



UNIVERSIDADE FEDERAL DE SERGIPE  
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA  
DEPARTAMENTO DE COMPUTAÇÃO

## **ml.js: Framework para Aprendizado de Máquina com JavaScript**

Trabalho de Conclusão de Curso

José Fernando Santana Andrade



São Cristóvão – Sergipe

2016

UNIVERSIDADE FEDERAL DE SERGIPE  
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA  
DEPARTAMENTO DE COMPUTAÇÃO

José Fernando Santana Andrade

## **ml.js: Framework para Aprendizado de Máquina com JavaScript**

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a): Prof. Dr. Hendrik Teixeira Macedo

São Cristóvão – Sergipe

2016

---

José Fernando Santana Andrade

ml.js: Framework para Aprendizado de Máquina com JavaScript/ José Fernando Santana Andrade. – São Cristóvão – Sergipe, 2016-  
114 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Hendrik Teixeira Macedo

Trabalho de Conclusão de Curso – UNIVERSIDADE FEDERAL DE SERGIPE  
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA  
DEPARTAMENTO DE COMPUTAÇÃO, 2016.

1. Aprendizagem de máquina. 2. JavaScript. I. Orientador Prof. Dr. Hendrik Teixeira Macedo. II. Universidade Federal de Sergipe. III. Título

CDU 02:141:005.7

---

*Dedico este trabalho a todos os meus familiares, amigos e professores que me deram o apoio necessário para chegar aqui.*

# Agradecimentos

Agradeço a Deus, que provê tudo a seu tempo e nada deixa falta;  
à minha família, minha mãe Edileuza que sempre me apoiou mesmo estando distante,  
meu pai José, meus irmãos Patrícia e Vagner, meus avós e tios pela compreensão apoio e força;  
aos meus colegas e amigos da república com os quais convivi grande parte da graduação  
e tenho muita gratidão por tudo;  
ao meus colegas de classe muitos hoje amigos;  
aos integrantes da Moobi Tech, colegas e amigos pela compreensão e apoio;  
ao professor e amigo Hendrik, por compartilhar comigo um pouco de seus conhecimentos,  
acreditar e me guiar na realização desse trabalho; e principalmente  
à minha amada esposa Joseane, por está ao meu lado em todos os momentos, contribuindo  
com meu crescimento pessoal e profissional, por todo apoio, conselhos e incentivos  
cruciais para a conclusão de mais essa etapa;  
enfim agradeço a todos.

*Cada escolha uma renúncia, isso é a vida.  
(Charlie Brown Jr)*

# Resumo

Aprendizagem de máquina possui técnicas que permite o desenvolvimento de sistemas computacionais que aprendem de forma automática a partir de dados passados. Com a constante evolução da tecnologia, o uso dessas técnicas está cada vez mais comum e muitas vezes passam despercebidas. As mais diversas áreas estão usando aprendizagem de máquina no intuito de melhorar seus resultados. Este trabalho objetiva o desenvolvimento de uma ferramenta que possa ser utilizada para a aplicação de técnicas de aprendizado de máquina de forma simples por meio de um navegador Web através da linguagem de programação JavaScript. O presente texto apresenta conceitos sobre aprendizagem de máquina, descreve as ferramentas e processo de desenvolvimento do *framework* ml.js, uma breve discussão sobre *Open Data*, descreve os experimentos e resultados obtidos neste trabalho.

**Palavras-chave:** Aprendizagem de máquina, JavaScript.

# **Abstract**

Machine learning has techniques that allows the development of computer systems that learn automatically from past data. With the constant evolution of technology, the use of these techniques is increasingly common and often go unnoticed. The various areas are using machine learning in order to improve their results. This study aims at the development of a tool that can be used for the application of learning techniques simply machine via a Web browser through JavaScript programming language. This paper presents concepts of machine learning, describes the tools and development process ml.js framework, a brief discussion of Open Date, describes the experiments and results obtained in this study.

**Keywords:** Machine learning, JavaScript.

# **Lista de ilustrações**

Figura 1 – Exemplo carregamento de arquivo com o WEKA ( <i>Print screen</i> ) . . . . .	16
Figura 2 – Exemplo de uso da biblioteca Encog para JavaScript (HEATON, 2012) . . . . .	17
Figura 3 – Exemplo de avore de decisão gerada por (SIMEONE; FERREIRA, 2014) . . . . .	18
Figura 4 – Representação do processo de AM (MACEDO, 2014) . . . . .	21
Figura 5 – Exemplos de aplicações de AM em problemas do mundo real - fontes Facebook, Google Play e Microsoft respectivamente . . . . .	22
Figura 6 – Hierarquia de aprendizado - adaptado de (FACELI et al., 2011) . . . . .	23
Figura 7 – Conjunto de dados de exemplo preço <i>versus</i> tamanho de imóveis (NG, 2012a)	24
Figura 8 – Exemplo de aproximação de funções para conjunto de dados - adaptado de (NG, 2012a) . . . . .	24
Figura 9 – Separatividade linear - (autoria própria) . . . . .	25
Figura 10 – Exemplo de regressão linear (WIECKI, 2015) . . . . .	26
Figura 11 – Exemplo de agrupamentos de objetos - adaptado de (FACELI et al., 2011) . .	27
Figura 12 – Exemplo aprendizagem por reforço <i>game Need For Speed Underground - Gamespot</i> . . . . .	28
Figura 13 – Mapa mental dos algoritmos de aprendizagem de máquina (BROWNLEE, 2013) . . . . .	31
Figura 14 – Os 1, 3 e 5 vizinhos mais próximos do ponto no centro dos círculos - adaptado de (TAN; STEINBACH; KUMAR, 2009) . . . . .	33
Figura 15 – Impacto da escolha do valor de $k$ para classificação de vizinhos mais próximos (autoria própria) . . . . .	34
Figura 16 – Representação de árvore de decisão (autoria própria) . . . . .	35
Figura 17 – Exemplo árvore de decisão e as regiões de decisão no espaço de objetos (MACEDO, 2014) . . . . .	36
Figura 18 – Gráfico da entropia (autoria própria) . . . . .	37
Figura 19 – Raiz da árvore de decisão com base nos dados da Tabela 1 (autoria própria)	40
Figura 20 – Árvore gerada seguindo o ramo sol do atributo clima (autoria própria) . . .	41
Figura 21 – Demonstração do algoritmo k-Means - adaptado de Wikipédia . . . . .	42
Figura 22 – Exemplos de conjuntos de dados em que o k-Means apresenta dificuldade para a formação dos grupos - adaptado de (MACEDO, 2014) . . . . .	44
Figura 23 – Ilustração densidade baseada em centro e tipos de pontos - adaptado de (TAN; STEINBACH; KUMAR, 2009) . . . . .	45
Figura 24 – Tipos de pontos encontrados por DBSCAN para um determinado conjunto de dados - adaptado de (CASSIANO, 2014) . . . . .	46
Figura 25 – Resultado algoritmo DBSCAN (autoria própria) . . . . .	46

Figura 26 – Pontos de dados de preços <i>versus</i> área dos imóveis junto com um hipótese de função linear - adaptado de (NG, 2012a) . . . . .	48
Figura 27 – Descida do gradiente - adaptado de (NG, 2016) . . . . .	49
Figura 28 – Contorno da função erro - adaptado de (NG, 2012b) . . . . .	51
 Figura 29 – Visualização dois atributos do <i>dataset</i> Íris resultante do Código 3 (autoria própria) . . . . .	56
Figura 30 – Classes para geração de gráficos e leitura de arquivos (autoria própria) . . . . .	56
Figura 31 – Exemplo de matriz scatter plots para conjunto de dados Atom . . . . .	57
Figura 32 – Principais classes e métodos do ml.js (autoria própria) . . . . .	58
Figura 33 – Tela inicial da GUI (autoria própria) . . . . .	59
Figura 34 – Exemplo de valores <i>default</i> (autoria própria) . . . . .	60
Figura 35 – Exemplo de métricas geradas (autoria própria) . . . . .	60
Figura 36 – Exemplo fluxo de uso da GUI (autoria própria) . . . . .	61
 Figura 37 – Matrix de <i>scatter plot</i> para atributos do conjunto de dados Íris (autoria própria) . . . . .	64
Figura 38 – Matriz de <i>scatter plot</i> para atributos do conjunto de dados <i>GPS Trajectories</i> (autoria própria) . . . . .	65
Figura 39 – Gráfico k × Acurácia para os valores da Tabela 8 (autoria própria) . . . . .	72
Figura 40 – Árvore de decisão gerada pelo modelo ID3 aplicado à base GPS Trajectories (autoria própria) . . . . .	73
Figura 41 – Árvore de decisão gerada pelo modelo ID3 aplicado à base de imigrantes (autoria própria) . . . . .	74
Figura 42 – Demonstração do k-Means aplicado ao <i>dataset</i> t4.8k para diferentes valores de k (autoria própria) . . . . .	75
Figura 43 – Demonstração do k-Means aplicado aos <i>datasets</i> da Tabela 4 para o valor de k igual ao seu número de classes (autoria própria) . . . . .	76
Figura 44 – Demonstração do algoritmo DBSCAN aplicado ao <i>dataset</i> t4.8k para diferentes valores de <i>minPts</i> e <i>Eps</i> (autoria própria) . . . . .	77
Figura 45 – Demonstração do DBSCAN aplicado aos <i>datasets</i> da Tabela 4 (autoria própria) . . . . .	78
Figura 46 – Gráfico da evolução do MSE para experimento com Gradiente descendente e <i>dataset</i> número de medalhas e atletas olímpíadas 2016 (autoria própria) . . . . .	79
 Figura 47 – Exemplo gráfico MSE para um conjunto de dados com valores muito altos (autoria própria) . . . . .	81
 Figura 48 – Tela inicial da GUI . . . . .	102
Figura 49 – Destaque menu . . . . .	103
Figura 50 – Tela Dataset . . . . .	103
Figura 51 – Tela selecionando arquivo . . . . .	104
Figura 52 – Tela informações do <i>dataset</i> . . . . .	104

Figura 53 – Tela visualização do <i>dataset</i> . . . . .	105
Figura 54 – Tela inicial para atividade de agrupamento . . . . .	105
Figura 55 – Tela parâmetros k-Means . . . . .	106
Figura 56 – Métricas para o modelo k-Means . . . . .	106
Figura 57 – Exemplo resultado de agrupamento modelo k-Means . . . . .	107
Figura 58 – Tela parâmetros DBSCAN . . . . .	107
Figura 59 – Exemplo resultado de agrupamento modelo DBSCAN . . . . .	108
Figura 60 – Tela inicial para atividade de classificação . . . . .	108
Figura 61 – Opções de teste para modelos classificadores . . . . .	109
Figura 62 – Tela parâmetros k-NN . . . . .	110
Figura 63 – Exemplo de métricas para modelo k-NN . . . . .	110
Figura 64 – Gráfico com ponto usados na avaliação do modelo k-NN . . . . .	111
Figura 65 – Tela parâmetros ID3 . . . . .	111
Figura 66 – Árvore de decisão gerada como resultado de ID3 . . . . .	112
Figura 67 – Matrix de confusão e métricas para modelo de classificação ID3 . . . . .	112
Figura 68 – Tela inicial para atividade de regressão . . . . .	113
Figura 69 – Tela parâmetros Gradiente descendente . . . . .	113
Figura 70 – Exemplo métricas para modelo Gradiente descendente . . . . .	114
Figura 71 – Gráfico MSE × iteração para o modelo Gradiente descendente . . . . .	114

# **Lista de tabelas**

Tabela 1 – Exemplo fim de semana adaptado de (COLTON, 2004) . . . . .	39
Tabela 2 – Exemplos com valor de Clima igual a Sol . . . . .	40
Tabela 3 – Dados de exemplo para preços <i>versus</i> área dos imóveis - adaptado de (NG, 2012b) . . . . .	50
Tabela 4 – Conjuntos de dados utilizados para os experimentos com os modelos de agrupamento . . . . .	63
Tabela 5 – Exemplos de aplicativos e serviços que utilizam dados abertos (LABHACKER; NIC.BR; W3C BRASIL, 2011) . . . . .	68
Tabela 6 – Matriz de confusão para um problema de 2 classes . . . . .	70
Tabela 7 – Matriz de confusão para um problema não binário . . . . .	71
Tabela 8 – Métricas obtidas pela aplicação do modelo k-NN para diferentes valores de k com a base GPS Trajectories (autoria própria) . . . . .	72
Tabela 9 – Métricas obtidas pela aplicação do ID3 com validação cruzada usando 10 partições à base GPS Trajectories (autoria própria) . . . . .	73
Tabela 10 – Resultados experimento usando o ID3 com a base de refugiados . . . . .	74
Tabela 11 – Métricas obtida pelo k-Means para diferentes valores de $k$ aplicado ao <i>dataset t4.8k</i> . . . . .	75
Tabela 12 – Predições de medalhas para regressão linear resultantes do ml.js e WEKA . . . . .	79
Tabela 13 – Navegadores e versão mínima para usa do ml.js . . . . .	81

# **Lista de códigos**

Código 1 – Exemplo de uso do <i>prototype</i> . . . . .	54
Código 2 – Exemplo de leitura de arquivo . . . . .	55
Código 3 – Exemplo geração de gráfico . . . . .	55
Código 4 – Exemplo de inclusão do ml.js a um projeto . . . . .	92
Código 5 – Exemplo de inclusão dos modelos do ml.js de forma separada . . . . .	92
Código 6 – Exemplo de uso do k-Means . . . . .	93
Código 7 – Exemplo de uso do DBSCAN . . . . .	93
Código 8 – Exemplo de uso do k-NN . . . . .	93
Código 9 – Exemplo de uso do k-NN com função de distância personalizada . . . . .	94
Código 10 – Exemplo de uso do ID3 . . . . .	95
Código 11 – Exemplo de uso do Gradiente descendente para regressão univariada . . . . .	95
Código 12 – Exemplo de uso do Gradiente descendente para regressão multivariada . . . . .	96
Código 13 – Exemplo de obtenção de resultados para o Gradiente descendente . . . . .	96
Código 14 – Exemplo de leitura de arquivo e execução de <i>callback</i> após o carregamento . . . . .	96
Código 15 – Exemplo carregamento conjunto de dados a partir de JSON . . . . .	97
Código 16 – Trecho de código para transformações aplicadas ao dataset <i>GPS Trajectories</i> . . . . .	98
Código 17 – Exemplo de geração de visualização para resultados de agrupamentos . . . . .	99
Código 18 – Exemplo de uso da classe ConfusionMatrix . . . . .	100
Código 19 – Exemplo de uso das funções da classe utils . . . . .	101

# Sumário

<b>1</b>	<b>Introdução</b>	<b>15</b>
<b>2</b>	<b>Aprendizado de máquina</b>	<b>20</b>
2.1	Paradigmas de aprendizagem	20
2.1.1	Aprendizagem supervisionada	23
2.1.2	Aprendizagem não supervisionada	26
2.1.3	Aprendizagem por reforço	27
2.2	Modelos de aprendizado de máquina	28
2.3	Os modelos selecionados	32
2.3.1	k-NN	32
2.3.2	ID3	34
2.3.3	k-Means	41
2.3.4	DBSCAN	44
2.3.5	Gradiente descendente	48
<b>3</b>	<b>O framework ml.js</b>	<b>53</b>
3.1	API	53
3.2	GUI	59
<b>4</b>	<b>Experimentos, resultados e discussão</b>	<b>62</b>
4.1	Bases de dados utilizadas	62
4.1.1	Conjuntos de dados clássicos	62
4.1.2	O conjunto de dados GPS Trajectories	63
4.1.3	Bases de dados do <i>Open data</i>	66
4.2	Métricas e avaliação de modelos	69
4.2.1	Avaliação de agrupamentos	71
4.3	Os experimentos realizados	72
4.3.1	k-NN	72
4.3.2	ID3	73
4.3.3	k-Means	74
4.3.4	DBSCAN	76
4.3.5	Gradiente descendente	78
<b>5</b>	<b>Conclusão</b>	<b>80</b>
	<b>Referências</b>	<b>82</b>

<b>Apêndices</b>	<b>91</b>
<b>APÊNDICE A Manual do ml.js</b>	<b>92</b>
A.1 k-Means . . . . .	92
A.2 DBSCAN . . . . .	93
A.3 k-NN . . . . .	93
A.4 ID3 . . . . .	94
A.5 Gradiente descendente . . . . .	95
A.6 Classes auxiliares . . . . .	96
<b>APÊNDICE B Manual da interface gráfica</b>	<b>102</b>
B.1 Dataset . . . . .	102
B.2 Agrupamento . . . . .	105
B.2.1 k-Means . . . . .	105
B.2.2 DBSCAN . . . . .	107
B.3 Classificação . . . . .	108
B.3.1 k-NN . . . . .	110
B.3.2 ID3 . . . . .	111
B.4 Regressão . . . . .	112
B.4.1 Gradiente descendente . . . . .	113

# 1

## Introdução

A Aprendizagem de Máquina é a subárea da Inteligência Artificial que busca o desenvolvimento de sistemas capazes de adquirir conhecimento de forma automática sem que tenham sido programados explicitamente para isso. Nos últimos anos, a aprendizagem de máquina foi responsável pelo surgimento de carros autônomos (THRUN et al., 2006; MARKOFF, 2010), recursos de reconhecimento de fala (JAITLEY et al., 2012), imagens (GOODFELLOW et al., 2013) e gestos (SOL, 2013), otimizou as buscas na Web (BORGES, 2016), melhorou os sistemas de recomendação (MOTTA, 2016; ALSHEIKH et al., 2014), possibilitou um avanço enorme na compreensão do genoma humano (SPANGLER et al., 2014; HTIKE; WIN, 2013) e em outras áreas como finanças (OLIVEIRA, 2016), agricultura (SANTOS et al., 2013; AL-HIARY et al., 2011), pecuária (SHAHINFAR et al., 2014; CORTEZ et al., 2006), energia (LEITE; CARNEIRO; CARVALHO, 2002; DELBEM; CARVALHO; BRETAS, 2005), medicina (RADIS, 2011; CLEOPHAS; ZWINDERMAN, 2013). Atualmente está tão difundida, que muitas vezes é utilizada sem que haja percepção: *smartphones*, relógios, eletrodomésticos, TVs, tudo está ficando “inteligente” para facilitar as tarefas do cotidiano (LANDAU, 2016; AGRELA, 2016; JUNQUEIRA, 2016; GABASOVA, 2016).

Pode ser desenvolvido a partir de varias linguagens de programação, tais como Python, R, MATLAB, Octave, C, Java, linguagens largamente utilizadas no meio acadêmico. Cada uma dessas, com suas particularidades, apresentam pacotes, bibliotecas, *frameworks* para o desenvolvimento de atividades voltadas à aprendizagem de máquina.

Desses grupos de ferramentas, uma das mais populares é o WEKA. O WEKA (*Waikato Environment for Knowledge Analysis*) é um pacote de *software* para mineração de dados e aprendizado de máquina, escrito na linguagem Java (WEKA, 2015). O WEKA apresenta uma API geral, que pode ser incorporada a outras bibliotecas ou a um código específico. Também contém uma GUI para interagir com arquivos de dados (ver Figura 1) e produzir resultados

visuais. Entretanto, em algumas situações, como por exemplo demonstrações em sala de aula, pequenas interações com usuários de sites, seria adequado o uso de uma linguagem *client-side* de fácil acesso e uso para a maioria de usuários comuns, sem a necessidade de configurações ou pacotes extras. Um exemplo desse tipo de linguagem é o JavaScript.

