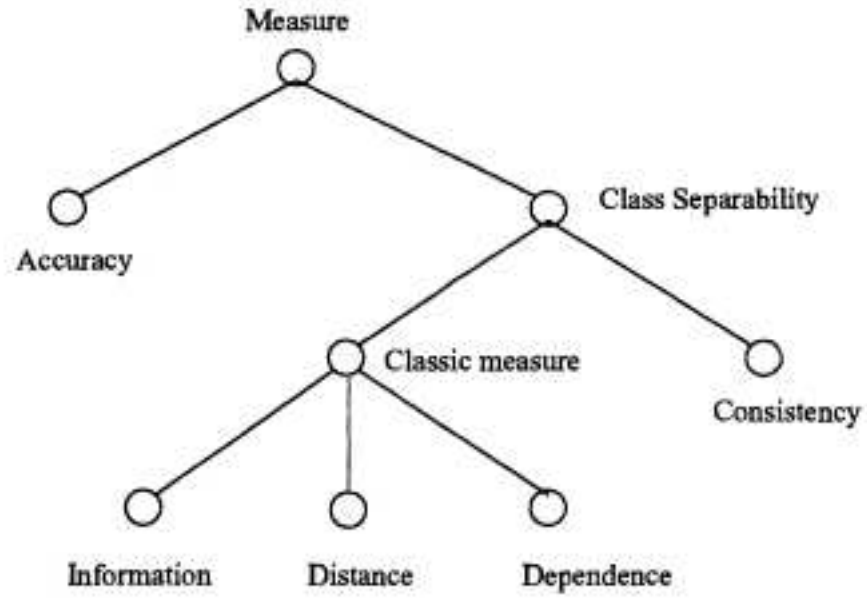

Figura 3 – Hierarquia de Medidas. (LIU; MOTODA., 1998)

Para poder dizer quão bom é um subconjunto é preciso definir um critério. Esses critérios são descritos de acordo com a hierarquia aprsentada na figura 3. As medidas para avaliar uma caracaterística são: medidas de informação (*information measures*), medidas de distancia (*distance measures*), medidas de dependencia (*dependancy measures*), medidas de consistência (*consistency measures*) e medidas de acurácia (acuracy measures). (YU, 2005; LIU; MOTODA., 1998) As medidas podem ser divididas em três grupos distintos: a) grupo de medidas classicas, que englobam as medidas de informação, medidas de distancia e medidas de dependencia b) grupo de medidas de consistência, que engloba a própria medida em si c) grupo de medidas de acurácia, que também engloba apenas a medida em si (LIU; MOTODA., 1998).

3.4.1 Medidas de informação

A medida de informação determina a informação obtida por uma dada característica (YU, 2005). Isso implica em dizer que uma característica A1 oferece mais informações do que uma característica A2, se dado uma função de incerteza F(x), A1 reduza o grau de incerteza mais do que A2.

3.4.2 Medidas de distancia

Também conhecido como medida de separabilidade, medida de divergencia ou medida discriminativa, essa medida é feita através do calculo da distancia entre duas

classes, ou seja, calcular o quanto uma característica consegue separar duas classes de maneira distinta. Sendo F(X) o calculo da distancia, uma característica A1 separar melhor duas classes do que A2 se F(A1) > F(A2). (LIU; MOTODA., 1998)

3.4.3 Medidas de dependencia

Conhecida também como medida de correlação ou de similaridade, essa medida visa medir a habilidade de poder predizer o valor de um variável X sabendo o valor de uma variável Y, sendo assim, dado uma função para medir a dependencia F(X, Y), sendo A1 e A2 características distintas, e C a classe, a característica A1 é preferida à característica A2, se F(A1, C) > F(A2, C), ou seja, a característica A1 está melhor associada com a classe C do que A2. (LIU; MOTODA., 1998; YU, 2005)

3.4.4 Medidas de consistência

As medidas anteriores tem o propósito de achar as melhores características que consigam destinguir uma classe de outra. Porém elas não conseguem distinguir características que sejam igualmente boas, ou seja, que apontem para um mesmo resultado e gere redundancia nos subconjuntos. A medida de consistência procura achar um menor número de características que seja tão consistente em seprar as classes quanto ao conjunto inicial. Ou seja, dado uma função de consistência F(X, Y) em relação a classe C, o subconjunto S1 será consistente se F(S1, C) = F(Si, C), sendo S o conjunto inicial (YU, 2005)

3.4.5 Medidas de acurácia

Essa já é uma medida que é utilizada dentro do classificador, ou seja, é a medida de performance dele. Para um dado subconjunto de características, quanto mais alta for a acurácia do classificador, melhor será esse subconjunto (LIU; MOTODA., 1998).

3.5 Critério de Parada

O critério de parada serve para que, dado um conjunto de características e um algoritmo de seleção de características, ao alcançar um determinado estado de seleção ele diga se o subconjunto gerado está bom ou se ainda precisa ser refinado (DASH, 1997). Critérios de parada comumente usados são: a) a busca é finalizada; b) algum valor pré-determinado é alcançado, como por exemplo um número minimo ou máximo de características; c) a adição ou remoçao de características não gera um subconjunto melhor; d) um subconjunto satisfatóriamente bom é encontrado, ou seja, ele cumpre alguns pré-requisitos estabelecidos.

Os critérios de parada são comumente utilizados para que não seja necessário realizar toda a busca, uma vez que o trabalho de exaustão da busca demanda muito tempo e recurso.

3.6 Validação de resultados

Para se avaliar os resultado, primeiramente deve-se ter algum conhecimento sobre os dados que foram submetidos para a seleção de características. Existem vários cálculos que podem ser feitos para que seja posível validar que o processo de seleção realmente resultou em um subconjunto com características que resolva o problema de uma maneira melhor do que o conjunto de características iniciais. Podemos utilizar três dimensões para poder verificar se um método de seleção realmente foi efetivo, são eles: acurácia de predição e o número de características selecionadas (LIU; MOTODA., 1998). Além dessas dimensões, devemos levar em consideração alguns fatos dos próprios algorítimos em si, como a velocidade de treinamento do algoritmo e a generalidade das características selecionadas.