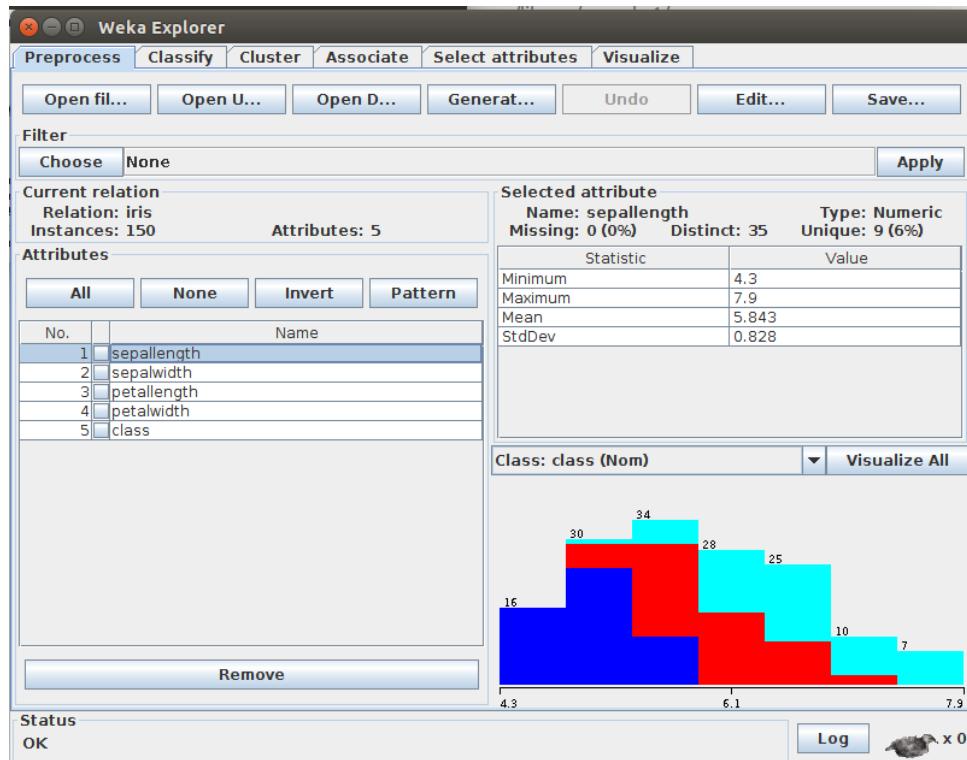


Figura 1 – Exemplo carregamento de arquivo com o WEKA (*Print screen*)

Uma das linguagens de programação mais populares da atualidade. Nos últimos anos, vem aparecendo entre as dez primeiras linguagens mais populares ([TIOBE, 2016](#); [STACKOVERFLOW, 2016](#)) em *rankings* como TIOBE e *Stack Overflow Developer Survey*. Com a evolução da própria linguagem e dos *browsers*, o JavaScript está deixando de ser apenas aquele componente *client-side* responsável pela simples interação com o usuário, validação de dados de entrada de um formulário de cadastro, manipulação do DOM, para atuar como componente principal de grandes aplicações, a exemplo do AngularJS ([ANGULAR.JS, 2016](#)) mantido pelo Google, passando a atuar também no *serve-side* de aplicações com o advento do NodeJS ([NODE.JS, 2016](#)).

Alguns exemplos de bibliotecas e *frameworks* para uso de aprendizado de máquina com JavaScript são listados a seguir.

Encog ([HEATON, 2012](#)) é um *framework* para aprendizado de máquina composto por uma variedade de algoritmos avançados e também possui uma GUI. A Figura 2 mostra um exemplo de utilização de sua versão para JavaScript.

ConvNetJS é uma biblioteca para redes neurais em JavaScript ([KARPATHY, 2014](#)) que permite trabalhar com *Deep Learning* diretamente no *browser* ou no *serve-side* através de No-

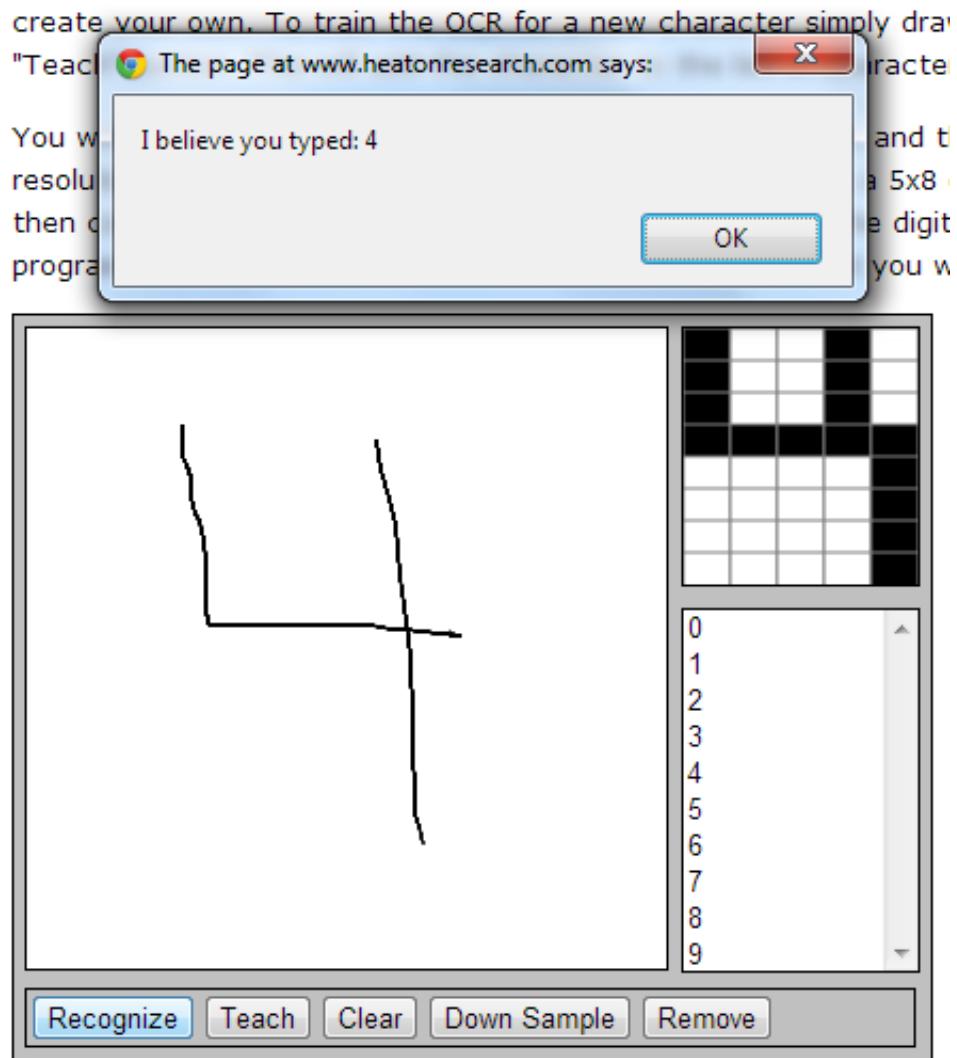


Figura 2 – Exemplo de uso da biblioteca Encog para JavaScript ([HEATON, 2012](#))

deJS.

*Machine Learning in Javascript* ([KANBER, 2012](#)) é uma série de *posts* com discussões e implementações de alguns algoritmos de aprendizado de máquina.

([SIMEONE; FERREIRA, 2014](#)) trazem uma implementação do ID3 de forma interativa, permitindo a manipulação do conjunto de dados diretamente na página por meio de campo de texto e visualização da árvore gerada a partir dos dados passados. A Figura 3 mostra a árvore gerada pelo ID3.

Como foi dito, o uso de aprendizado de máquina está se tornando muito comum entre as mais diversas áreas. Diante disso, e de que alguns desses exemplos são específicos para apenas um tipo de atividade, notou-se a necessidade de uma ferramenta que abrangesse e permitisse a aplicação com fácil obtenção dos resultados, das principais atividades de aprendizado de máquina, classificação, agrupamento e regressão tanto para usuários que têm conhecimento em linguagens de programação, quanto para usuários comuns que queiram apenas extrair conhecimento.

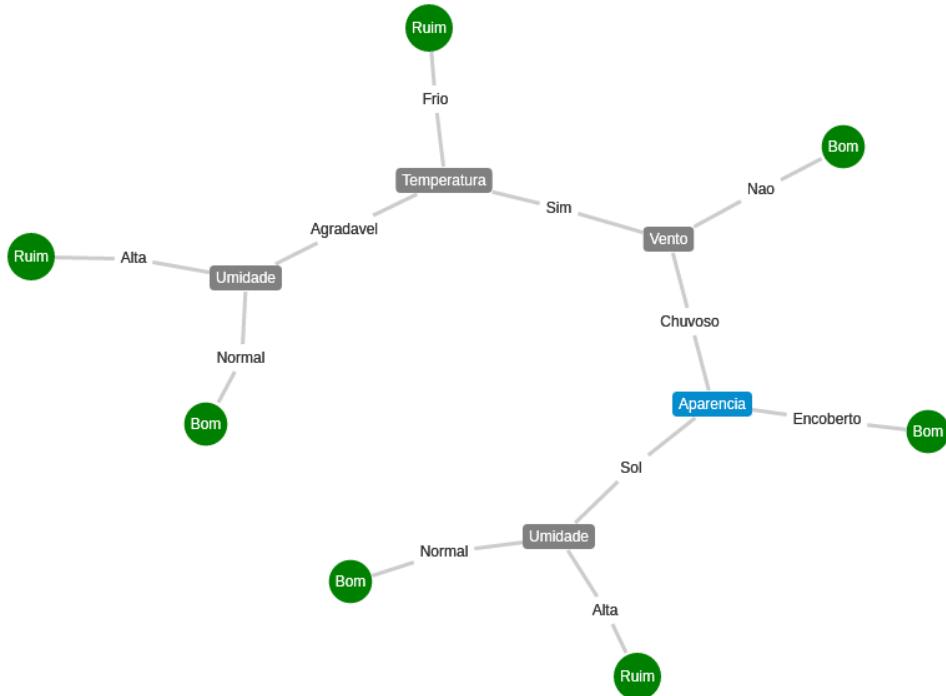


Figura 3 – Exemplo de avore de decisão gerada por ([SIMEONE; FERREIRA, 2014](#))

mento de seus conjuntos de dados afim de solucionar determinados problemas.

O presente trabalho tem como objetivo a construção de uma ferramenta na linguagem de programação JavaScript que auxilie no desenvolvimento de aplicações simples, que executem no navegador Web, que utilizem aprendizado de máquina e também possa ser usada no processo de ensino-aprendizado de Inteligência Artificial e outras disciplinas afins.

A linguagem JavaScript foi escolhida por ser uma linguagem simples e popular. Dessa forma, o *framework* aqui batizado de “ml.js” pode ser usado de forma simples, bastando que se tenha um navegador atualizado. Essa necessidade existe por causa das tecnologias que foram empregadas para seu desenvolvimento, dentre elas SVG, APIs do HTML5 e funções nativas das versões recentes do JavaScript.

O ml.js pode ser usado para as atividades de classificação, agrupamento e regressão necessitando apenas sua inclusão no projeto. O usuário terá acesso aos modelos básicos de cada atividade, podendo assim aplicá-los a seus conjuntos de dados.

Além da versão para usuários desenvolvedores, foi desenvolvida também uma versão gráfica. Por meio da GUI, os usuários podem aplicar de forma rápida e simples os modelos a seus *datasets* e conferir os resultados. Com uma *interface* simples inspirada no *software* WEKA, sobre o qual descreve-se mais detalhes no Capítulo 3, o ml.js permite a visualização do conjunto de dados e dos resultados obtidos por meio de gráficos e informações textuais organizadas.

Este trabalho é organizado da seguinte forma. O Capítulo 2 apresenta conceitos sobre aprendizado de máquina e descreve os modelos implementados no trabalho. O Capítulo 3 apre-

senta detalhes do desenvolvimento do ml.js. O Capítulo 4 descreve os experimentos realizados e os resultados obtidos. Finalmente, a conclusão do trabalho é apresentada no Capítulo 5.

# 2

## Aprendizado de máquina

Imagine o proprietário de uma pequena loja virtual. Nela há uma variedade reduzida não só de livros como também de clientes. Objetivando o aumento no número de vendas, ele passou a formar grupos de clientes que compram livros semelhantes e com características específicas para cada agrupamento, afim de obter um melhor resultado nas operações de *marketing*. O trabalho foi feito a partir dos registros de compras anteriores. Pela pouca quantidade de clientes com vendas efetivadas, a tarefa pode ser realizada de forma simples. Porém, caso esse número fosse mais extenso, ou seja, milhares de clientes com inúmeras vendas efetivadas, esse trabalho teria um grau de dificuldade elevado, impossibilitando a formação desses grupos por um ser humano.

Com o aumento do volume de dados que são gerados por diversos setores e o alto grau de complexidade dos problemas a serem tratados computacionalmente, sobretudo problemas do mundo real, tornou-se necessário o desenvolvimento de programas mais sofisticados, que conseguissem resolver determinados problemas sem que houvesse tanta intervenção humana. Na programação explícita, a depender do problema, é impossível considerar todas as variações, principalmente tratando-se de problemas do mundo real que são muito imprevisíveis. A solução então, é deixar que as máquinas aprendam sozinhas a partir de experiências passadas. A essa técnica dá-se o nome de Aprendizado de Máquina (AM) ([FACELI et al., 2011](#)).

### 2.1 Paradigmas de aprendizagem

São várias as definições de AM na literatura. Uma dessas data de 1959, do pioneiro nas áreas de inteligência artificial, jogos computacionais e aprendizagem de máquina, Arthur Samuel: "*O campo de estudo que dá a computadores a capacidade de aprender, sem ser programado de forma explícita*" ([SIMON, 2013](#), p. 89, tradução nossa). Outra definição apresentada por Tom Mitchell: "*Um programa de computador aprende com uma experiência E relativo a*

uma tarefa  $T$  e uma medida de desempenho  $P$ , se o seu desempenho em  $T$  medido por  $P$  melhora com a experiência  $E$ " (MITCHELL, 1997, p. 2, tradução nossa).

A Figura 4 apresenta uma representação do processo de AM. A partir de um conjunto de pares de entrada e saída, o modelo deve encontrar uma função  $g$  que mais se aproxime da função alvo  $f : x \rightarrow y$  que prevê a saída para novas entradas.

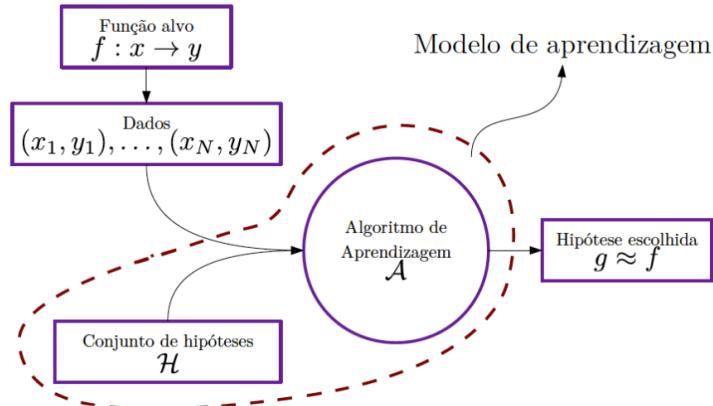


Figura 4 – Representação do processo de AM (MACEDO, 2014)

Ao contrário do que acontece na programação tradicional, onde o papel do programador é modelar uma possível solução para o problema em questão, no caso do AM a máquina é que será a responsável por encontrar essa solução a partir de um conjunto de experiências que são passadas à ela. Essa técnica está por trás de grande parte dos sistemas inteligentes presentes no dia a dia. Em virtude do AM, o Facebook reconhece as pessoas nas fotos (Figura 5a), o Kinect reconhece gestos (Figura 5c), o aplicativo Google Translate consegue traduzir texto a partir de imagens (Figura 5b), entre outros.

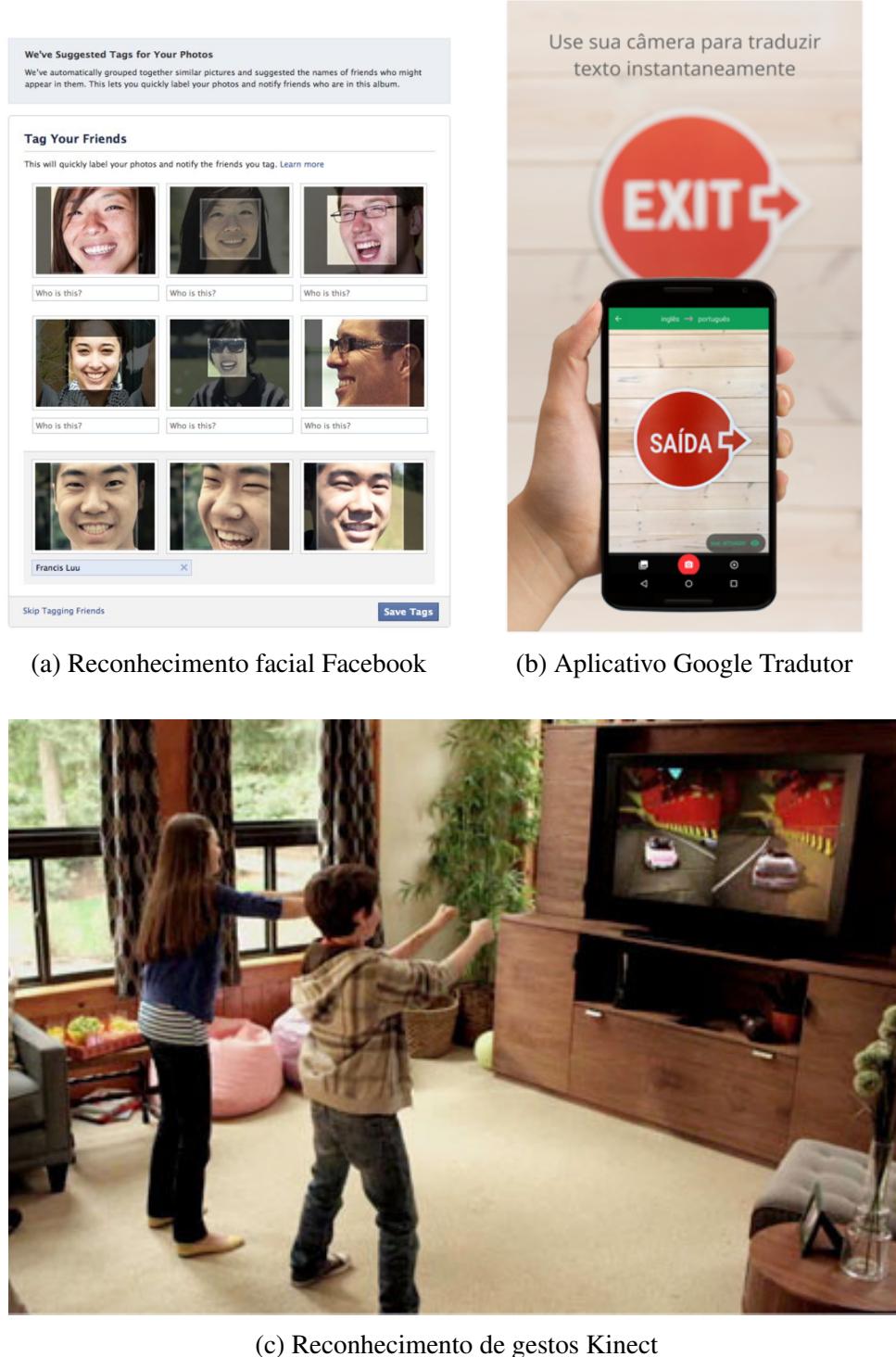


Figura 5 – Exemplos de aplicações de AM em problemas do mundo real - fontes Facebook, Google Play e Microsoft respectivamente

Um dos componentes mais importantes no processo de AM são os dados. É através deles que a máquina irá aprender. A eficiência dessa aprendizagem depende da capacidade de extração de informações dos dados que são passados à máquina, relevantes para resolver o problema que se deseja tratar. Quanto mais informações se têm, melhor o reconhecimento de padrões, maior diversidade de exemplos, mais situações podem ser aprendidas e previstas.

O desenvolvimento do AM não pode acontecer sem os dados. Felizmente, nunca se produziu tantos dados. Cada *like* no Facebook, cada novo twitte, vídeos, fotos, texto que são publicados em sites ou blogs, alimentam esses sistemas que estão em constante aprendizado.

O AM pode ser comparado com o aprendizado humano com relação às experiências: não é mostrado à máquina qual o caminho para se chegar à solução do problema, ela o experimentará várias vezes e criará o próprio modelo da solução, de uma forma bem genérica. Isso significa reconhecer os padrões presentes nos dados e quanto mais dados, mais experiências a máquina terá para experimentar.

A Figura 6 apresenta uma hierarquia de aprendizado de máquina. No topo da hierarquia aparece o aprendizado indutivo, processo pelo qual são realizadas as generalizações a partir dos dados. Tem-se em seguida os tipos de aprendizado supervisionado (preditivo), não supervisionado (descritivo) e por reforço. As tarefas supervisionadas se distinguem pelo tipo dos rótulos dos dados: discretos, no caso de classificação; e contínuo, no caso de regressão. As tarefas descritivas são genericamente divididas em: agrupamento, em que os dados são agrupados de acordo com sua similaridade; sumarização, cujo objetivo é encontrar uma descrição simples e compacta para um conjunto de dados; e associação, que consiste em encontrar padrões frequentes de associações entre atributos de um conjunto de dados (FACELI et al., 2011). Na aprendizagem por reforço, o algoritmo aprende a partir de uma série de reforços (recompensas ou punições).

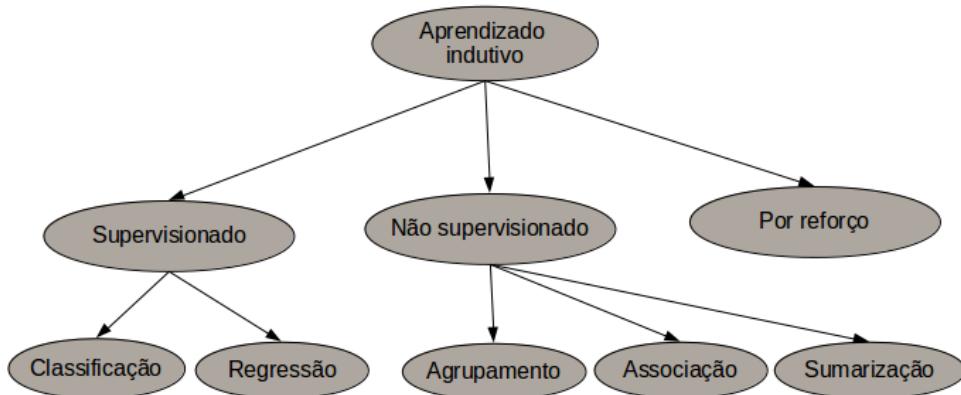


Figura 6 – Hierarquia de aprendizado - adaptado de (FACELI et al., 2011)

Com exceção da associação e sumarização, as demais tarefas são brevemente abordadas neste capítulo.

### 2.1.1 Aprendizagem supervisionada

A Figura 7 mostra um gráfico de um conjunto de dados que contém informações sobre os valores e tamanhos de determinados imóveis. No eixo horizontal é representado o tamanho do imóvel em  $m^2$  e no eixo vertical o preço em milhares.

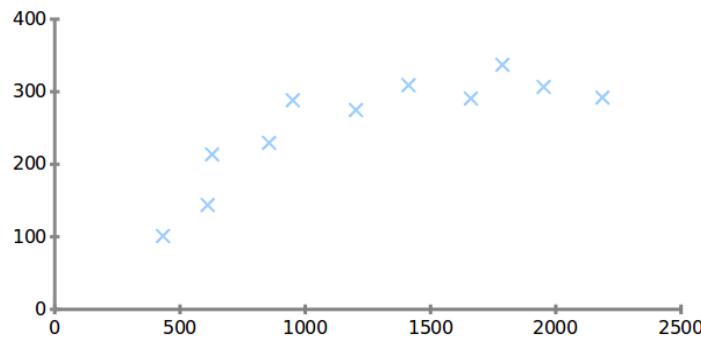


Figura 7 – Conjunto de dados de exemplo preço *versus* tamanho de imóveis (NG, 2012a)

Supondo que alguém tenha um imóvel de  $750\text{ m}^2$  à venda e queira saber quanto pode conseguir por esse imóvel. Com base nesse conjunto de dados, como a aprendizagem de máquina pode ajudar nessa questão? Uma coisa que um algoritmo de AM pode fazer é encontrar uma reta através dos dados, como na Figura 8a e, baseando-se nisso, o preço seria aproximadamente de 150.000. Mas talvez, essa não seja a melhor forma de generalização para esse conjunto de dados. Através de outra técnica de AM, é possível fazer uma aproximação por meio de uma função quadrática, por exemplo, como ilustrado na Figura 8b. Agora, com base nessa nova aproximação a previsão do preço do imóvel é mais ou menos de 200.000. Esses são dois exemplos de aprendizagem de máquina supervisionada, mais especificamente um problema de regressão. A questão de qual aproximação tem o melhor resultado será discutida mais adiante.

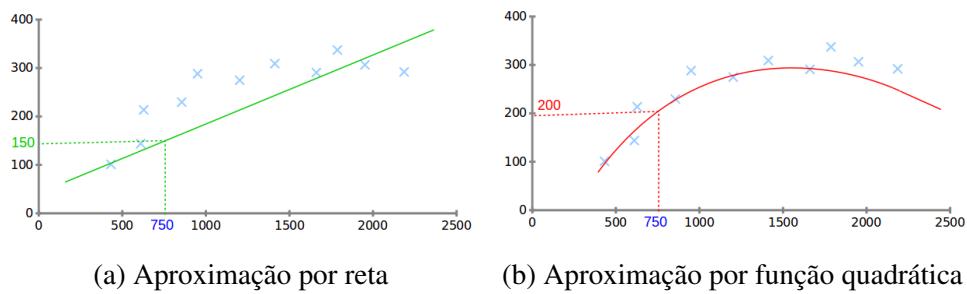


Figura 8 – Exemplo de aproximação de funções para conjunto de dados - adaptado de (NG, 2012a)

O termo aprendizado supervisionado se refere ao fato de que, o conjunto de dados passado ao algoritmo contém as “respostas certas”. Nesse caso, para cada exemplo do conjunto de dados de imóveis, o preço correto foi informado previamente. A partir de um conjunto de dados rotulados, a máquina aprende o padrão que generaliza o problema, sendo capaz de solucionar novas instâncias desse mesmo problema. Esse tipo de aprendizagem é aplicada a problemas de classificação e regressão.

No caso da atividade de classificação, o conjunto de dados de entrada é dividido em duas ou mais classes, e o objetivo do algoritmo é produzir um modelo ou uma função (hipótese)

que generalize ao máximo o conjunto de dados e possa prever a saída de qualquer entrada. A saída nesse caso é um conjunto finito de valores, por exemplo, *ensolarado*, *nublado* ou *chuva*, e quando existirem apenas dois valores, será chamado de classificador booleano ou binário.

Na classificação, deve-se levar em consideração a separabilidade. Problemas linearmente separáveis são problemas de classificação de padrões que podem ser resolvidos a partir de uma superfície de decisão linear (Figura 9a). Nesse caso, as duas classes podem ser separadas por uma linha reta. Já os problemas não linearmente separáveis, não é possível fazer a separação dos padrões através de uma superfície de decisão linear (Figura 9b).

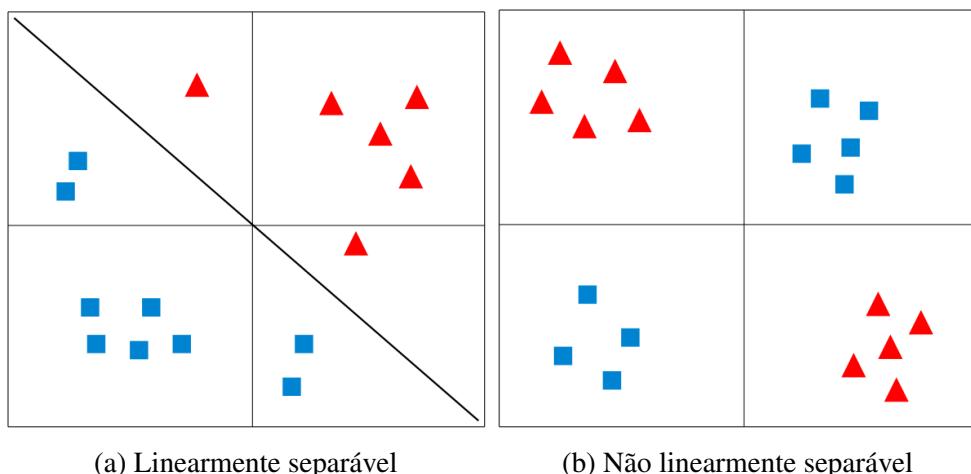


Figura 9 – Separatividade linear - (autoria própria)

Para problemas de regressão, ao contrário da classificação, a saída do algoritmo é contínua em vez de discreta. Por exemplo, temperatura ou valor de um imóvel. A forma mais simples de regressão é com uma função linear de uma única variável, onde a saída do algoritmo pode ser visto como uma linha reta. A Figura 10, mostra uma tentativa de ajuste de uma função na forma  $y = w_1x + w_0$ . A linha amarela representa a melhor aproximação da função.

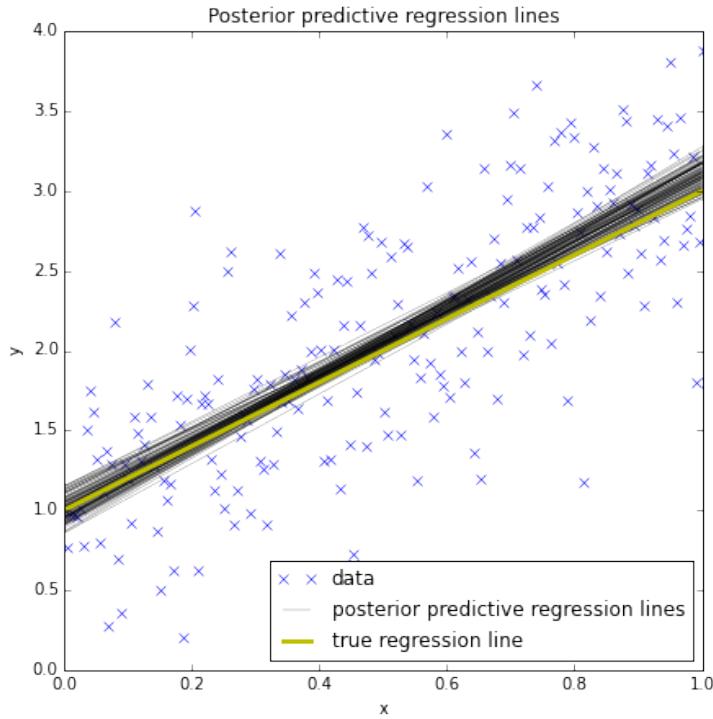


Figura 10 – Exemplo de regressão linear (WIECKI, 2015)

### 2.1.2 Aprendizagem não supervisionada

Diferentemente da aprendizagem supervisionada, na abordagem não supervisionada, o conjunto de dados que é passado à máquina contém exemplos não rotulados. Nada é informado sobre do que se trata cada exemplo. O objetivo é encontrar uma estrutura de *clusters* (grupos) nos dados onde os exemplos que pertencem ao mesmo grupo compartilham alguma característica ou propriedade em comum. Como os *clusters* são inicialmente desconhecidos, a depender do conjunto de dados, o algoritmo pode encontrar similaridade entre os objetos que de inicio pareçam não ter características em comum. Esse tipo de atividade é chamada de agrupamento.

Por exemplo, observa-se na Figura 11a um conjunto de objetos. É possível agrupá-los de diferentes maneiras como ilustram as Figuras 11b, 11c e 11d. Nessas figura são mostradas três diferentes formas de agrupamento, onde cada uma delas levou em consideração uma ou mais características dos objetos. Na Figura 11b, a forma geométrica é o aspecto dominante do grupo. Para os dois grupos formados na Figura 11c foi levado em consideração o preenchimento dos objetos. Já em 11d, onde são apresentados quatro *clusters*, a divisão levou em consideração uma combinação dessas duas características. Cada um desses agrupamentos é uma estrutura ou modelo que descreve os dados e poderiam ter sido gerados por uma técnica de agrupamento.

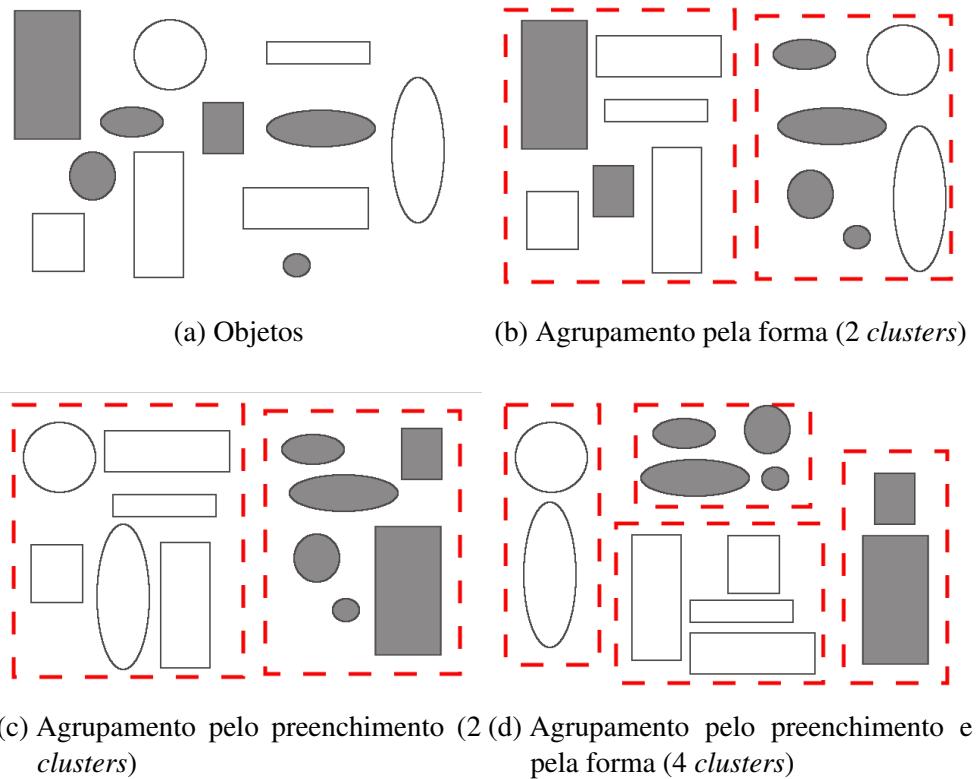


Figura 11 – Exemplo de agrupamentos de objetos - adaptado de (FACELI et al., 2011)

### **2.1.3 Aprendizagem por reforço**

Esse tipo de aprendizagem é parecida com a aprendizagem não supervisionado porque os dados também não são rotulados, mas existe um tipo de *feedback* na forma de recompensa caso o sistema execute ações positivas, ou punições, caso execute ações negativas (RUSSELL; NORVIG, 2013). Por exemplo, uma criança ganhou um novo jogo de *video game* e começou a jogar sem ter lido as regras e as instruções de como jogar. O que ela vai fazer é experimentar uma ação ou um conjunto de ações e acabar percebendo que em alguns casos, consegue aumentar a pontuação ou mesmo ganhar um bônus, ou seja, recompensa (Figura 12a); e em outros casos, perde-se a pontuação (Figura 12b) ou até mesmo acontece um *game over*, punição. Quanto mais joga mais consegue entender as regras do jogo, mesmo que o manual nunca tenha sido lido. A criança irá aprender a jogar apenas com suas próprias experiências a partir de *feedbacks* de recompensa e punição. O aprendizado por reforço funciona dessa forma.

Em alguns problemas o padrão para se chegar a solução é desconhecido. No entanto, é possível identificar quando um resultado é bom ou ruim. Passando isso como *feedback* para um algoritmo de aprendizado por reforço, ele será capaz de encontrar o padrão das ações e chegar à solução do problema proposto.



(a) Recompensa pelo atalho

(b) Punição pela colisão

Figura 12 – Exemplo aprendizagem por reforço game *Need For Speed Underground* - Gamespot

## 2.2 Modelos de aprendizado de máquina

Esta seção, apresenta de forma resumida alguns algoritmos de aprendizagem de máquina. Na Seção 2.3 será descrito mais detalhes dos modelos implementados neste trabalho.

Pode-se agrupar os algoritmos de AM de várias maneiras, a mais comum encontrada na literatura especializada é o agrupamento baseado no estilo de aprendizagem (supervisionada, não supervisionada, por reforço). Uma outra maneira é realizar um agrupamento baseado na semelhança da maneira como funcionam, métodos baseados em árvore de decisão e inspirados em redes neurais, por exemplo. Com base em (BROWNLEE, 2013), alguns grupos e algoritmos são mostrados a seguir.

- **Algoritmos baseados em regressão**

Nos modelos de aprendizagem baseados em regressão, o algoritmo cria modelo que apresenta uma relação entre as variáveis. Esse modelo é refinado iterativamente usando uma medida de erro nas previsões feitas. Pode-se citar como exemplos Ordinary Least Squares Regression (OLSR) (BENOIT, 2010), Linear Regression (NG, 2012b), Logistic Regression (DREISEITL; OHNO-MACHADO, 2002), Multivariate Adaptive Regression Splines (MARS) (FRIEDMAN, 1991), Locally Estimated Scatterplot Smoothing (LOESS) (JACOBY, 2000).

- **Algoritmos baseados em instâncias**

O algoritmo treina a partir do conjunto de dados de treinamento que lhe é passado como entrada. Por meio de medidas de similaridade, faz uma previsão para novos dados de entrada a fim de encontrar a melhor correspondência entre o novo dado e seu conjunto de treinamento. Alguns dos representantes desse grupo são k-Nearest Neighbour (kNN) (PETERSON, 2009), Learning Vector Quantization (LVQ) (KOHONEN, 1997), Self-Organizing Map (SOM) (KOHONEN, 1997), Locally Weighted Learning (LWL) (AT-KESON; MOORE; SCHAAL, 1997).

- **Algoritmos de agrupamento**

Também conhecido como clusterização, o algoritmo deve dividir um conjunto de dados de entrada em grupos de máxima semelhança entre os dados. Os grupos não são conhecidos antecipadamente. Uma abordagem desse modelo é baseada em centroide, o elemento central do grupo é usado como base para criação do grupo. Exemplos de algoritmos de clusterização são K-Means ([JAIN, 2008](#)), K-Medians ([BERKHIN, 2006](#)), Expectation Maximisation (EM) ([GUPTA; CHEN, 2011](#)), Density-Based Spatial Clustering of Applications With Noise (DBSCAN) ([ESTER et al., 1996](#)), Hierarchical Clustering ([MATTEUCCI, 2003](#)).

- **Algoritmos bayesianos**

Esse modelo de aprendizagem também conhecido como modelo de Bayes ingênuo ([MILLIDIÚ, 2006](#)), assume que o efeito do valor de um atributo sobre uma determinada classe é independente dos valores dos outros atributos, deixando os cálculos mais simples e mais rápidos. Pode-se citar como exemplos Naive Bayes ([ZHANG, 2004](#)), Gaussian Naive Bayes ([LOWD; DOMINGOS, 2005](#)), Multinomial Naive Bayes ([TELECOMMUNICATIONS; METSIS, 2006](#)), Averaged One-Dependence Estimators (AODE) ([WEBB; BOUGHTON; WANG, 2005](#)), Bayesian Belief Network (BBN) ([KRIEG, 2001](#)), Bayesian Network (BN) ([HECKERMAN; GEIGER; CHICKERING, 1995](#)).