Uma maneira bem comum de se validar os resultados é verificar se a acurácia do classificador foi melhorada, ou se a diminuição do número de características reduziu o tempo de treinamento e execução de um dado classificador sem prejudicar a sua acurácia (YU, 2005).

4 Modelos de seleção de características

4.1 Modelos de Seleção de Características

Como visto, existem várias maneiras e técnicas para se montar um modelo para seleção de características, podendo-se combinar as formas de busca, e as várias maneiras de avaliar os subconjuntos gerados, e os critérios de parada para chegar a um modelo que seja adequado ao problema, ou seja, encontrando um subconjunto que sejá otimizado em relação ao conjunto inicial. A combinação das técnicas geram três possíveis categorias de modelos: o modelo de filtro (*filter*), o modelo de envelopamento (*wrapper*) e o modelo híbrido (*hybrid*).

4.1.1 Modelo de Filtro (*filter*)

Os algoritmos de filtro não necessitam de um algoritmo de *machine learning* para poder fazer a seleção de características, utilizando-se somente das próprias características para poder avaliar os subconjuntos, geralmente necessitando de um menor poder computacional para poder serem utilizados. A Figura 4 ilustra como trabalha um algoritmo de filtro.

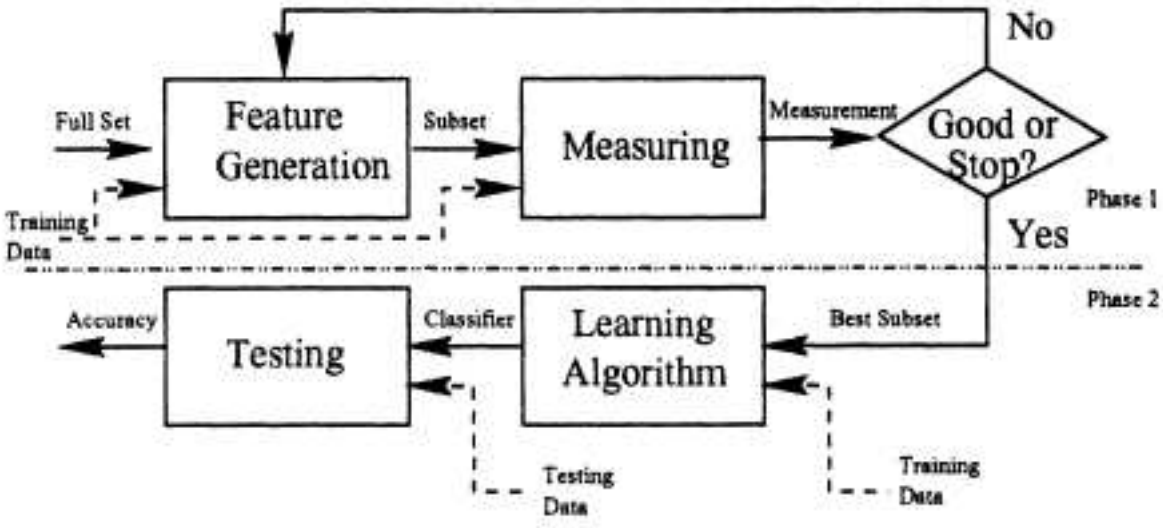

Figura 4 – Fluxograma do modelo de filtro (*filter*). (LIU; MOTODA., 1998)

O modelo é composto por duas fases, a primeira fase consiste em selecionar o melhor subcojunto possível utilizando alguma das medidas citadas no capítulo anterior para poder avaliar esses subconjuntos. A segunda fase consiste em utilizar o subconjunto

gerado em um classificador. Nessa fase também é colocado os dados de treino no classificador para avaliar se o resultado foi satisfatório. Os algoritmos de filtro não dependem da avaliação do classificador, e sim das informações e medidas obtidas diretamente dos dados. (LIU; MOTODA., 1998)

Pode-se generalizar o modelo de filtro de acordo com a Figura 5, onde para um dado conjunto D de características, é escolhido um subconjunto *S0* através de alguma das formas de busca. Cada subconjunto gerado é avaliado e comparado com o seu anterior, sempre sendo escolhido aquele que tenha um melhor valor em seu critério de avaliação (γ). Esse processo será repetido até que um critério de parada (α) seja alcançado ou que sejam testados todos os conjuntos possíveis. (YU, 2005)

Filter Algorithm
input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
δ // a stopping criterion
output: S_{best} // an optimal subset

```
01 begin
02     initialize: S_best = S_0;
03     γ_best = eval(S_0, D, M); // evaluate S_0 by an independent measure M
04     do begin
05         S = generate(D); // generate a subset for evaluation
06         γ = eval(S, D, M); // evaluate the current subset S by M
07         if (γ is better than γ_best)
08             γ_best = γ;
09             S_best = S;
10     end until (δ is reached);
11     return S_best;
12 end;
```

Figura 5 – Generalização do algoritmo de filtro (*filter*). (YU, 2005)

4.1.2 Modelo de Envelopamento (*wrapper*)

Os algoritmos que se baseiam no modelo de envelopamento necessitam de um algoritmo de *machine learning* pré determinado para que seja possível utiliza-lo. Isso se deve ao fato de que o modelo usa o próprio classificador para avaliar o quão bom é o subconjunto de dados gerado, utilizando a acurácia do classificador para avaliar os subconjuntos. A Figura 6 ilustra como funciona um algoritmo de envelopamento.