- **Algoritmos baseados em árvore de decisão**

Um dos modelos mais práticos e simples para inferência indutiva, uma árvore de decisão representa uma função que toma como entrada um vetor de atributos e retorna um único valor de saída, uma decisão. Esse modelo é bem flexível, podendo lidar com relações não lineares entre atributos e classes, dados em diferentes escalas e medidas. Outra característica importante é que sua representação parece ser muito natural para o entendimento: sequências de "*se, então*" facilmente interpretadas. Exemplos de algoritmos de árvore de decisão são Iterative Dichotomiser 3 (ID3) ([QUINLAN, 1986](#)), C4.5 e C5.0 ([QUINLAN, 2014](#)), Chi-squared Automatic Interaction Detection (CHAID) ([WILKINSON, 1992](#)), Classification and Regression Tree (CART) ([BREIMAN et al., 1984](#)), Decision Stump ([IBA; LANGLEY, 1992](#)), Conditional Decision Trees (CTree) ([HOTHORN; HORNIK; ZEILEIS, 2006](#)).

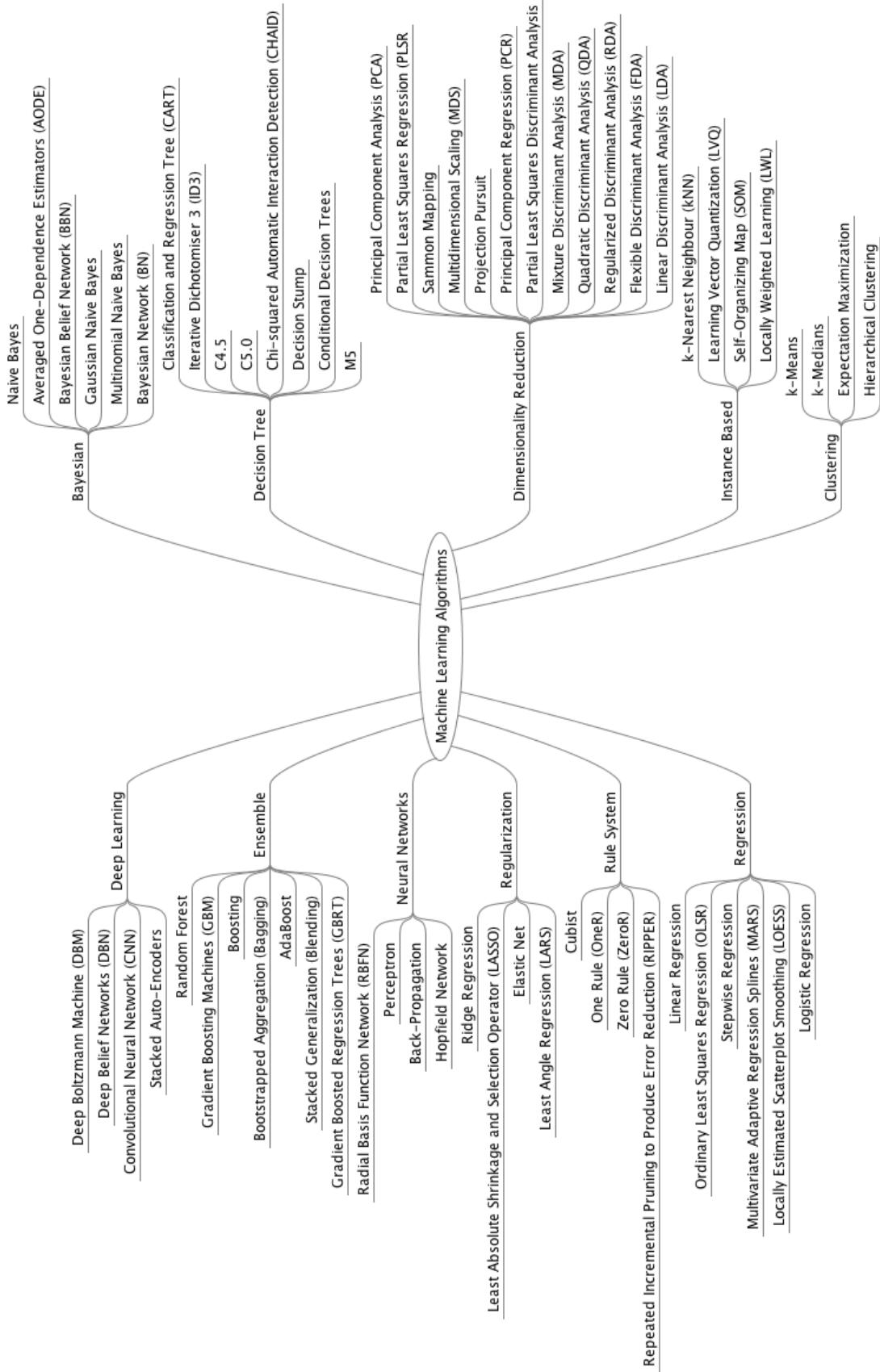
- **Algoritmos baseados em redes neurais artificiais**

Redes neurais artificiais (RNAs) representam um modelo matemático inspirado no sistema nervoso central de organismos inteligentes, que são capazes de adquirir conhecimento através de experiências. Representantes de redes neurais são Perceptron ([FREUND; SCHAPIRE, 1999](#)), Back-Propagation ([SCHMIDHUBER, 2015](#)), Hopfield Network ([SATHASIVAM; ABDULLAH, 2008](#)), Radial Basis Function Network (RBFN) ([SCHWENKER; KESTLER; PALM, 2001](#)).

- **Agregação de algoritmos**

Classificadores considerados mais fracos são treinados separadamente e as decisões são combinadas em um único classificador. Técnicas empregadas são Boosting ([SCHAPIRE, 2002](#)), Bootstrapped Aggregation (Bagging) ([BREIMAN, 1996](#)), AdaBoost ([FREUND; SCHAPIRE, 1995](#)), Random Forest ([HO, 1995](#)).

A Figura 13 traz um mapa mental dos algoritmos de aprendizagem de máquina.

Figura 13 – Mapa mental dos algoritmos de aprendizagem de máquina ([BROWNLEE, 2013](#))

## 2.3 Os modelos selecionados

A seguir são descritos os modelos selecionados para o presente trabalho. Para a atividade de classificação foram selecionados os modelos k-NN e ID3. Para o agrupamento, k-Means e DBSCAN. Para a atividade de regressão, o modelo Gradiente descendente para regressão linear.

### 2.3.1 k-NN

O *k*-Vizinhos mais próximos (do inglês, *k*-Nearest Neighbors) é um método para classificação baseado em distância. Esse classificador representa cada exemplo como um ponto no espaço  $d$ -dimensional, onde  $d$  é o número de atributos do exemplo. Para classificar um novo exemplo, são levados em consideração os objetos do conjunto de treinamento que estão próximos a ele, que são chamados de vizinhança

O princípio básico pode ser visto como o ditado popular "*Diga-me com quem andas e te direi quem és!*". Dado um novo exemplo de teste, primeiro é calculada sua distância (ou similaridade) com o resto dos pontos de dados do conjunto de treinamento. Os  $k$  exemplos mais próximos formarão a vizinhança, onde  $k$  é um parâmetro informado pelo usuário. O exemplo de teste é classificado baseado na classe majoritária dos seus vizinhos mais próximos. Um resumo de alto nível do k-NN é dado no Algoritmo 1.

---

#### Algoritmo 1: Algoritmo básico para o modelo k-NN

---

```

1 início
2   para cada exemplo  $x_i \in D$  faça
3     calcule a distância  $d(x_i, y)$ ;
4   fim
5   Ordene de forma crescente as distâncias  $d(x_i, y)$ ;
6   Selecione os  $k$  vizinhos mais próximos;
7   Atribua a  $y$  a classe com maior frequência;
8 fim
```

---

Na Figura 14, está ilustrada a vizinhança do ponto situado no centro de cada círculo para os valores de  $k$  igual a 1, 3 e 5. Em 14a o ponto é atribuído à classe azul, pois seu vizinho mais próximo pertence a essa classe. Quando o valor de  $k$  for três (Figura 14b), a vizinhança do ponto contém dois exemplos da classe vermelha e um da classe azul, portanto é classificado como vermelho por votação da maioria. O mesmo acontece na Figura 14c, com três exemplos vermelhos e dois azuis.

Assim como outros modelos baseados em distância, o k-NN também depende de uma boa função de distância para obter bons resultados. O computo da proximidade dos pontos de dados, pode ser feito utilizando-se diversas métricas. A mais usual é a distância Euclidiana para atributos numéricos e a distância de Hamming para atributos categóricos ([WILSON; MARTI-](#)

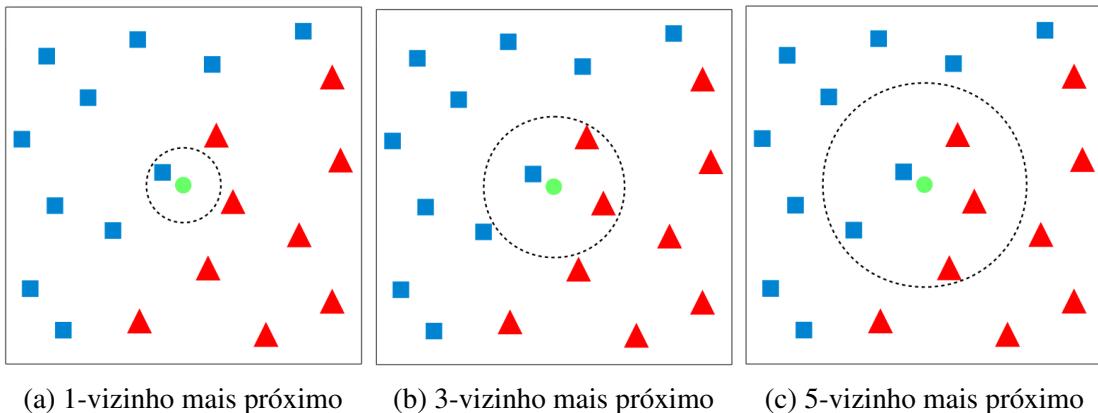


Figura 14 – Os 1, 3 e 5 vizinhos mais próximos do ponto no centro dos círculos - adaptado de (TAN; STEINBACH; KUMAR, 2009)

NEZ, 1997). A Equação 2.1 define a distância de Hamming.

$$H(x, y) = \sum_{a=1}^n h_a(x_a, y_a) \quad h_a = \begin{cases} 0 & \text{se } x_i = y_i \\ 1 & \text{se } x_i \neq y_i \end{cases} \quad (2.1)$$

A escolha da classe é efetuada de maneira diferente em problemas de classificação e regressão. No caso da classificação, é usado uma abordagem de votação majoritária, onde cada vizinho possui igual impacto sobre a escolha da classe. Formalmente esse processo equivale à moda.

Em problemas de regressão, a depender da função de custo utilizada, podem ser usadas duas estratégias. Se a função de custo for minimizar o erro quadrático, a média dos valores dos  $k$  vizinhos deve ser utilizada. Já se a função de custo a ser minimizada for o desvio absoluto, a mediana deverá ser utilizada (FACELI et al., 2011).

De acordo com (HALL; PARK; SAMWORTH, 2008), k-NN é muito simples e intuitivo. No entanto, o que atrapalha sua aplicação é a falta de conhecimento sobre suas propriedades, sobretudo, a maneira que é influenciado pela escolha do valor de  $k$ . Se for escolhido um valor muito pequeno, então k-NN pode estar sujeito à *overfitting* por causa dos ruídos nos dados de treinamento. Por outro lado, se  $k$  for muito grande, a classificação poderá ser feita de maneira errada, por que a lista vizinhos mais próximos pode conter pontos de dados que estejam localizados longe da vizinhança da instância de teste. Ou seja, o valor de  $k$  deve ser grande o suficiente para evitar *overfitting*, mas pequeno o suficiente para evitar a simplificação do conjunto de dados (JOHNSON, 2013). A Figura 15 ilustra esse fato.

Como foi mostrado, k-NN é sensível à escolha do valor de  $k$ , quando usa-se a *moda*, pois cada vizinho tem o mesmo impacto sobre a classificação. Uma maneira de diminuir o impacto de  $k$  é utilizando uma estratégia de pesos para cada vizinho mais próximo do exemplo de teste, onde pontos mais próximos possuem maior peso.

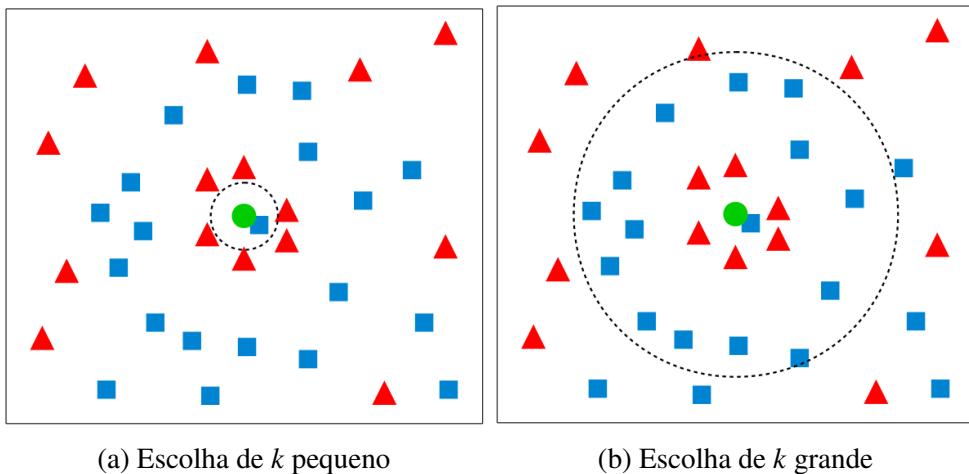


Figura 15 – Impacto da escolha do valor de  $k$  para classificação de vizinhos mais próximos (autoria própria)

Pode-se citar como vantagens do modelo k-NN, sua simplicidade, a possibilidade de ser aplicado mesmo a problemas complexos e ser um modelo incremental, pois para um novo treinamento basta armazenar os novos exemplos na memória. Por ser um modelo *lazy*, toda a computação é adiada para a fase de classificação, que requer calcular a distância do exemplo de teste a todos os exemplos de treinamento, o que pode ser custoso se o número de exemplos for grande.

Assim como os demais modelos baseados em distância, k-NN tem seu desempenho afetado pela medida ou função de distância utilizada. Por exemplo, para a classificação de um determinado grupo de pessoas, de acordo com os atributos peso (medido em  $Kg$ ) e altura (medida em metros). O atributo peso tem uma variabilidade alta, indo de 60 a 140  $Kg$ , por exemplo, enquanto que a altura das pessoas varia de 1,2 a 1,85 metros. Se a escala dos atributos não for levada em consideração, a medida de proximidade pode ser dominada pelo atributo peso (VALE, 2005).

Outro problema que atrapalha o desempenho do k-NN é a maldição da dimensionalidade. Quando a dimensionalidade dos dados aumenta, a análise se torna significativamente mais difícil e os dados ficam mais dispersos no espaço que ocupam, os pontos que podem ser similares podem ter distâncias muito grandes (BROWNLEE, 2016). O modelo pode não conseguir classificar de forma confiável todos os objetos possíveis do conjunto de dados.

### 2.3.2 ID3

O ID3 (*Iterative Dichotomiser 3*) é um modelo baseado em árvore de decisão proposto por Ross Quinlan (QUINLAN, 1986). Possui uma abordagem *top-down* para a construção da árvore a partir da raiz. Alguns detalhes sobre árvores de decisão e sua forma de aprendizado precisam ser descritos inicialmente.

Árvores de decisão são construídas utilizando a estratégia de dividir para conquistar. Problemas maiores são divididos em problemas mais simples, aos quais recursivamente é aplicado a mesma estratégia (GARCIA, 2003).

Modelos de árvore de decisão representam uma função que tem como entrada um vetor de atributos e retorna uma decisão. Os valores de entrada e saída podem ser discretos ou contínuos (POZZER, 2006).

Na representação de um problema por árvore de decisão, cada nó da árvore contém um teste num atributo, cada ramo corresponde a um possível valor do atributo, cada nó folha atribui uma classificação (Figura 16). Para classificar um novo exemplo, a partir da raiz da árvore basta seguir as ramificações apropriadas até alcançar um nó folha. Cada percurso deste corresponde a uma regra de classificação, uma decisão.

Formalmente, uma árvore de decisão é um grafo acíclico direcionado.

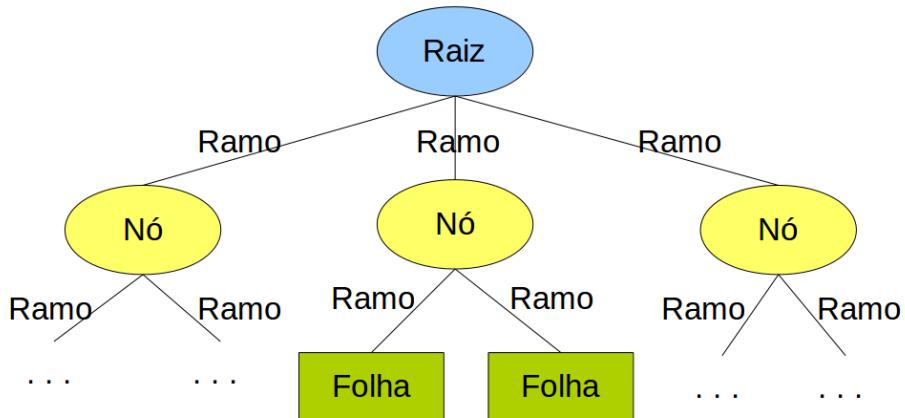


Figura 16 – Representação de árvore de decisão (autoria própria)

Uma árvore de decisão representa um classificador não linear e consegue resolver, decidir, problemas não linearmente separáveis. A Figura 17a representa um espaço de instâncias de um problema que não pode ser resolvido com apenas um classificador linear. A Figura 17b representa uma árvore de decisão que consegue classificar corretamente as instâncias para o problema representado.

Algumas questões estão ligadas à construção da árvore de decisão. O ponto de divisão (decisão) que deve ser escolhido é uma questão muito importante. O critério de divisão dos dados deve fazer com que os novos subconjuntos formados sejam mais homogêneos, menos impuros. O atributo a ser escolhido deve dividir o conjunto de treinamento de forma que os novos subconjuntos sejam formados por exemplos de apenas uma classe. Um atributo que na divisão, mantém as mesmas proporções de classes em todas as partições é inútil (GAMA, 2002). Por exemplo, um conjunto com distribuição de classe (0,1) possui impureza zero. Por outro lado, um conjunto que apresenta uma distribuição uniforme possui o maior grau de impureza.

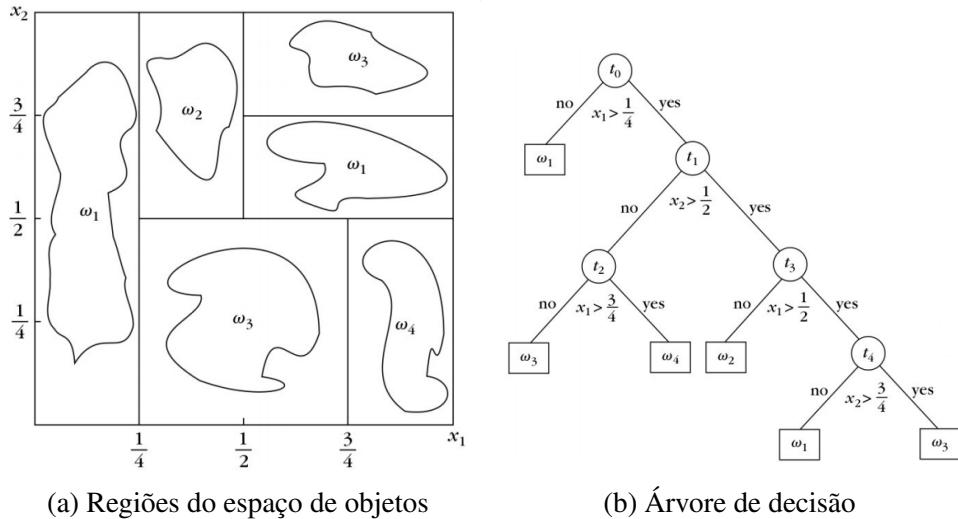


Figura 17 – Exemplo árvore de decisão e as regiões de decisão no espaço de objetos ([MACEDO, 2014](#))

As métricas para selecionar a melhor divisão são muitas vezes baseadas no grau de impureza dos nós filhos. Uma medida de impureza comumente usada é a entropia, que é uma medida da incerteza de uma variável aleatória ([FACELI et al., 2011](#)). A entropia é definida pela Equação 2.2.

$$Entropia(t) = - \sum_{i=1}^k P(i|t) \log_2 P(i|t) \quad (2.2)$$

Sendo  $P(i|t)$  a fração de registros que pertencem à classe  $i$  de um determinado nó  $t$  e  $k$  denota o número de rótulos de classes diferentes. A Figura 18 mostra o gráfico da entropia para um problema de classificação binária.  $P$  representa a fração de registros que pertencem a uma das duas classes, observe que quando  $P$  se aproxima de 0.5, ou seja, uma distribuição uniforme, o valor da entropia se aproxima do seu valor máximo 1.

A exemplo pode ser citado o lançamento de uma moeda. No caso de uma moeda honesta, existe igual possibilidade do resultado ser cara ou coroa, 0 ou 1, o que corresponde a um bit de entropia. Esse resultado é mostrado na Equação 2.3. Como a entropia é uma medida de impureza, a forma  $I(\cdot)$  a representará, a partir deste ponto. Considerando que a moeda foi alterada e dá cara em 75% dos lançamentos, nota-se que sua entropia mudou. O cálculo da entropia para a moeda adulterada pode ser visto na Equação 2.4.

$$I(moedaHonest) = -P(cara) \log_2 P(cara) - P(coroa) \log_2 P(coroa) \quad (2.3)$$

$$I(moedaHonest) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5$$

$$I(moedaHonest) = 1 \text{ bit}$$

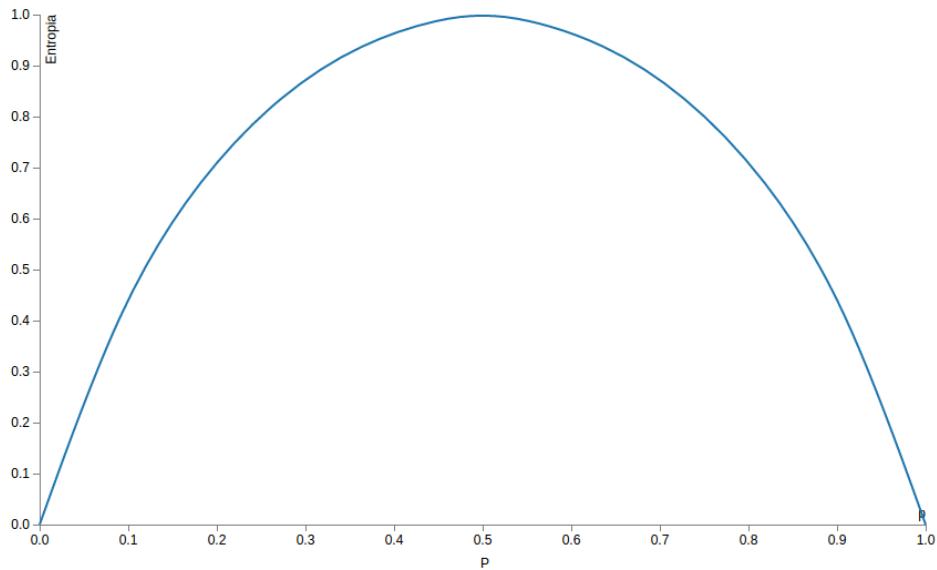


Figura 18 – Gráfico da entropia (autoria própria)

Verifica-se que a entropia da moeda adulterada tem um valor menor que o da moeda honesta, ou seja, tem uma menor incerteza.

$$I(\text{moedaAdulterada}) = -P(\text{cara}) \log_2 P(\text{cara}) - P(\text{coroa}) \log_2 P(\text{coroa}) \quad (2.4)$$

$$I(\text{moedaAdulterada}) = -0.75 \log_2 0.75 - 0.25 \log_2 0.25$$

$$I(\text{moedaAdulterada}) = 0.811 \text{ bit}$$

A ideia do ID3 é diminuir gradativamente a entropia de todo o conjunto de treinamento escolhendo de forma apropriada o próximo atributo a ser testado na montagem da árvore. A diminuição da entropia é conhecida como ganho de informação. Modelos baseados em árvores de decisão muitas vezes escolhem como condição de teste maximizar o ganho de informação. O ID3 segue esses modelos. O ganho de informação está definido na Equação 2.5.

$$\Delta I(t) = I(\text{pai}) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j) \quad (2.5)$$

Onde  $N$  é número total de registros do nó pai,  $k$  representa o número de valores dos atributos, e  $N(v_j)$  é o número de registros associados ao nó filho  $v_j$ .

ID3 está ilustrado no Algoritmo 2.

---

### **Algoritmo 2:** Algoritmo ID3

---

**Entrada:** Exemplos (exemplos de treinamento)  
 Atributo-alvo (atributo cujo o valor deve ser redito pela árvore)  
 Atributos (lista dos atributos que podem ser testados pela árvore de decisão)

**Saída:** Árvore de decisão

```

1 início
2   Crie um nó Raiz para a árvore;
3   se todos os Exemplos são positivos então
4     retorna Raiz da árvore com rótulo = sim;
5   fim
6   se todos os Exemplos são negativos então
7     retorna Raiz da árvore com rótulo = não;
8   fim
9   se Atributos for vazio então
10    retorna Raiz da árvore com rótulo = a moda de Atributo-alvo em
        Exemplos;
11   senão
12     A ← um atributo de A com maior ganho de informação;
13     Raiz ← A;
14     para cada  $v_i \in A$  faça
15       Adicione um novo ramo a Raiz correspondente a  $A = v_i$ ;
16       Exemplos $v_i$  ← subconjunto de Exemplos que têm valor  $v_i$  para A;
17       se Exemplos $v_i$  for vazio então
18         Acescente na extremidade do ramo um nó folha com rótulo = a
            moda de Atributo-alvo em Exemplos;
19       senão
20         Acrescente na extremidade do ramo a sub árvore ID3(Exemplos $v_i$ ,
            Atributo-alvo, Atributos - {A});
21       fim
22     fim
23     retorna Raiz;
24   fim
25 fim
```

---

Pode-se usar como exemplo um final de semana planejado, usando dados tais como clima, a presença de visitante, e dinheiro disponível, como apresentado na Tabela 1. A partir desses dados é possível construir uma árvore de decisão, que dado um conjunto de atributos retorne uma decisão com umas das seguintes opções: jogar tênis, ficar em casa, ir ao cinema ou

ao shopping.

Tabela 1 – Exemplo fim de semana adaptado de (COLTON, 2004)

Clima	Presença de visitantes	Dinheiro	Destino
Sol	Sim	Sim	Cinema
Sol	Não	Sim	Tênis
Vento	Sim	Sim	Cinema
Chuva	Sim	Não	Cinema
Chuva	Não	Sim	Casa
Chuva	Sim	Não	Cinema
Vento	Não	Não	Cinema
Vento	Não	Sim	Shopping
Vento	Sim	Sim	Cinema
Sol	Não	Sim	Tênis

O primeiro passo é descobrir qual atributo será colocado na raiz da árvore. Para isso, é necessário calcular a entropia do atributo alvo ou classe, que no exemplo é o atributo *Destino* (Equação 2.6).

$$\begin{aligned}
 I(\text{Destino}) &= -\left(\frac{6}{10}\right) \times \log_2\left(\frac{6}{10}\right) - \left(\frac{2}{10}\right) \times \log_2\left(\frac{2}{10}\right) \\
 &\quad - \left(\frac{1}{10}\right) \times \log_2\left(\frac{1}{10}\right) - \left(\frac{1}{10}\right) \times \log_2\left(\frac{1}{10}\right) \\
 &= 0.442 + 0.464 + 0.332 + 0.332 = 1.571
 \end{aligned} \tag{2.6}$$

Calcular qual dos atributos clima, presença de visitantes ou dinheiro tem o melhor ganho de informação é o próximo passo. Os cálculos estão listados nas Equações 2.7, 2.8 e 2.9 respectivamente.

$$\Delta I(\text{Clima}) = 1.571 - \left(\frac{3}{10}\right) \times 0.918 - \left(\frac{4}{10}\right) \times 0.811 - \left(\frac{3}{10}\right) \times 0.918 = 0.70 \tag{2.7}$$

$$\Delta I(\text{Visitantes}) = 1.571 - \left(\frac{5}{10}\right) \times 0 - \left(\frac{5}{10}\right) \times 1.922 = 0.61 \tag{2.8}$$

$$\Delta I(\text{Dinheiro}) = 1.571 - \left(\frac{7}{10}\right) \times 1.842 - \left(\frac{3}{10}\right) \times 0 = 0.281 \tag{2.9}$$

Assim, o atributo clima será a raiz da árvore de decisão, pois apresenta o maior valor para  $\Delta I$ . A partir do nó com valor clima cria-se uma árvore como a ilustrada na Figura 19.

Para escolha do próximo atributo, o cálculo do melhor ganho de informação é repetido. Então, os valores do ganho para os atributos presença de visitantes e dinheiro em relação a cada um dos possíveis valores do atributo clima (Sol, Vento, Chuva) deve ser calculado. Seguindo

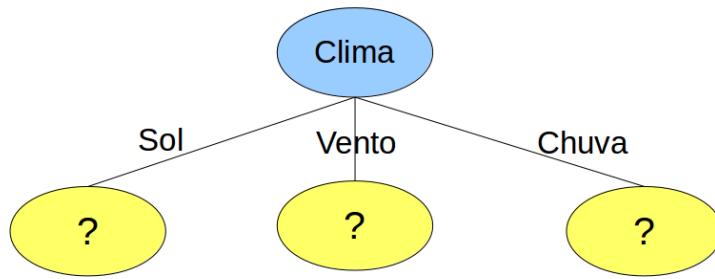


Figura 19 – Raiz da árvore de decisão com base nos dados da Tabela 1 (autoria própria)

Tabela 2 – Exemplos com valor de Clima igual a Sol

Clima	Presença de visitantes	Dinheiro	Destino
Sol	Sim	Sim	Cinema
Sol	Não	Sim	Tênis
Sol	Não	Sim	Tênis

pelo ramo com valor igual a Sol, apenas os exemplos da Tabela 2 são necessários no momento, pois estes apresentam o valor Sol para o atributo clima.

Primeiramente, calcula-se a entropia para o conjunto de dados representado na Tabela 2. Assim,  $I(C_{sol}) = 0.918$ .

$$\Delta I(C_{sol}, Visitantes) = 0.918 - \left(\frac{1}{3}\right) \times 0 - \left(\frac{2}{3}\right) \times 0 = 0.918 \quad (2.10)$$

$$\Delta I(C_{sol}, Dinheiro) = 0.918 - \left(\frac{3}{3}\right) \times 0.918 - \left(\frac{0}{3}\right) \times 0 = 0 \quad (2.11)$$

O atributo visitantes tem o maior valor de ganho e será o próximo nó de decisão na árvore. Continuando o processo, o atributo *Presença de visitantes* pode assumir os valores *Sim* ou *Não*. Então acrescenta-se esses ramos na árvore (Figura 20). Seguindo pelo ramo com valor *Sol* na árvore, os registros a serem considerados estão na Tabela 2. Nota-se que para o valor *Sim* do atributo *Presença de visitantes*, apresenta elementos de uma única classe, o que representa um conjunto puro. Assim, o ramo *Sim* encerra em uma folha de categorização com o valor *Cinema*. O mesmo acontece para o valor *Não*, que termina com um nó folha com o valor *Tênis*. A árvore gerada seguindo o ramo sol do atributo clima está ilustrada na Figura 20.

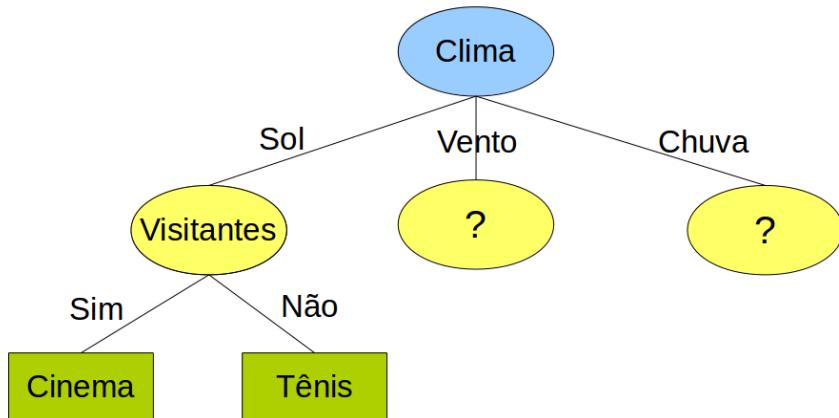


Figura 20 – Árvore gerada seguindo o ramo sol do atributo clima (autoria própria)

### 2.3.3 k-Means

Modelo de agrupamento de abordagem não supervisionada baseado em protótipo, k-Means é bastante simples e amplamente usado. Primeiro, escolhe-se  $K$  centroides iniciais, onde  $K$  é o número de grupos desejados que deve ser informado pelo usuário e os centroides são pontos no espaço de atributos, o qual pertencem os dados. Cada ponto é atribuído ao centroide mais próximo, e cada conjunto de pontos com o mesmo centroide forma um grupo (*cluster*). O próximo passo é a atualização do centroide de cada grupo baseado nos pontos que lhe foram atribuídos. Os passos de atribuição e atualização dos centroides são repetidos até que não haja mais alterações nos grupos, o que equivale aos centroides permanecerem os mesmos em duas iterações sucessivas.

O Algoritmo 3 descreve formalmente k-Means. Os passos de sua execução estão ilustrados na Figura 21. A Figura 21a mostra a escolha dos centroides iniciais. A operação de atribuição dos pontos ao centroide mais próximo é ilustrada na Figura 21b e a atualização na Figura 21c.

---

#### Algoritmo 3: Algoritmo k-Means básico

---

- 1 **início**
  - 2     Selecione  $k$  centroides iniciais
  - 3     **repita**
  - 4         Forme  $k$  grupos atribuindo cada ponto ao centroide mais próximo;
  - 5         Recalcule o centroide de cada grupo;
  - 6         **até os centroides não mudem (nenhum ponto mude de grupo);**
  - 7 **fim**
- 

Para algumas combinações de funções de proximidade e tipos de centroides, k-Means sempre converge para uma solução, ou seja, atinge um estado onde nenhum ponto mude de um grupo para outro e, consequentemente, os centroides permaneçam os mesmos. No entanto, as

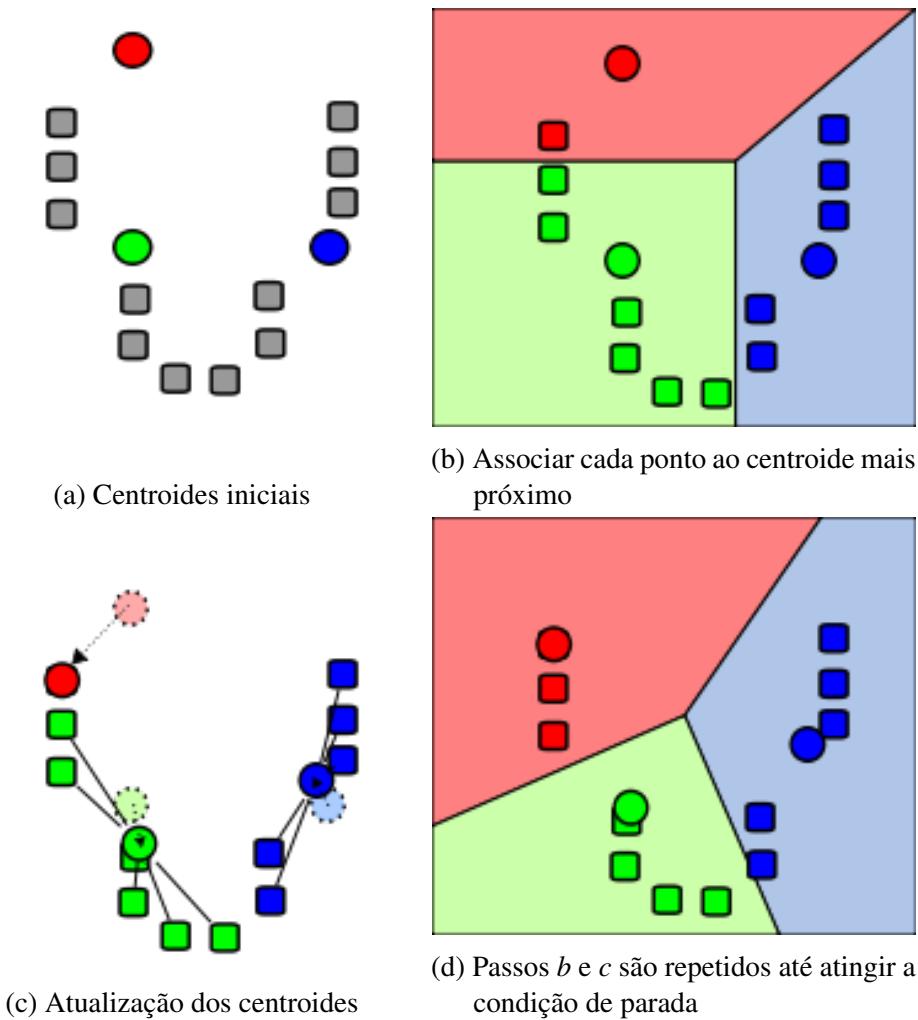


Figura 21 – Demonstração do algoritmo k-Means - adaptado de Wikipédia

vezes a condição de convergência é substituída por condições mais fracas, como por exemplo, repetir até que apenas 1% dos pontos mudem de grupo ou até que um número máximo de execuções seja alcançado (TAN; STEINBACH; KUMAR, 2009).

Para medir a qualidade de um agrupamento encontrado por k-Means quando os dados estão no espaço Euclidiano, usa-se a soma do erro quadrado (SSE) que está definida na Equação 2.12. Calcula-se o erro de cada ponto  $x$  presente no cluster  $C_i$ , que corresponde à sua distância Euclidiana ( $dist$ ) até o centroide  $c_i$  e depois calcula-se a soma total dos erros quadrados. Dados dois agrupamentos resultantes de diferentes execuções de k-Means, prefere-se aquele com menor SSE, pois este corresponde ao agrupamento em que os protótipos (centroides) melhor representam os pontos de seu grupo (TAN; STEINBACH; KUMAR, 2009). O objetivo é minimizar a variância *intra-cluster*, ao passo que aumenta a variância *inter-cluster*.

$$SSE = \sum_{i=1}^k \sum_{x \in C_i} dist(c_i, x)^2 \quad (2.12)$$

k-Means é sensível à escolha dos centroides iniciais. Indicá-los de maneira apropriada é a etapa chave para se ter um bom resultado no agrupamento. Uma abordagem comum é a escolha de forma aleatória, mas os grupos resultantes são frequentemente pobres (AGHA; ASHOUR, 2012).