O modelo é composto por duas fases, a primeira fase consiste em escolher um subconjunto utilizando um classificador para poder avaliar o quão bom é esse subconjunto. Já a segunda fase é igual a do modelo de filtro, onde o subconjunto é utilizado no classi-

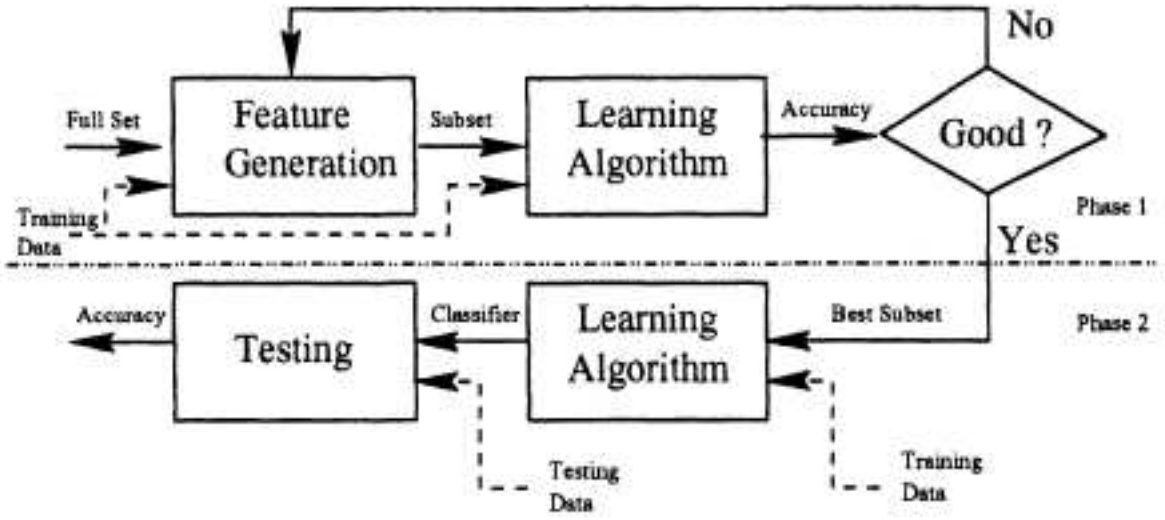

Figura 6 – Fluxograma do modelo de envelopamento (*wrapper*). (LIU; MOTODA., 1998)

ficador e então são inseridos os dados de treino para verificar se o resultado foi realmente satisfatório.

Pode-se generalizar o modelo de envelopamento de acordo com a Figura 7, onde para um dado conjunto D de características, é escolhido um subconjunto $S0$ através de alguma das formas de busca. Cada conjunto gerado é avaliado e comparado com o seu anterior utilizando um algoritmo de *machine learning*, sempre sendo escolhido aquele que tenha um melhor valor em seu critério de avaliação (γ). Esse processo será repetido até que um critério de parada (α) seja alcançado ou que sejam testados todos os conjuntos possíveis. (YU, 2005)

4.1.3 Modelo Híbrido (*hybrid*)

Os algoritmos que se baseiam no modelo híbrido são compostos por técnicas do modelo de filtro e de envelopamento. O modelo híbrido busca utilizar da melhor forma possível os dois modelos para poder alcançar um melhor subconjunto. Geralmente utiliza-se o modelo híbrido para problemas como muitos dados (YU, 2005). Na Figura 8 veremos como seu algorítimo funciona.

Para um dado conjunto D de características, é escolhido um subconjunto $S0$ através de alguma das formas de busca, geralmente utiliza-se um conjunto vazio e é adicionado características a cada iteração. A cada iteração se adiciona uma característica e é comparada subconjunto a subconjunto dessa iteração utilizando medidas independentes (γ), ou seja, das características em si, que terá c + 1 características, até que seja possível encontrar o melhor subconjunto. Repete-se o processo até que se ache o melhor subconjunto de cada uma das iterações, e então esses subconjunto serão então comparados utilizando

Wrapper Algorithm
input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
δ // a stopping criterion
output: S_{best} // an optimal subset
01 **begin**
02 **initialize:** $S_{best} = S_0$;
03 $\gamma_{best} = eval(S_0, D, A)$; // evaluate S_0 by a mining algorithm A
04 **do begin**
05 $S = generate(D)$; // generate a subset for evaluation
06 $\gamma = eval(S, D, A)$; // evaluate the current subset S by A
07 if (γ is better than γ_{best})
08 $\gamma_{best} = \gamma$;
09 $S_{best} = S$;
10 **end until** (δ is reached);
11 **return** S_{best};
12 **end**;

Figura 7 – Generalização do algoritmo de envelopamento (*wrapper*). (YU, 2005)

algum classificador (θ), para que seja obtido o melhor subconjunto possível. (YU, 2005)

4.2 Modelos Escolhidos

4.2.1 *Relief-F* - Modelo de Filtro

Esse modelo foi escolhido pelo fato de ser muito utilizado e obter bons resultados, como apontado por Dash (1997) e Yu (2005). O maior problema encontrado com o algoritmo foi o fato de seu predecessor, o Relief, tratar apenas de classes binárias, o que traria certos problemas quanto a escolha das bases de dados a serem utilizadas no trabalho, assim como problemas quanto a compatibilidade de bases que seriam submetidas ao serviço que esse trabalho se propõe a implementar. *Relief-F* resolve esse impasse quanto ao tipo das bases de dados.

O algoritimo *Relief-F* utiliza métodos estatísticos para selecionar as características relevantes, é um método que se baseia nos pesos atribuídos às caracterísicas (DASH, 1997). O algorítmo realiza a busca de maneira sequêncial e utiliza medidas de distância para poder realizar a avaliação entre as características. Primeiramente o algorítmo inicializa os pesos (W) com 0, e escolhe um valor de limite (ϕ), então para uma característica n ele calcula os valores de *Near Hit* e *Near Miss*, ambos calculados através da distância euclidiana. *Near Hit* é a menor distância entre instancias de mesma classe, já o *Near Miss* é o menor valor entre instancias de classes diferentes. Depois de percorrer todas as características ele retorna aquelas que têm o valor do peso maior do que o limite

Hybrid Algorithm
input: $D(F_0, F_1, ..., F_{n-1})$ // a training data set with N features
S_0 // a subset from which to start the search
output: S_{best} // an optimal subset

```
01 begin
02     initialize: S_best = S_0;
03     c_0 = card(S_0); // calculate the cardinality of S_0
04     γ_best = eval(S_0, D, M); // evaluate S_0 by an independent measure M
05     θ_best = eval(S_0, D, A); // evaluate S_0 by a mining algorithm A
06     for c = c_0 + 1 to N begin
07         for i = 0 to N − c begin
08             S = S_best ∪ {F_j}; // generate a subset with cardinality c for evaluation
09             γ = eval(S, D, M); // evaluate the current subset S by M
10             if (γ is better than γ_best)
11                 γ_best = γ;
12                 S'_best = S;
13         end;
14         θ = eval(S'_best, D, A); // evaluate S'_best by A
15         if (θ is better than θ_best);
16             S_best = S'_best;
17             θ_best = θ;
18         else;
19             break and return S_best;
20     end;
21     return S_best;
22 end;
```

Figura 8 – Generalização do algoritmo híbrido (*hybrid*). (YU, 2005)

estabelecido. Esse processo está representado na Figura 9.

4.2.2 Método da Árvore de Decisão - *Decision Tree Method (DTM)* - Modelo de Filtro

O modelo DTM foi escolhido por ser um modelo de funcionamento um tanto peculiar, uma vez que se utiliza o algoritmo C4.5 para realizar a seleção de características, ou seja, utiliza-se um classificador para poder escolher as características, porém ele não é um algorítmo de seleção de Envelopamento (*wrapper*), ele continua sendo um algorítmo do tipo Filtro (*filter*) pelo fato de utilizar medidas internas do classificador relacionadas a erro, e não só as medidas de acurácia. (DASH, 1997). Após montar a árvore de decisão, é feita a 'poda' da árvore, e é dessa árvore 'podada' que são extraídas as características que serão utilizadas no classificador final, não sendo necessário ser o C4.5.

Na Figura 10 é visto como é proposto esse modelo. Pode parecer simples, porém o maior trabalho advém do algoritmo C4.5, que realizará todos os calculos e montará a

Relief(D, S, *NoSample*, *Threshold*)
(1) $T = \phi$
(2) Initialize all weights, W_i, to zero.
(3) For $i = 1$ to *NoSample*/* Arbitrarily chosen */
Randomly choose an instance x in D
Finds its *nearHit* and *nearMiss*
For $j = 1$ to N
$W_j = W_j - diff(x_j, nearHit_j)^2 + diff(x_j, nearMiss_j)^2$
(4) For $j = 1$ to N
If $W_j \geqslant Threshold$
Append feature f_j to T
(5) Return T

Figura 9 – Representação do Modelo Relief-F. (DASH, 1997)

árvore que servirá de insumo para o classificador final.

DTM(D)
(1) $T = \phi$
(2) Apply C4.5 to the training set, D
(3) Append all features appearing in the pruned decision tree to T
(4) Return T

Figura 10 – Representaçao do Modelo DTM. (DASH, 1997)

4.2.3 *Linear Forward Selection (LFS)* - Modelo de Envelopamento

A escolha do modelo LFS foi feita com base na premissa de que já haviam sido escolhidos dois modelos do tipo filtro, então seria bom para o trabalho a escolha de um modelo de envelopamento. O modelo LFS é um modelo de Envelopamento que utiliza a *sequential forward generation* para realizar a busca. Para reduzir o número de conjuntos avaliados pelo algoritmo, primeiramente todos os conjuntos são ranqueados usando alguma medida de Filtro, em seguida são construídos N subconjuntos, em que o primeiro subconjunto possui a característica melhor ranqueada, o segundo subconjunto possui as duas características melhores ranqueadas, o terceiro subconjunto as três características melhores ranqueadas, até que se construa todos os subconjuntos com todas as características. Então é feito a avaliação desses subconjuntos utilizando algum classificador, no caso será utilizado o classificador KNN. (GUTLEIN, 2009).

4.2.4 Implementação dos Modelos

Para realizar a implementação dos modelos, foram feitas pesquisas de ferramentas de mineração de dados. Durante as pesquisas foram encontradas ferramentas capazes de auxiliar o projeto, sendo as que mais se destacam: Scikit-learn (COURNAPEAU, 2007), Weka (WITTEN, 2005) e Apache Spark (FOUNDATION, 2014). Dentre as ferramentas, a que mais se destacou a esse trabalho foi a Weka, pelo fato dela ter alguns dos métodos já implementados, e pelo fato dela ser uma biblioteca código aberto, o que facilitaria caso alguma mudança fosse necessária. A biblioteca também possui uma grande popularidade e comunidade, o que auxilia no seu uso.

A ferramenta Weka é composta por um conjunto de algoritmos de aprendizado de máquina e de mineração de dados. (WITTEN, 2005). Esses algoritmos podem ser utilizados na própria ferramenta como também podem ser utilizados em códigos java, através da biblioteca Weka, que é de código aberto e possui licença GPL 3 (GPLV3, 2007).

Os modelos foram implementados com o auxílio da biblioteca Weka, e seus resultados estão descritos no próximo capítulo.

5 Validação dos Resultados

Nesse capítulo são analisados os modelos propostos no capítulo anterior. Cada um dos modelos deverá selecionar um conjunto de características para ser utilizado em um classificador, e então deverão ter seus resultados coletados e comparados.

5.1 Validação dos Modelos

Os modelos selecionados devem contribuir para um melhor desempenho de um dado modelo de Aprendizado de Máquina, sendo assim, serão testados utilizando uma base de dados e levando em consideração a acurácia do classificador. A base de dados a ser utilizada é a dos sobreviventes do titanic, e a MADELON, uma base utilizada para competições. Cada um dos modelos serão testados utilizando essas bases seguindo os seguintes critérios:

1. Será utilizado o classificador kNN (*k-Nearest Neighbours*)

2. Será realizado um pré-processamento na base de dados para:

   Remover características que sejam do tipo texto

   Substituir os campos vazios pela mediana daquele campo

3. Da base de dados, será utilizado 2/3 dela para realizar o treinamento e 1/3 para realizar os testes

4. Cada modelo deverá ser executado 10 vezes e o desempenho final será a média desses resultados.

O classificador escolhido foi o *k-Nearest Neighbours* por ser um classificador simples apesar de ser computacionalmente exaustivo. Um dos motivos que levaram a escolha desse classificador é o fato de que ele é muito sensível às características irrelevantes, fazendo com que o processo de seleção de características seja um forte aliado a esse classificador.

Para cada uma das bases será realizada o procedimento apontado no diagrama da Figura 11.