Uma técnica que pode ser usada para mitigar esse problema, é fazer múltiplas execuções de k-Means a partir de centroides escolhidos aleatoriamente e depois selecionar o conjunto de grupos com menor SSE. Apesar de simples, essa estratégia pode não funcionar muito bem, dependendo do conjunto de dados e do número de grupos procurados (JAIN, 2010). Outra abordagem seria escolher o primeiro centroide de forma aleatória ou pegar o centroide de todos os pontos. Em seguida, para cada centroide inicial sucessivo, selecionar o ponto que estiver mais distante de qualquer um dos centroides iniciais já selecionados. Contudo, tal abordagem pode selecionar pontos externos, em vez de pontos em regiões densas (grupos), além de ser muito custosa. Para superar esses problemas, essa técnica pode ser aplicada a uma pequena amostra dos pontos.

k-Means é muito geral e simples, podendo ser usado em uma ampla variedade de tipos de dados, como documentos e séries temporais. Também é bastante eficiente, ainda que sejam necessárias várias execuções durante o processo de agrupamento. No entanto, ele não é apropriado para todos os tipos de dados. Mas, não tem um bom desempenho com grupos que apresentam ruído, que diferem em tamanho (Figura 22a) e densidade (Figura 22b) ou grupos não globulares (Figura 22c), embora consiga encontrar subgrupos puros quando o número de grupos especificado for grande o suficiente (ERTÖZ; STEINBACH; KUMAR, 2003). Na Figura 22, as figuras da esquerda mostram o que seria o agrupamento natural dos dados e as da direita, o agrupamento encontrado por k-Means.

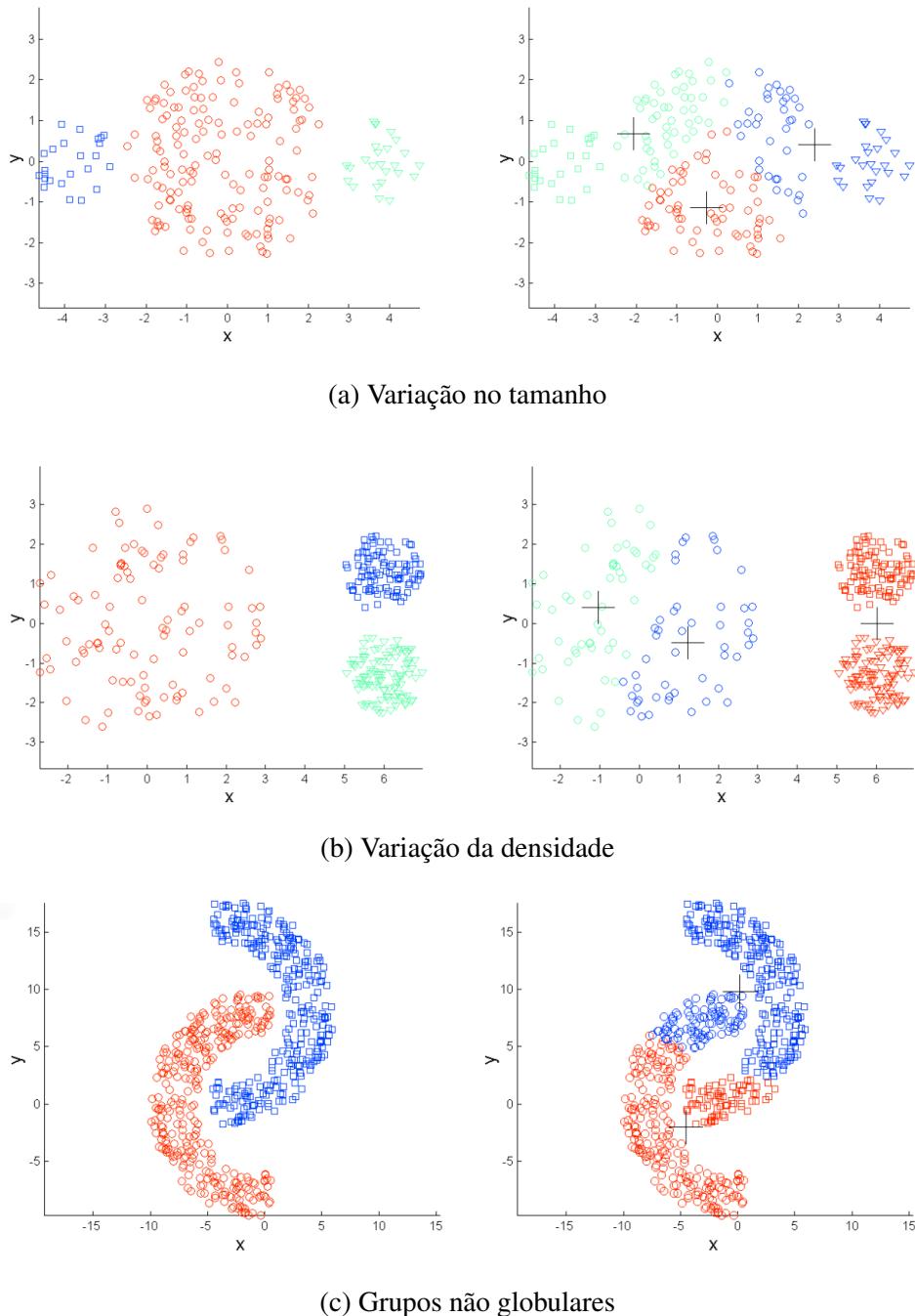


Figura 22 – Exemplos de conjuntos de dados em que o k-Means apresenta dificuldade para a formação dos grupos - adaptado de ([MACEDO, 2014](#))

### 2.3.4 DBSCAN

O DBSCAN (*Density-Based Spatial Clustering of Applications With Noise*<sup>1</sup>) é um modelo de agrupamento de dados proposto por Martin Ester, Hans-Peter Kriegel, Jörg Sander e Xiaowei Xu ([ESTER et al., 1996 apud CASSIANO, 2014](#)) baseado em densidade. Modelos com essa abordagem localizam regiões de alta densidade que estejam separadas entre si por re-

<sup>1</sup> Clusterização Espacial Baseada em Densidade de Aplicações com Ruído

giões de baixa densidade. Mais precisamente, DBSCAN usa a abordagem de densidade baseada em centro.

Nesse tipo de abordagem, a densidade de um determinado ponto do conjunto de dados é avaliada contando-se o número de pontos dentro de um determinado raio, chamado de  $Eps$ , incluindo o próprio ponto (TAN; STEINBACH; KUMAR, 2009). A Figura 23a ilustra graficamente esta técnica, onde o número de pontos dentro de raio  $Eps$  é 7 para o ponto A.

A densidade baseada em centro permite classificar os pontos em três tipos. **Ponto central**, quando o número de pontos que estão dentro de uma determinada vizinhança em torno do ponto, que é definida pela função de distância e o parâmetro  $Eps$  especificado pelo usuário, for maior do que um determinado limite,  $MinPts$ , também informado pelo usuário. Na Figura 23b, o ponto A é um ponto central, para o  $Eps$  especificado e  $MinPts \leq 7$ . **Ponto de borda** é o ponto que não é central, mas fica dentro da vizinhança de um ponto central. O ponto B na Figura 23b é um ponto de borda. **Ponto de ruído** é qualquer ponto que não seja central ou de borda. Na Figura 23b C é um ponto de ruído.

A vizinhança de um dado ponto  $p$ , pertencente ao conjunto de dados  $D$ , para um função de distância  $dist$  pode ser definida como na Equação 2.13.

$$Vizinhança(p) = \{q \in D \mid dist(q, p) \leq Eps\} \quad (2.13)$$

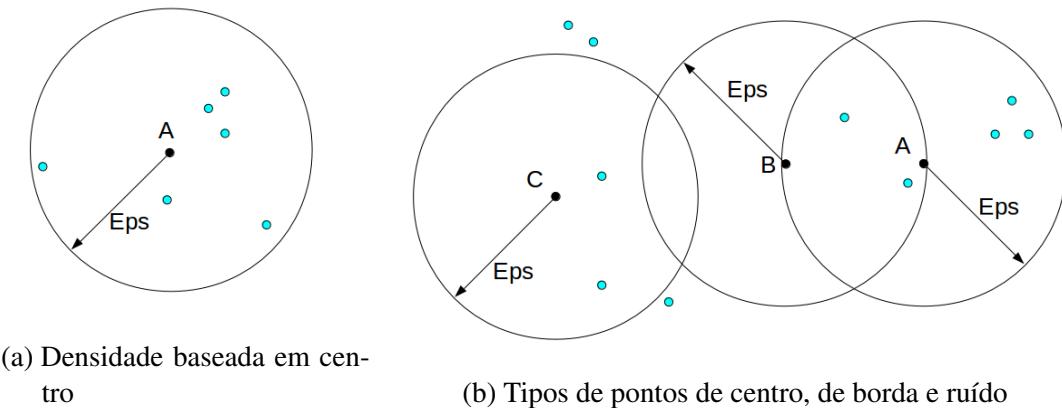


Figura 23 – Ilustração densidade baseada em centro e tipos de pontos - adaptado de (TAN; STEINBACH; KUMAR, 2009)

Na Figura 24 é possível ver a categorização dos pontos aplicada a um conjunto de dados que visualmente apresenta grupos com formatos arbitrários. Em 24a mostra-se os pontos originais. Em 24b os pontos verdes são pontos centrais, os que estão em azul representam os pontos de borda e em vermelho os ruídos.

O algoritmo inicia por um ponto arbitrário  $p$  e verifica sua vizinhança. Se  $p$  é um ponto central, então forma-se um novo grupo tendo  $p$  como centro. Se  $p$  é um ponto de borda, nenhum ponto pode ser alcançado por densidade a partir de  $p$  e, então, o algoritmo visita o próximo ponto

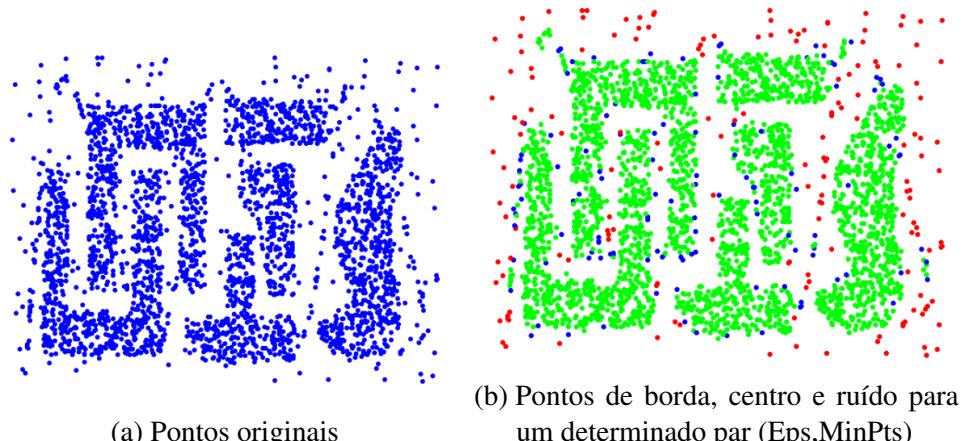


Figura 24 – Tipos de pontos encontrados por DBSCAN para um determinado conjunto de dados  
- adaptado de (CASSIANO, 2014)

(ESTER et al., 1996). O DBSCAN de forma iterativa busca por pontos diretamente alcançáveis de pontos centrais. Dessa forma, a união de dois ou mais grupos pode acontecer. O processo é feito até que nenhum novo ponto possa ser adicionado a qualquer *cluster* (CASSIANO, 2014). A Figura 25 mostra o resultado do algoritmo aplicado ao *dataset* visto na Figura 24a.

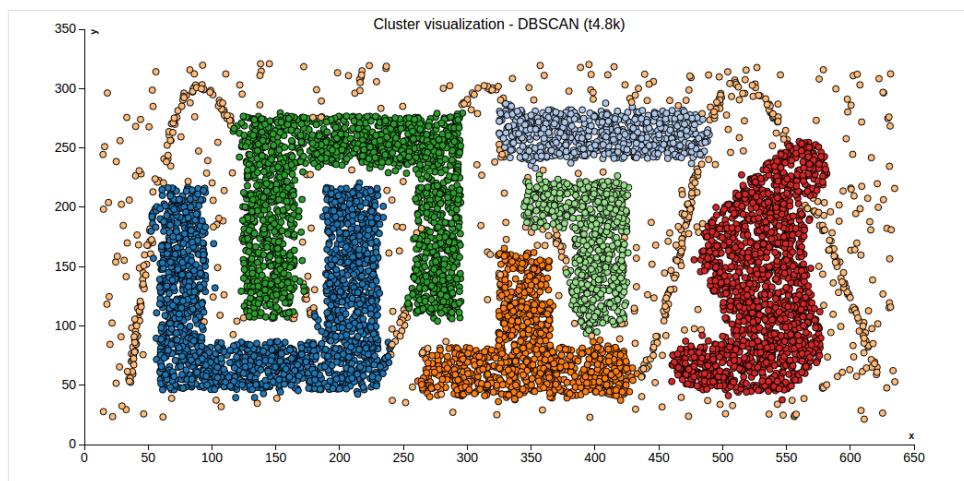


Figura 25 – Resultado algoritmo DBSCAN (autoria própria)

Cita-se como vantagens do DBSCAN a possibilidade de identificar grupos com os mais variados formatos possíveis, incluindo grupos cercados por outros grupos. Outra vantagem é não precisar que seja especificado o número de *clusters* a priori, ao contrário do k-Means por exemplo, além de possuir a noção de ruído. Como desvantagens, apresenta-se a necessidade de dois parâmetros pré-definidos (*Eps* e *MinPts*) e quais valores escolher para que um único par (*Eps*, *MinPts*) sirva para todos os pontos, principalmente quando existirem grupos com grandes diferenças em densidade.

O DBSCAN é ilustrado no Algoritmo 4.

---

**Algoritmo 4:** Algoritmo básico DBSCAN

---

**Entrada:** Um conjunto de dados  $D$ ,  $Eps$  e  $MinPts$   
**Saída:** Uma partição de  $D$  em *clusters*

```

1 início
2     grupoAtual ← 0;
3     para cada  $p \in D$  NÃO VISITADO faça
4         marque  $p$  como visitado;
5         vizinhosDeP ← pontos na vizinhança de  $p$  limitada por  $Eps$ ;
6         se NÚMERO DE  $vizinhosDeP < MinPts$  então
7             marque  $p$  como ruído;
8         senão
9             grupoAtual ← grupoAtual + 1;
10            expandirCluster( $p$ , vizinhosDeP, grupoAtual, Eps, MinPts);
11        fim
12    fim
13 fim

```

---



---

**Procedimento** Expandir Cluster

---

**Entrada:** Um ponto  $p$ ; Os vizinhos de  $p$   $vizinhosDeP$ ; O rótulo para o *cluster*  $C$ ;  
 $Eps$ ;  $MinPts$ ;

```

1 início
2     Adicione  $p$  ao grupo  $C$ ;
3     para cada  $q \in vizinhosDeP$  faça
4         se  $q$  AINDA NÃO FOI VISITADO então
5             marque  $q$  como visitado;
6             vizinhosDeQ ← pontos na vizinhança de  $q$  limitada por  $Eps$ ;
7             se NÚMERO DE  $vizinhosDeQ \geq MinPts$  então
8                 vizinhosDeP ← vizinhosDeP ∪ vizinhosDeQ;
9             fim
10        fim
11        se  $q$  AINDA NÃO TEM GRUPO então
12            Adicione  $q$  ao grupo  $C$ ;
13        fim
14    fim
15 fim

```

---

### 2.3.5 Gradiente descendente

O método do gradiente é um método de otimização que busca o mínimo de uma função por meio de um esquema iterativo que percorre o domínio da função na direção oposta ao seu gradiente (BRAGA, 2005). No presente trabalho, aborda-se o caso da regressão linear, onde o modelo tenta encontrar um hiperplano que melhor se adeque ao conjunto de dados.

A função linear de entrada  $x$  e saída  $y$  a ser encontrada, no caso de funções univariadas (de uma única variável) é da forma  $y = w_1x + w_0$ , onde os coeficientes  $w_1$  e  $w_0$  são valores reais a serem aprendidos e são chamados de pesos. O valor de  $y$  é alterado a partir da mudança dos valores dos pesos. Assim, defini-se  $\mathbf{w}$ , o vetor  $[w_0, w_1]$  de pesos e a Equação 2.14 a função hipótese (NG, 2016).

$$h_w(x) = w_1x + w_0 \quad (2.14)$$

A Figura 26 traz um exemplo de conjunto de dados de preços de imóveis *versus* área. A tarefa do modelo é encontrar uma reta,  $h_w$  que melhor se encaixe nesses dados. Para isso, é necessário encontrar o vetor de pesos  $[w_0, w_1]$  que minimize o erro médio quadrático (MSE) definido na Equação 2.15.

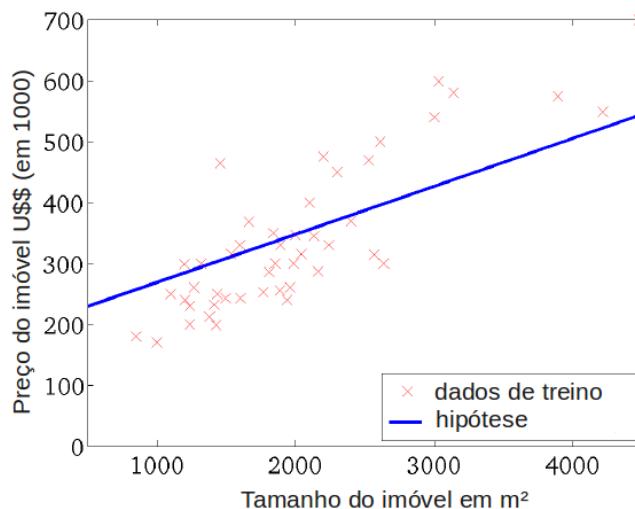


Figura 26 – Pontos de dados de preços *versus* área dos imóveis junto com um hipótese de função linear - adaptado de (NG, 2012a)

$$MSE(w_0, w_1) = \frac{1}{N} \sum_{j=1}^N (h(x) - y^{(j)})^2 \quad (2.15)$$

A descida do gradiente determina um vetor de pesos que minimiza o erro. O algoritmo inicia com um vetor de pesos arbitrários que vão sendo modificados repetidas vezes em pequenos passos, até atingir um erro mínimo local. A Figura 27 mostra um exemplo da superfície de

erro como uma função dos seus dois parâmetros  $w_0$  e  $w_1$ . A cada passo o vetor é alterado na direção que produz a maior queda ao longo da superfície de erro.

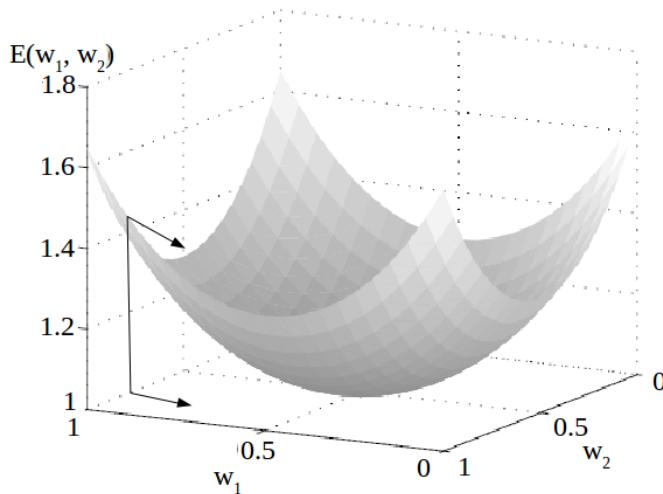


Figura 27 – Descida do gradiente - adaptado de (NG, 2016)

A regra de atualização para os pesos  $w_0$  e  $w_1$  são definidas pelas Equações 2.16.

$$w_0 \leftarrow w_0 + \alpha(y - h_w(x)); \quad w_1 \leftarrow w_1 + \alpha(y - h_w(x)) \cdot x \quad (2.16)$$

A atualização dos pesos ocorre da seguinte forma: quando  $h_w(x) > y$ , ou seja, a saída da hipótese for muito grande, deve-se reduzir  $w_0$ . Reduzir  $w_1$  se  $x$  for uma entrada positiva ou aumentar  $w_1$  se  $x$  for uma entrada negativa (NG, 2012a).