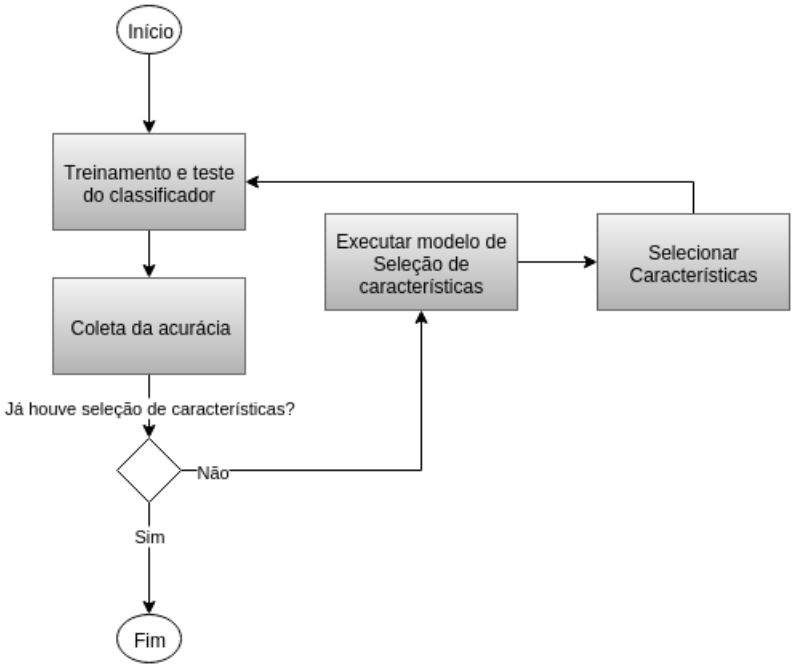

Figura 11 – Fluxograma da validação dos resultados

1. Treinamento do classificador com 2/3 da base de dados sem nenhum processo de seleção

2. Coleta da acurácia obtida do classificador após serem feitos os testes com 1/3 da base

3. Realização do processo de seleção de características

4. Treinamento do classificador utilizando somente as características selecionadas

5. Coleta da acurácia obtida do classificador após serem feitos os processos de seleção e os testes com 1/3 da base

As características da base de dados do Titanic, após ser realizado o pré processamento, se encontram da seguinte maneira descrito na Tabela 1.

Tabela 1 – Características da base Titanic

Característica	**Tipo**
survived	Nominal {0, 1}
pClass	Numérico
sex	Nominal {male, female}
age	Numérico
sibsp	Numérico
parch	Numérico
fare	Numérico
embarked	Nominal {Q,S,C}

O classificador a ser utilizado, as instancias e a acurácia inicial do classificador se encontram descritos na Tabela 2.

Tabela 2 – Informações sobre o classificador e instancias da base Titanic

Classificador	**kNN (Weka.IBk)**
N. de características inicial	8
Acurácia inicial média	72.8977%
Instancias para treino	607
Instancias para teste	303

Após cada um dos métodos de seleção serem aplicados à base de dados, os resultados obtidos estão expostos na Tabela 3:

Tabela 3 – Resultados dos modelos - base Titanic

Modelo	**Características Mantidas**	**Acurácia média Classificador**
Relief-F	sex, pClass, age	80.5281%
DTM	sex, age, sibsp, pclass, fare	78.5479%
LFS	sex, fare	81.538%

Pode-se verificar que o processo de seleção de características foi efetivo, uma vez que ele diminuiu o número de características utilizadas inicialmente, além de aumentar a acurácia do classificador. Não foi feita qualquer alteração no classificador, a única mudança foi no número de características utilizadas.

Os resultados acima são para uma base com poucas características mas ainda sim foi possível ver uma melhora no desempenho do classificador. A base do Titanic é executada em poucos segundos, sendo impossível verificar o fato de que a diminuição

de características leva a um melhor desempenho no quesito computacional. Para poder melhor exemplificar esse atributo será utilizado uma base maior.

MADELON (2003) é uma base de dados artificial que foi parte de um desafio chamado NIPS, realizado em 2003. É um problema de duas classes que tem como característica ser altamente não linear. A base de dados consta com 1820 instancias e 500 características, alem da classe. Mantendo o planejamento inicial, que é de utilizar 1/3 da base para realizar os testes, temos o status inicial do classificador da seguinte maneira:

Tabela 4 – Informações sobre o classificador e instancias da base MADELON

Classificador	**KNN (Weka.IBk)**
Base de dados	MADELON
N. de características inicial	500
Acurácia inicial média	52.0194%
Tempo para treinamento	12 segundos
Instancias para treino	1227
Instancias para teste	613

Após cada um dos métodos de seleção ser aplicado à base de dados, os resultados obtidos estão expostos na Tabela 5:

Tabela 5 – Resultados dos modelos - base MADELON

Modelo	**Características Mantidas**	**Acurácia média do Classificador**	**Tempo de treinamento médio**	**Tempo medio para selecionar características**
Relief-F	a_28, a_48, a_64, a_105, a_128, a_153, a_241, a_281, a_318, a_336, a_338, a_378, a_433, a_442, a_451, a_453, a_455, a_472, a_475, a_493.	88.6914%	<1 segundo	37 segundos
DTM	a_105, a_442, a_338, a_475, a_451, a_128, a_28, a_336, a_119, a_493, a_48, a_251, a_292, a_318, a_433	80.937%	<1 segundo	2 segundos
LFS	a_105, a_128, a_241, a_336, a_338, a_442	91.2601%	<1 segundo.	332 segundos

Como é possível observar, o número de características selecionado foi menor do que 10% do número inicial e o desempenho do classificador foi aumentado em até 40%. O tempo gasto para realizar seu treinamento foi diminuído drásticamente, houve um gasto de tempo para realizar a seleção, porém os ganhos obtidos em questão de performance computacional e performance da acurácia de seleção mostram que o processo de seleção proporcionou melhorias de grande peso.

6 Implementação Do Serviço

6.1 Visão Geral

Como proposto inicialmente, este trabalho de conclusão de curso visa a implementação de um Serviço que seja capaz de receber uma base de dados e aplicar os modelos de seleção de características, informando então quais são as características mais relevantes ao problema. O sistema foi implementado para a Web e possui uma interface onde o usuário pode, além de realizar a seleção de características, criar seus projetos e executar os modelos aplicados a cada um deles.

6.2 Arquitetura do Serviço

Durante a elaboração da arquitetura do serviço era preciso pensar em uma maneira de criar uma aplicação Web para que facilitasse o seu acesso, além de poder integrar o uso de outras ferramentas já utilizadas durante esse trabalho. Como os modelos já haviam sido implementados utilizando a biblioteca Weka, foi então necessário encontrar um conjunto de ferramentas que se adequasse à ela, e que pudesse desenvolver uma aplicação Web. Levando em consideração essas premissas foram escolhidos as ferramentas Ruby on Rails (HANSSON, 2008) e a linguagem JRuby (SIEGER, 2010). A combinação da linguagem JRuby com a ferramenta Ruby on Rails, permitiria utilizar o poder e agilidade de desenvolvimento Web da ferramenta Ruby on Rails, junto ao uso de bibliotecas implementadas em Java através do JRuby, sendo então uma ótima combinaçao para desenvolver o sistema. Alguns problemas foram encontrados ao tentar realizar a comunicação das ferramentas, como por exemplo as divergências entre a estrutura da documentação do Weka com a sua estrutura de classes, bibliotecas Ruby que não eram compatíveis com JRuby e tratamento de exceções da biblioteca Weka. Todos os problemas foram resolvidos ao longo do desenvolvimento da aplicação.

Após a escolha das técnologias e ferramentas a serem utilizadas, foi então pensado como seria a arquitetura. A arquitetura do serviço foi pensada para que o usuário submetesse um arquivo contendo a base de dados em que ele gostaria que fosse realizado a Seleção de Características, esse processo segue as seguintes fases:

1. Pré Processamento: Onde a base é trabalhada a fim de deixá-la pronta para que um dos modelos de seleção seja aplicado sobre ela. Essa fase consiste em preencher as lacunas e remover as características que não possam ser utilizadas no classificador kNN, como ja utilizado anteriomente no Capítulo 5

2. Executar Base de Dados: Após a base ser pré processada, ela será executada escolhendo um dos métodos escolhidos e citados no Capítulo 4

3. Atualizar Status da Execução: Quando o processo de execução acabar, será realizada a atualização dos dados da execução, tais como as características escolhidas, a acurácia obtida e o tempo gasto para executar a base de dados.

O Serviço utiliza-se da arquitetura mostrada na Figura 12.

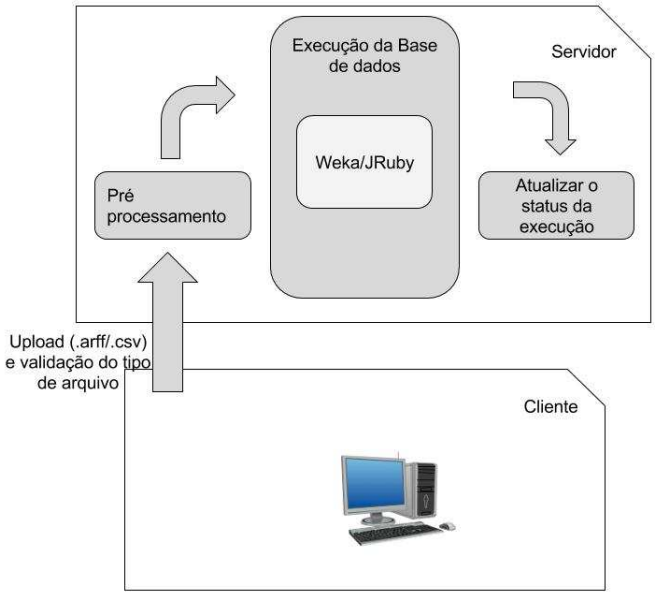

Figura 12 – Arquitetura do Serviço de Seleção de Características

Podemos ver que o sistema age em sua grande maioria no lado servidor, onde ele realiza o Pré Processamento, a Execução da base de dados e a Atualização do Status da Execução. Podemos perceber também o uso de Threads de processamento para que seja possível melhorar o desempenho do sistema, uma vez que o JRuby utiliza de Threads reais em seu processamento (SIEGER, 2010), fazendo com que o uso de Threads possua um resultado satisfatório.

Tendo essa visão da arquitetura do sistema, foi pensado como seria o relacionamento entre as diversas entidades do serviço. As principais entidades são *User, Project e Execution*. Essas entidades compõem o sistema para que seja possivel melhor organizar o seu funcionamento.

Na Figura 13 podemos observar o fluxo de funcionamento do serviço que conta com um sistema de *Login*, onde os usuários fornecem suas credenciais, além de poderem se registrar fornecendo um email e uma senha. Após o usuário se registrar no sistema, ele tem acesso a criação de um projeto, ou selecionar um projeto ja existente para que possa

executá-lo. No caso da criação de um novo projeto, o usuário deve fornecer um nome, uma descrição e uma base de dados para que seja possível a sua criação, essa base de dados deve respeitar os pré requisitos descritos mais adiante na fase de Pré Processamento.