O Algoritmo 5 ilustra o Gradiente descendente para regressão linear univariada.

---

#### **Algoritmo 5:** Gradiente descendente para regressão linear univariada

---

```

1 início
2     initialize( $w_1, w_0$ );
3     repita
4          $w_0 = w_0 - \alpha \frac{1}{m} \sum_{j=1}^m (h_w(x^{(j)}) - y^{(j)})$ 
5          $w_1 = w_1 - \alpha \frac{1}{m} \sum_{j=1}^m (h_w(x^{(j)}) - y^{(j)}) \cdot x^{(j)}$ 
6     até convergir;
7  fim
```

---

A constante  $\alpha$  no Algoritmo 5 representa a taxa de aprendizagem. Quando  $\alpha$  é muito pequeno, a convergência pode ser lenta, o treinamento ocorre a “passos” curtos. Se  $\alpha$  for um valor muito alto, o treinamento ocorre com “passos” largos, o que talvez implique em uma convergência mais rápida, porém isso também pode fazer com que o gradiente descendente “pule” o mínimo global e não consiga convergir (NEDRICH, 2014).

Pode-se estabelecer diversos critérios de parada para o Algoritmo 5, tais como tolerância de erro, número de iterações, índice de acertos.

O Algoritmo 5 pode ser facilmente estendido para problemas de regressão linear multivariada, em que cada exemplo  $x_j$  do conjunto de treinamento é um vetor de  $n$  elementos e cada elemento representa uma característica de  $x_j$ . Retomando ao exemplo dos preços dos imóveis, o *dataset* seria algo como os dados apresentados na Tabela 3.

Tabela 3 – Dados de exemplo para preços *versus* área dos imóveis - adaptado de (NG, 2012b)

Tamanho ( $m^2$ )	# Quartos	# Andares	Idade do imóvel	Preço (\$ 1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Dessa forma, a função hipótese é definida como a Equação 2.17:

$$h_w(x_j) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = w_0 + \sum_{j=1}^n w_jx_j \quad (2.17)$$

Por convenção, como o termo  $w_0$  difere dos demais termos, deve-se criar um atributo de entrada fictício  $x_0$  com valor sempre igual a 1. É possível representar variáveis e pesos como vetores:

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}, W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

Assim, a hipótese é simplesmente o produto escalar dos pesos e do vetor de entrada (NG, 2012b):

$$h(x) = \mathbf{W}^\top \cdot \mathbf{X}$$

O Gradiente descendente para regressão linear multivariada é apresentado no Algoritmo

6.

**Algoritmo 6:** Gradiente descendente para regressão linear multivariada

---

```

1 início
2     initialize o vetor de pesos W;
3     repita
4         para  $j = 0$  até  $n$  faça
5              $w_j = w_j - \alpha \frac{1}{m} \sum_{i=1}^m (h(x^{(i)}) - y^{(i)}) x_j^{(i)}$ 
6         fim
7     até convergir;
8 fim

```

---

Ao lidar com a regressão linear multivariada, é preciso ter cuidado com a superadaptação, ou seja, é possível que alguma dimensão que não seja relevante para o problema pareça útil, uma vez que o conjunto de dados apresenta um números de dimensões superior. Outra questão que deve ser levada em consideração é a variação dos atributos. Por exemplo, voltando-se aos dados na Tabela 3 do exemplo dos preços dos imóveis, o atributo tamanho tem valores entre 800 e 2200, enquanto o número de quartos possui valores entre 1 e 5. Percebe-se que existe uma grande diferença entre os dois intervalos. Se o Gradiente descendente for aplicado com essa diferença nos dados, o contorno da função do erro será parecido à Figura 28a, formado por várias elipses, aumentando o tempo para a convergência. Uma possível solução para esse problema é a normalização dos atributos. Dessa forma o contorno da função erro torna-se mais uniforme, como mostrado na Figura 28b (NG, 2012b).

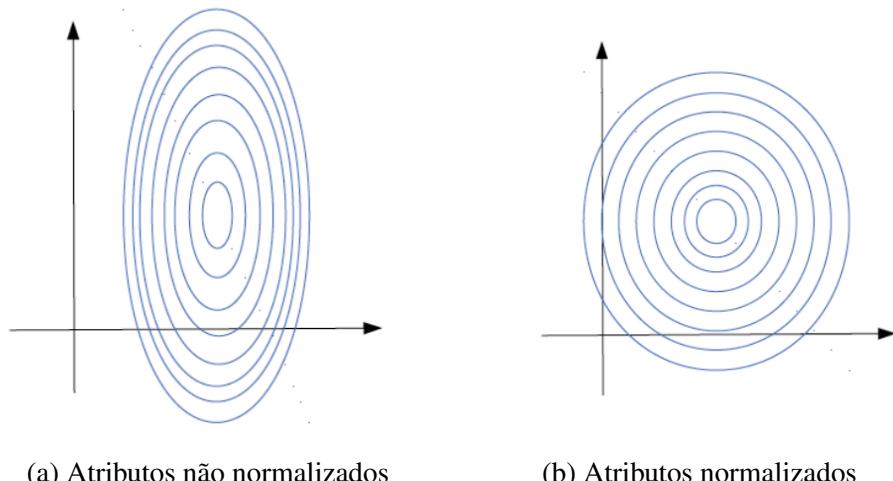


Figura 28 – Contorno da função erro - adaptado de (NG, 2012b)

Uma solução comumente usada é o da padronização de uma variável em estatística, através da Equação 2.18 que cria uma nova variável  $x_i$  com média 0 e um desvio padrão  $s_x$  igual a 1 (TRIOLA et al., 2005). Outra solução seria normalizar os valores por escala *min-max*

(Equação 2.19).

$$x_i = (x_i - \bar{x}) / s_x \quad (2.18)$$

$$x_i = \min + \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} (\max - \min) \quad (2.19)$$

Onde  $\min$  e  $\max$  representam respectivamente o limite inferior e superior dos novo valor para  $x_i$ . De forma semelhante,  $\min(x_i)$  corresponde ao menor valor da variável  $x_i$  no conjunto de dados e,  $\max(x_i)$ , o maior valor.

# 3

## O framework ml.js

Este capítulo descreve detalhes do desenvolvimento do ml.js. Na Seção 3.1, as implementações dos modelos e das classes auxiliares são descritas. Em seguida, a estrutura e o funcionamento da GUI são mostrados na Seção 3.2.

### 3.1 API

Alguns detalhes sobre o desenvolvimento da ferramenta para uso de aprendizagem de máquina com JavaScript são descritos nesta seção.

O paradigma da programação orientada a objetos é bastante utilizado para construção de aplicações, tanto no meio acadêmico quanto no mercado de trabalho. Em versões anteriores ao ECMAScript<sup>1</sup> 6, o JavaScript não apresenta suporte ao uso de classes no estilo clássico de código orientado a objetos (CÁSSIO, 2014; ECMA, 2015).

Classes em JavaScript são funções construtoras, usadas para criar objetos das quais utiliza-se o recurso do *prototype* (protótipo). Protótipo é um objeto associado a uma função construtora, onde é possível definir propriedades e funcionalidades que serão automaticamente aplicadas a instâncias de objetos (CÁSSIO, 2014). O Código 1 traz um exemplo de classe com *prototype*.

---

<sup>1</sup> ECMAScript é uma especificação padronizada da linguagem JavaScritp, a sua sexta versão ECMA-262 foi publicada em junho de 2015

### Código 1 – Exemplo de uso do *prototype*

```

1 //função construtora
2 function Carro(cor, velocidadeMaxima) {
3     this.cor = cor;
4     this.velocidadeMaxima = velocidadeMaxima;
5     this.velocidadeAtual = 0;
6 }
7
8 //prototype com métodos
9 Carro.prototype = {
10     acelerar: function() {
11         this.velocidadeAtual += 10;
12     }
13 }
14
15 var Lamborghini = new Carro("amarelo", 354);
16 Lamborghini.acelerar();
17 console.log(Lamborghini.cor + " : " + Lamborghini.velocidadeAtual);

```

Segundo (RESIG; BIBEAULT, 2013), a inclusão de orientação a objetos por meio de protótipos de função pode fornecer um volume de riqueza a desenvolvedores que sejam mais voltados para a codificação clássica de orientação a objetos. Ao permitir o maior grau de controle e estrutura que a orientação a objetos é capaz de trazer para o código, aplicações JavaScript podem melhorar em clareza e qualidade.

Para cada modelo proposto, foram criadas uma ou mais classes que implementam o seu comportamento. Além dessas, foi criada uma classe para manipulação e leitura de arquivos e para geração de visualização dos dados.

A classe *DataSet*, responsável pela leitura e manipulação dos dados, faz o uso do FileReader.js, que é uma pequena biblioteca (*wrapper*) para o uso da File API do JavaScript (GRINSTEAD, 2010). A File API permite a manipulação de objetos de arquivo no *client-side* de aplicações Web, bem como carregar arquivos e acessar seus dados (RANGANATHAN; SICKING, 2015; BIDELMAN, 2010) sem a necessidade de uma linguagem *serve-side* como PHP, o que torna a leitura de arquivos bem simplificada. A única exigência é que se tenha um *browser* moderno que dê suporte à API<sup>2</sup>.

Com a classe *DataSet*, é possível fazer a leitura de diversos tipos de arquivos, desde que sejam em formato de texto e tenham um padrão que possa ser mapeado para o método de leitura da classe. Assim, é possível carregar bases de treinamento que vão do formato *csv* aos clássicos arquivos *arff* do repositório UCI, formatos utilizados pelo *software WEKA*. Além da leitura, é

<sup>2</sup> Navegadores que dão suporte à File API: Internet Explorer: 10+, Firefox: 10+, Chrome: 13+, Opera: 12+, Safari: parcial. [Can I use File API](#). Acessado em 27 de maio de 2016.

possível remover, retornar, substituir e converter valores dos atributos dos conjuntos de dados.

O Código 2 traz um exemplo simples para a leitura de um arquivo. O método *loadFile* aceita como um de seus parâmetros um *callback* que será executado quando o carregamento do arquivo for concluído. O exemplo em questão apenas escreve no *console* do navegador o nome e o tamanho do arquivo que foi selecionado pelo usuário.

Código 2 – Exemplo de leitura de arquivo

---

```

1 var myData = new dataSet();
2 myData.loadFile({
3   loadend: function() {
4     //função executada após o carregamento do arquivo
5     console.log("Arquivo carregado: ", myData.file.name);
6     console.log("Tamanho: ", myData.file.extra.prettySize);
7   }
8 });

```

---

Para a geração da visualização dos dados é utilizada a biblioteca D3 (abreviação para *Data Driven Documents*), uma biblioteca JavaScript que permite a geração de gráficos usando HTML, SVG, e CSS ([MACLEAN, 2015](#)).

O SVG (*Scalable Vector Graphics*) trata-se de uma linguagem XML para descrever de forma vetorial desenhos e gráficos bidimensionais, que podem ser apresentados de forma estática, dinâmica ou animada ([FERRAIOLLO, 2001](#)).

No Código 3, é apresentado um exemplo do uso da classe *plot* para visualização de dados. A Figura 29 ilustra um exemplo de visualização para o dataset Íris ([LICHMAN, 2013](#)).

Código 3 – Exemplo geração de gráfico

---

```

1 var plotDataSet = new plot({
2   "containerPlot": "container-plot-iris",
3   "title": "Dataset: Íris"
4 }).plot2D(
5   datasetIris,
6   {"label": "petallength"},
7   {"label": "petalwidth"}
8 );

```

---

A Figura 30 mostra mais detalhes das classes *plot* e *dataSet*. Outros exemplos de uso podem ser vistos no Apêndice A.

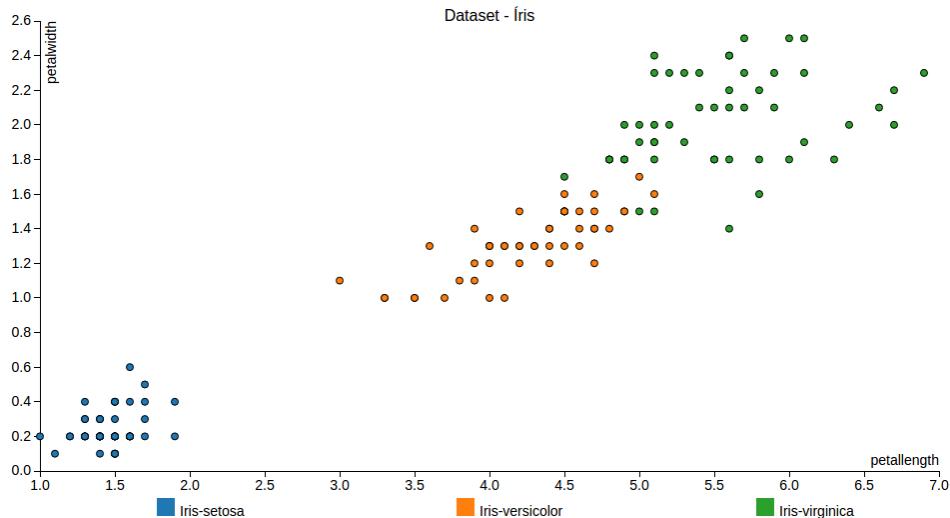


Figura 29 – Visualização dois atributos do dataset Íris resultante do Código 3 (autoria própria)

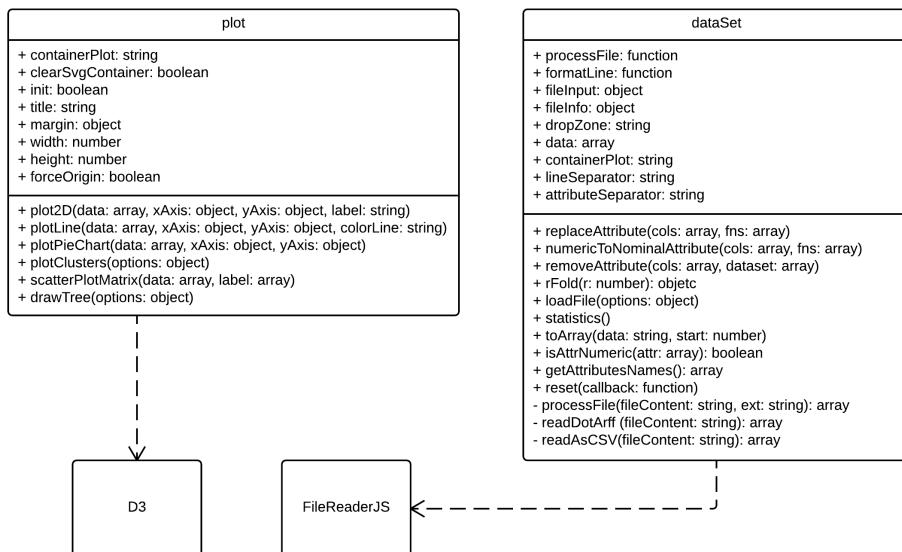
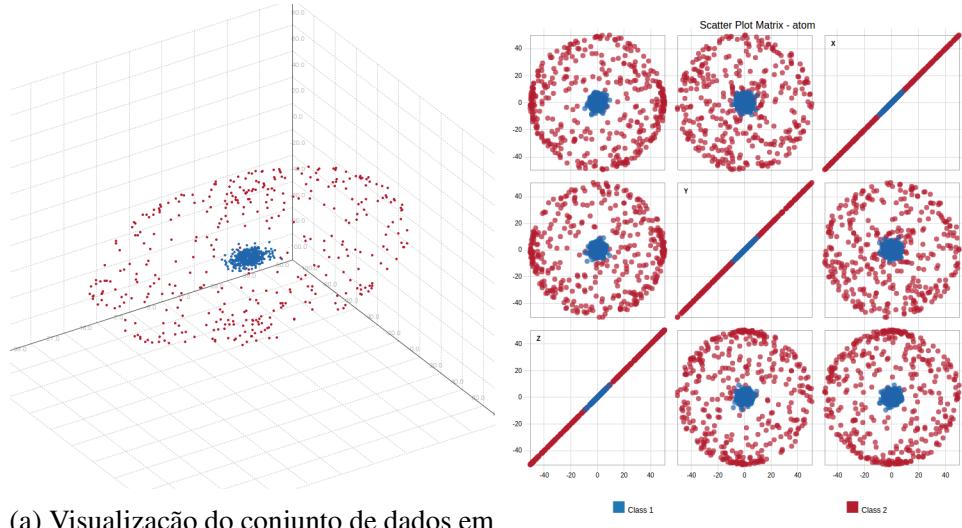


Figura 30 – Classes para geração de gráficos e leitura de arquivos (autoria própria)

Além de gráficos no estilo *X versus Y*, é possível plotar gráficos em formato de pizza, gráficos de linha, matriz de *scatter plots* (ver Figura 31) para visualização de conjuntos de dados com mais de duas dimensões.



(a) Visualização do conjunto de dados em três dimensões ([BARTON, 2016](#))      (b) Matriz scatter plots (autoria própria)

Figura 31 – Exemplo de matriz *scatter plots* para o conjunto de dados Atom<sup>3</sup>

As classes que implementam os algoritmos possuem um método principal com o nome em comum de acordo com o tipo de atividade. Por exemplo, k-Means e DBSCAN possuem o método *buildClusterer* e *buildClassifier* para os classificadores. A Figura 32 mostra as principais classes e métodos do ml.js. O Apêndice A trás um exemplo de aplicação de cada um deles.

<sup>3</sup> Conjunto de dados composto por 800 instâncias com 3 dimensões e separadas em duas classes. Disponível em: [Fundamental Clustering Problem Suite](#). Acessado em 20 de julho de 2016

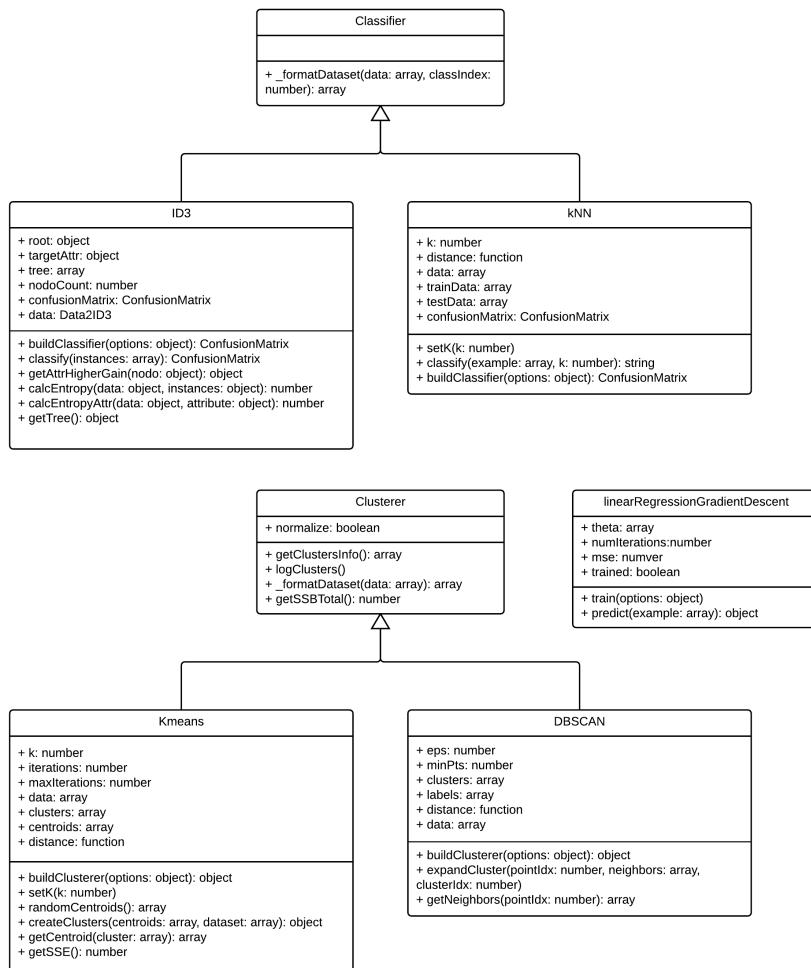


Figura 32 – Principais classes e métodos do ml.js (autoria própria)

Na implementação do Gradiente descendente para a regressão linear, são usados dois critérios de parada: (1)  $\Delta E r r o$  e (2) o número máximo de iterações passada como parâmetro pelo usuário. O  $\Delta E r r o$  é a diferença entre o erro médio quadrático calculado antes e após a atualização dos pesos. Quando  $\Delta E r r o$  é menor ou igual ao valor da precisão fornecida como entrada ao modelo, ele atingiu a convergência. O segundo critério serve para evitar que a execução entre em *loop* infinito.