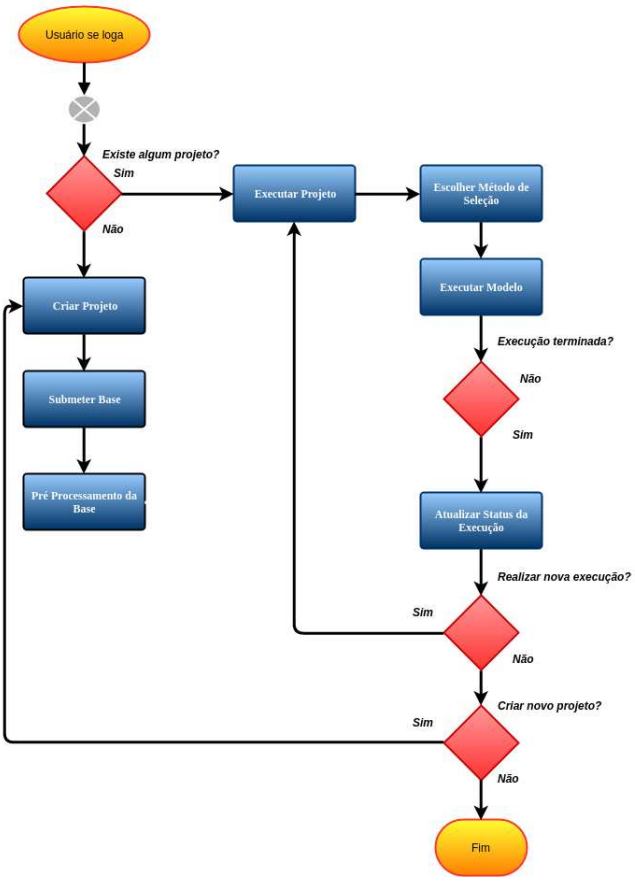

Figura 13 – Fluxograma do Serviço

Após realizado esse pré processamento e o projeto tendo sido criado, pode-se executa-lo. Ao escolher executar um projeto, será necessário escolher um modelo de seleção de características dentre os modelos estudados nesse trabalho. Ao escolher um dos modelos, o projeto será executado e então as características começarão a ser escolhidas. Esse processo de seleção é feito em paralelo, e após selecionadas as características, serão mostradas as características selecionadas e a acurácia obtida com essas características. As fases do serviço estão melhor explicadas na próxima seção.

6.3 Fases do Serviço

O serviço implementado segue as fases que estão listadas abaixo. Essas fases corroboram para o funcionamento da aplicação de modo que o seu fluxo de execução, mostrado na Figura 14, seja executado e, a partir disso, selecionar as características que sejam mais preponderantes ao problema de acordo com os métodos de seleção de características, conforme mostrado nos capítulos anteriores.

6.3.1 Pré Processamento

A fase de Pré Processamento é de suma importancia ao serviço pois a base de dados é adquirida nessa fase, e ao ser adquirida ela deve ser analisada para que seja possivel constatar a necessidade de algum tipo de pre processamento, tal como a conversão, pois a biblioteca Weka, que é utilizada para trabalhar as bases de dados, funciona apenas com arquivos do formato *Attribute-Relation File Format* (.arff), sendo assim, qualquer base que não esteja nesse formato não será aceita, salvo a exceção do formato *Comma-separated values* (.csv), onde é utilizado um conversor para que seja possível utilizar essa base se ele estiver no formato descrito na Figura 14.

```
Length,Diameter,Height,Whole_weight,Shucked_weight,Viscera_weight,Shell_weight,Rings
0.455,0.365,0.095,0.514,0.2245,0.101,0.15,15
0.35,0.265,0.09,0.2255,0.0995,0.0485,0.07,7
0.53,0.42,0.135,0.677,0.2565,0.1415,0.21,9
0.44,0.365,0.125,0.516,0.2155,0.114,0.155,10
0.33,0.255,0.08,0.205,0.0895,0.0395,0.055,7
0.425,0.3,0.095,0.3515,0.141,0.0775,0.12,8
```

Figura 14 – Arquivo Formato .csv

Conforme mostrado na figura, a primeira linha deve descrever o nome das características do problema, e as demais linhas devem compor os dados da base, estando cada um dos dados alinhados a sua característica. Percebe-se que todas as informações são separadas por vírgulas, e esse padrão deve ser mantido para que o conversor reconheça a estrutura do arquivo e consiga convertê-lo.

O resultado dessa conversão pode ser observado na Figura 15, onde é possível ver um arquivo .arff já convertido, e como ele é configurado.

Figura 15 – Arquivo Formato .arff

Podemos observar que na primeira linha existe o nome da base, depois é descrito cada uma das características e o seu tipo (numérico, nominal, etc. como já dito em capítulos anteriores). Após descrever cada uma das características, é descrito os dados que compõem essa base, seguindo o mesmo padrão que é seguido no .csv, ou seja, separados por virgula.

Além dessa analise da base, também é analisado se existem lacunas na base de dados. Essas lacunas são dados que não estão inclusos, ou seja, estão nulos, e esse tipo de lacuna tende a não ser aceito por alguns classificadores, além de gerar ruído ou atrapalhar o treinamento do modelo (BATISTA, 2002). Para mitigar esse problema o sistema substituirá todas as lacunas pela média, em caso de atributo quantitativo, ou pela moda, em caso de atributo qualitativo. O sistema assume essa abordagem para que se tente evitar o *overfitting*, que é quando um modelo se ajusta aos dados de treinamento e não necessariamente consegue generalizar quando se recebe outros dados (HAWKINS, 2004).

Após realizadas todas essas etapas, a base finalmente estará pronta para ser executada na fase de Execução da Base de Dados.

6.3.2 Execução da Base de Dados

Durante essa etapa, a base submetida na fase de Pré Processamento será submetida a um dos modelos selecionado pelo usuário, e que foram implementados e mostrados anteriomente neste trabalho, sendo eles: *Relief-F, Decision Tree Method e Linear Forward Selection with kNN*. Quando o usuário seleciona o modelo, e dá inicio a execução da base de dados, o sistema aciona a biblioteca Weka, onde está a implementação dos modelos, e criará uma *Thread* para executa-lo. Quando a *Thread* finaliza o seu processamento, ela irá concluir a execução e passará para a próxima fase.

6.3.3 Atualização do Status da Execução

Após realizado a seleção de características pela fase de Execução da Base de Dados, terá inicio a fase de Atualização do Status da Execução, onde serão atualizados os dados da execução criada, mostrando as características que foram selecionadas após a execução do modelo, a acurácia dessas características aplicadas a um classificador kNN, a acurácia com todas as características, e o tempo gasto para que fosse executado esse método. O usuário poderá então visualizar quais são as características mais preponderantes ao seu modelo, e todos os outros dados citados.

7 Considerações Finais

Este projeto é composto por duas etapas, sendo este trabalho a primeira e o Trabalho de Conclusão de Curso 2 a segunda, a ser realizada em 2016. A primeira parte consistiu na fundamentação teórica e na demonstração dos algoritmos a serem utilizados no serviço que será implementado. Consistiu também na experimentação dos algoritmos e verificação da sua eficácia, como apontado no capítulo 4. Já a segunda parte, que consistirá na implementação do serviço de submissão de base de dados e seleção de características.

7.1 Trabalho Realizado

As etapas propostas nesse trabalho foram descritas no capítulo introdutório, e resumidas no capítulo de Desenvolvimento, sendo elas:

1. Especificar o processo de seleção de características
2. Pesquisar modelos de seleção de características a serem utilizados
3. Implementar os modelos de seleção de características
4. Realizar testes nos modelos
5. Específicar arquitetura do serviço
6. Construir protótipo do serviço
7. Realizar testes no protótipo do serviço

Após realizadas todas as etapas foi possível alcançar um serviço que condiz com o objetivo inicial desse trabalho, que era: "Esse trabalho tem como objetivo principal a implementação de um serviço que fosse capaz de analisar um conjunto de características e extrair o melhor subconjunto possível analisando-o em meio a três modelos de seleção de características". Os três modelos foram desenvolvidos e implementados junto ao serviço, e o serviço foi concluído utilizando um conjunto de tecnologias que permitiu o seu desenvolvimento e ainda deixando possíveis espaços para novos modelos. O trunfo desse trabalho é poder dar ao usuário o poder de realizar a seleção de características sem saber muito sobre o processo de seleção em si, e sem saber codificar, apenas necessita da base de dados a ser executada e que ela siga os padrões estabelecidos nos capítulos anteriores.

Depois de revisitar todo o conteúdo estudado e gerado nesse trabalho, pode-se concluir que foi satisfatório o desempenho alcançado com a geração do serviço e que o trabalho deixa continuidade para futuras melhoras e implementações.

7.2 Trabalhos Futuros

Apesar do trabalho ter tido bons frutos, ainda existem possiveis melhoras a serem feitas, pois não houve tempo hábil para implementar mais funcionalidades. Abaixo estão enumeradas alguns futuros trabalhos que podem ser realizados.

1. Desevolver mais modelos: Como mostrado nesse trabalho, existem vários modelos que podem ser implementados
2. Pesquisa no desenvolvimento de mais modelos: Utilizando as diversas técnicas de busca, avaliação e seleçào, podem-se criar vários novos modelos.
3. Utilizar outros classificadores: Nesse trabalho foi utilizado o classificador kNN, mas pode-se fazer o estudo utilizando outros classficadores
4. Deixar o serviço mais customizável: Apesar do serviço ser um tanto flexível quanto aos modelos, pode-se colocar mais classficadores, combinações de modelos, etc.

Tais avanços seriam de grande auxílio para o sistema, deixando-o ainda mais completo e tornando-o uma ferramenta ainda mais poderosa.

Referências

BATISTA, C. M. G. An Analysis of Four Missing Data Treatment Methods for Supervised Learning. p. 142–152, 2002. Citado na página 47.

BRINK, H.; RICHARDS, J. *Real World Machine Learning*. [S.l.]: Manning Publications C.O, 2014. Citado 2 vezes nas páginas 15 e 23.

BUSINESSWEEK, B. *The Current State of Business Analytics: Where Do We Go from Here?* 2011. Disponível em: <http://www.sas.com/resources/asset/busanalyticsstudy_wp_08232011.pdf>. Citado na página 15.

COURNAPEAU, D. *Scikit-Learn*. 2007. <http://scikit-learn.org/stable/>. [Online; accessed 10-June-2016]. Citado 2 vezes nas páginas 20 e 34.

DASH, H. L. M. Feature selection for classification. p. 131–156, 1997. Citado 7 vezes nas páginas 9, 19, 25, 27, 32, 33 e 34.

FOUNDATION, A. *Apache Spark*. 2014. <http://spark.apache.org/>. [Online; accessed 10-June-2016]. Citado 2 vezes nas páginas 20 e 35.

GPLV3. *General Public License Version 3*. 2007. Disponível em: <http://www.gnu.org/licenses/gpl-3.0.en.html>. Citado na página 35.

GUTLEIN, M. Large-scale attribute selection using wrappers. p. 332–339, 2009. Citado 2 vezes nas páginas 19 e 34.

GUYON, I. An introduction to variable and feature selection. p. 1157–1182, 2003. Citado na página 15.

HANSSON, D. *Ruby on Rails*. 2008. <http://rubyonrails.org/>. [Online; accessed 10-June-2016]. Citado 2 vezes nas páginas 20 e 43.

HAWKINS, D. The Problem of Overfitting. p. 1–12, 2004. Citado na página 47.

LIU, H.; MOTODA., H. *Feature selection for knowledge discovery and data mining*. [S.l.]: SPRINGER SCIENCE+BUSINESS MEDIA, LLC, 1998. ISBN 978-1-4615-5689-3. Citado 9 vezes nas páginas 9, 24, 25, 26, 27, 28, 29, 30 e 31.

MADELON. *Madelon Dataset*. 2003. Disponível em: <https://archive.ics.uci.edu/ml/datasets/Madelon>. Citado 2 vezes nas páginas 17 e 40.

MITCHELL, T. *Machine Learning*. New York, NY: McGraw-Hill, 1997. ISBN 0-07-042807-7. Citado 2 vezes nas páginas 15 e 23.

MOLINA, L. C. Feature selection algorithms: A survey and experimental evaluation. v. 17, p. 306–313, 2002. Citado na página 24.

OBSERVATORY, B. I. Big data: Artificial intelligence. Netherlands, p. 1 – 15, 2013. Citado na página 15.

SIEGER, N. *JRuby*. 2010. <http://jruby.org/>. [Online; accessed 10-June-2016]. Citado 3 vezes nas páginas 20, 43 e 44.

WITTEN, E. F. I. H. *Data Mining Practical Machine Learning Tools and Techniques*. [S.l.]: Diane Cerra, 2005. ISBN 0-12-088407-0. Citado 4 vezes nas páginas 17, 20, 34 e 35.

YU, H. L. e L. Toward integrating feature selection algorithms for classification and clustering. v. 17, p. 491–502, 2005. ISSN 1041-4347. Citado 11 vezes nas páginas 9, 23, 24, 25, 26, 27, 28, 30, 31, 32 e 33.