A implementação do modelo ID3 foi baseada em (SIMEONE; FERREIRA, 2014). Duas classes são utilizadas: a primeira representa a estrutura dos nós da árvore e contém funções de manipulação dos dados e a segunda é propriamente a implementação do ID3.

Para os modelos que utilizam alguma função de distância, o ml.js possibilita que essa função seja passada como um parâmetro, bem como outras variáveis que são utilizadas, tornando assim seu uso mais flexível.

Buscou-se parametrizar o máximo de opções para cada modelo, deixando o usuário com a possibilidade de personalizar os parâmetros de forma adequada para o problema que se quer resolver ou simplesmente demonstrar alguma propriedade do mesmo. Uma pequena

observação é que nem todos os parâmetros estão disponíveis para personalização por meio da GUI, necessitando dessa forma que se tenha alguma experiência com a linguagem JavaScript para desenvolver exemplos mais elaborados.

## 3.2 GUI

Por meio da interface gráfica, é possível trabalhar com as atividades de AM de maneira simples e direta, sem a necessidade de uma codificação para a execução e visualização dos resultados.

O layout da GUI foi baseado na versão gráfica do *software WEKA*, que apresenta esquema de abas para o carregamento dos dados e para as atividades de classificação, agrupamento e regressão. Para sua montagem utilizou-se o *framework* para desenvolvimento de páginas Web Bootstrap ([BOOTSTRAP, 2015a](#)), um *template* ([BOOTSTRAP, 2015b](#)) fornecido pelo mesmo e a popularmente conhecida biblioteca para JavaScript jQuery ([JQUERY, 2015](#)). A Figura 33 traz a tela inicial da GUI.

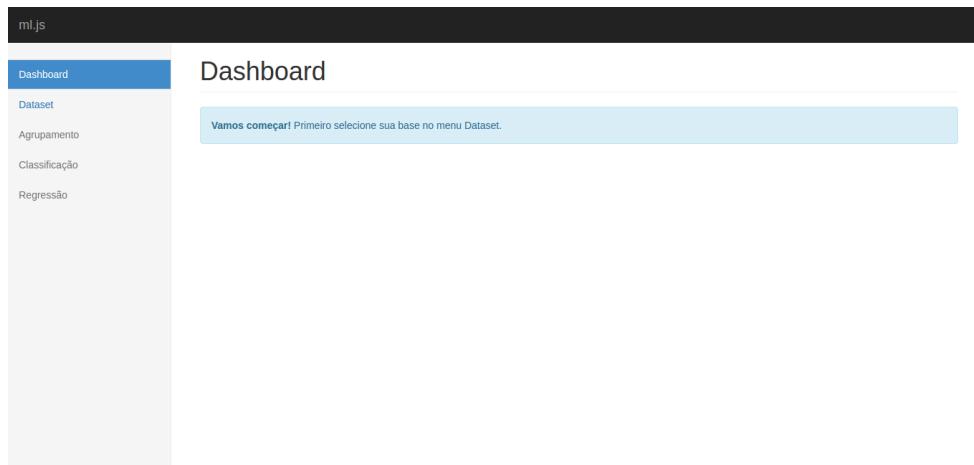
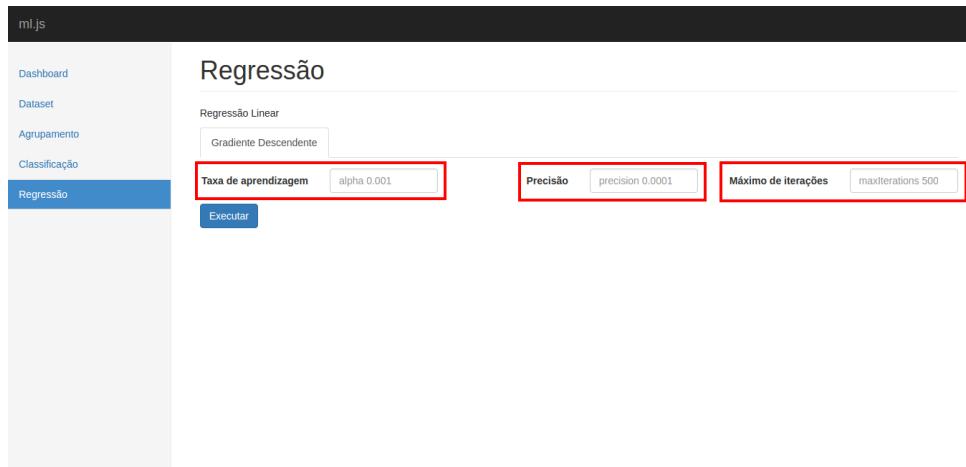


Figura 33 – Tela inicial da GUI (autoria própria)

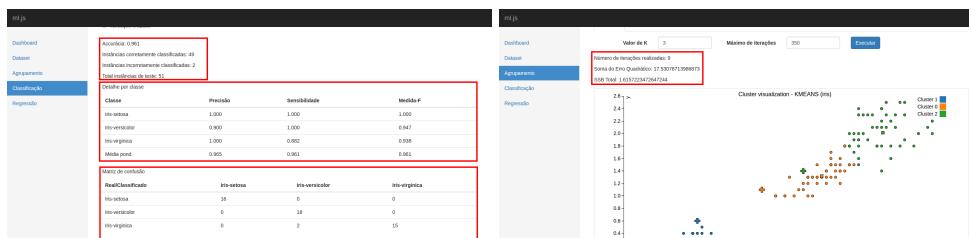
A estrutura básica do *background* da GUI é composta por uma instância da classe *dataset* nomeada de *dataMain* responsável pelo carregamento dos dados a serem usados pelos modelos implementados. Cada modelo possui um formulário por onde o usuário pode configurar os parâmetros necessários para sua execução. O atributo *placeholder*<sup>4</sup> dos campos dos formulários apresentam o nome do parâmetro e seu valor padrão (Figura 34), o que acaba funcionando como um *help* para os desenvolvedores e eliminando a necessidade de informar todos os parâmetros para usar o modelo.

Cada formulário contendo os parâmetros do modelo a ser submetido, faz o uso da classe *modelFactory* criada com base no padrão de projeto *Factory* ([DIAZ; HARMES, 2007](#)), que

<sup>4</sup> O atributo *placeholder* especifica uma pequena dica que descreve o valor esperado de um campo de entrada (por exemplo, um valor de amostra ou uma curta descrição do formato esperado). A dica é exibida no campo de entrada antes que o usuário entre com um valor (Fonte: w3schools).

Figura 34 – Exemplo de valores *default* (autoria própria)

irá retornar uma instância do modelo que lhe foi solicitado. A partir da instância retornada, o método *main* do modelo é executado e, ao final de sua execução, os resultados são exibidos, podendo este ser visualizações do *dataset*, métricas de avaliação dos modelos, métricas referentes a execução (por exemplo, número de iterações necessárias até atingir a convergência), um gráfico em formato de árvore (no caso específico do ID3), como pode ser visto na Figura 35.



(a) Exemplo métricas ID3

(b) Exemplo métricas k-Means

Figura 35 – Exemplo de métricas geradas (autoria própria)

Mais detalhes e exemplos de como usar a GUI podem ser vistos no Apêndice B. A Figura 36 mostra um fluxo de uso da GUI.

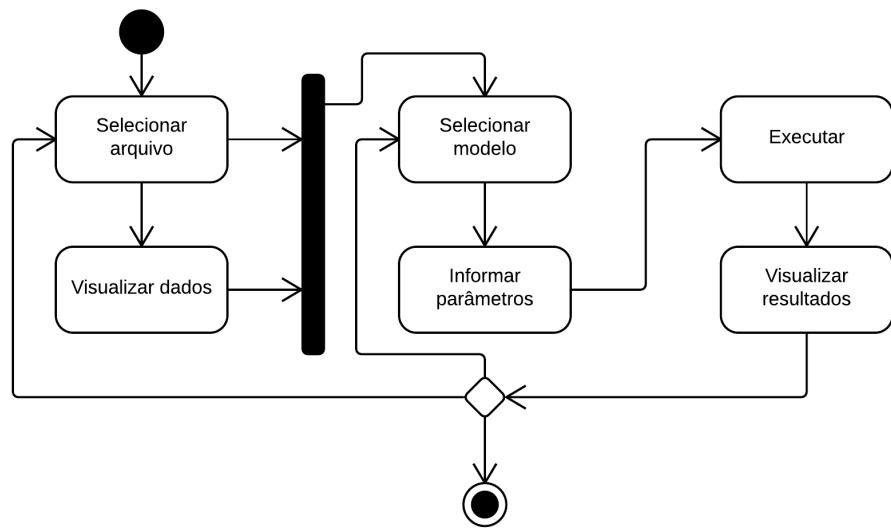


Figura 36 – Exemplo fluxo de uso da GUI (autoria própria)

# 4

## Experimentos, resultados e discussão

Neste capítulo são descritos os resultados obtidos a partir dos experimentos realizados para demonstrar o funcionamento do *framework* ml.js. Inicia-se descrevendo os conjuntos de dados escolhidos para os experimentos, suas características e os ajustes necessários para permitir tal tarefa.

Após a apresentação das bases de dados utilizadas, é feita uma breve exposição de métricas e formas de avaliar modelos de aprendizado de máquinas, as quais também foram implementadas e adicionadas ao *framework*. Por fim, são descritos os experimentos e resultados. Vale destacar que a maioria dos experimentos foram realizados diretamente pelo *interface* gráfica do ml.js, com os devidos ajustes nas bases de dados e parâmetros de entrada dos modelos.

Os experimentos foram realizados utilizando o navegador Google Chrome Versão 50 (64-bit), sistema operacional Ubuntu 15.04.

### 4.1 Bases de dados utilizadas

Esta seção descreve os conjuntos de dados que foram utilizados para realizar os experimentos do presente trabalho. É feita ainda uma breve discussão sobre o *Open data* (AUER et al., 2007), trazendo definições e exemplos de aplicações que utilizam dados abertos em outros países e também no Brasil.

#### 4.1.1 Conjuntos de dados clássicos

Existem conjuntos de dados largamente citados e utilizados para exemplificar o uso de técnicas de aprendizado de máquina pela literatura especializada. É bem possível que o mais

referenciado seja o conjunto de dados Íris (LICHMAN, 2013)<sup>1</sup>. Ele consiste de informações de 150 flores Íris, sendo 50 exemplos de cada uma das três espécies: Setosa, Multicolor e Virgíñica. Cada linha do *dataset* contém cinco atributos: comprimento da sépala, largura da sépala, comprimento da pétala, largura da pétala e classe (espécie). Esse conjunto de dados está disponível a partir do repositório *Machine Learning Repository* da Universidade da Califórnia em Irvine (UCI) (FISHER, 1936).

Na Figura 37 é possível ver uma matriz de *scatter plot* do *dataset* Íris. Uma característica desse conjunto de dados é que as flores da espécie setosa estão bem separadas das outras duas espécies.

Alguns *datasets* clássicos foram utilizados para demonstração e comparação dos modelos de agrupamento implementados e estão disponíveis em (ULTSCH, 2016; BARTON, 2016). As bases estão descritas na Tabela 4.

Tabela 4 – Conjuntos de dados utilizados para os experimentos com os modelos de agrupamento

Nome	# Exemplos	# Atributos	# Classes
Banana <sup>2</sup> (BARTON, 2016)	4811	3	2
Cluto-t4.8k <sup>3</sup> (KARYPIS; HAN; KUMAR, 1999)	8000	3	6
3-Spiral <sup>4</sup> (CHANG; YEUNG, 2008)	312	3	3
Smile1 <sup>5</sup> (HANDL; KNOWLES, 2004)	1000	3	4
Target <sup>6</sup> (ULTSCH, 2005)	770	3	6

Uma modificação realizada nas bases, foi a alteração dos valores dos nomes das classes, algumas eram representadas por um número, então foram trocadas para a palavra Classe seguida do número, por questões de estética e geração das visualizações.

#### 4.1.2 O conjunto de dados GPS Trajectories

Um conjunto de dados alimentado pelo aplicativo Android GO!Track (GO!TRACK, 2015), está disponível a partir do repositório UCI (CRUZ et al., 2015). Derivado de um projeto de pesquisa (DCOMP/UFS & PROCC/UFS), o objetivo do aplicativo é fornecer ao usuário um conjunto de funcionalidades úteis para o trânsito diário como previsão de tempo de viagem, sugestão de rotas, previsão de congestionamento.

<sup>1</sup> O conjunto de dados conta com mais de 1 milhão de acessos

<sup>2</sup> Disponível em: <<https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/banana.arff>>

<sup>3</sup> Disponível em: <<https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/cluto-t4.8k.arff>>

<sup>4</sup> Disponível em: <<https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/3-spiral.arff>>

<sup>5</sup> <<https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/smile1.arff>>

<sup>6</sup> Disponível em: <<https://github.com/deric/clustering-benchmark/blob/master/src/main/resources/datasets/artificial/target.arff>>

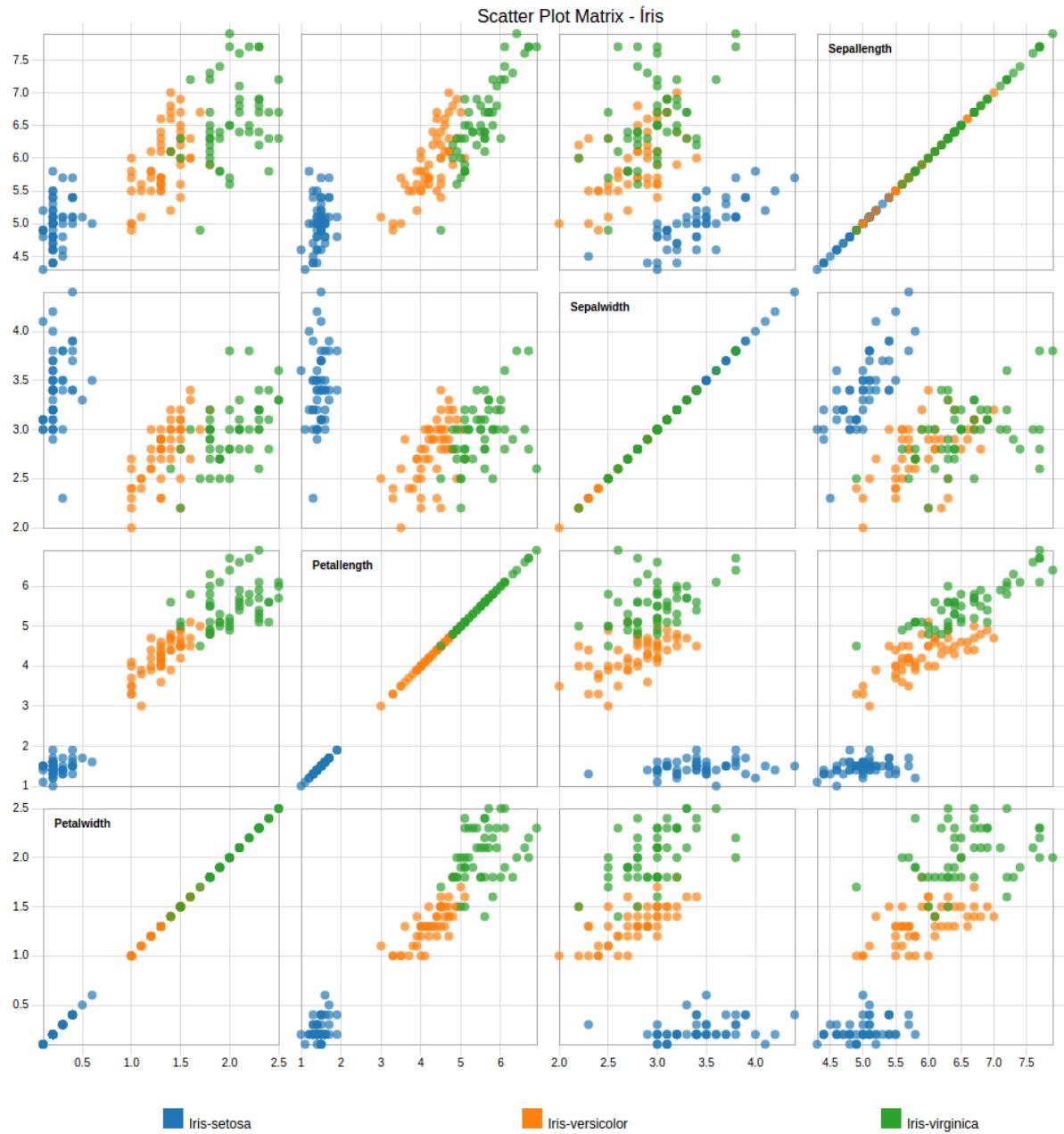


Figura 37 – Matrix de *scatter plot* para atributos do conjunto de dados Íris (autoria própria)

A base contém 163 exemplos<sup>7</sup>. Os exemplos são representados por atributos como velocidade média, distância percorrida, tempo, meio de transporte (carro ou ônibus), avaliação de percurso (bom, ruim, regular), avaliação de tempo (chuvisco, ensolarado), avaliação do ônibus (cheio, normal, vazio). Na Figura 38, uma matriz de *scatter plot* mostra a correlação entre os atributos do conjunto de dados *GPS Trajectories*.

Para os experimentos com a base *GPS Trajectories*, foi realizado um pré-processamento dos dados com as seguintes operações: (i) remoção dos atributos que identificam os usuários (*id*, *id\_android*), a avaliação do ônibus (*rating\_bus*), avaliação do clima (*rating\_weather*) (pelo mo-

<sup>7</sup> Até a data de 29 de agosto de 2016



Figura 38 – Matriz de *scatter plot* para atributos do conjunto de dados *GPS Trajectories* (autoria própria)

tivo de apresentar os mesmos valores para a maioria dos exemplos) e *linha* que identifica a rota do ônibus, (ii) conversão para a forma nominal dos atributos *speed*, *time*, *distance*, aplicando-se uma função que classifica o valor em  $\leq$  ou  $>$  que a sua mediana ([GARCIA, 2003](#)), (iii) apenas para melhorar a visualização dos resultados o atributo *rating* foi convertido para *good*, *normal* ou *bad* e o atributo *car\_or\_bus* para seu respectivo valor, *car* ou *bus*.

### 4.1.3 Bases de dados do *Open data*

Segundo a definição da *Open Knowledge Foundation*<sup>8</sup>, dados abertos (*open data*) são dados que qualquer pessoa pode acessar, usar e compartilhar, estando sujeitos à, no máximo, exigência de atribuição da fonte e compartilhamento pelas mesmas regras.

Um movimento que teve seu início em 2007, na California, e ganhou mais força em 2009 com a adesão dos EUA, o *open data* movimenta comunidades por todo o mundo para cobrar informações públicas do governo (CHIGNARD, 2013). Com a liberação desses dados, a população pode acompanhar e colaborar com os processos de governo e com o controle social das políticas. A natureza do *open data* pode ser de qualquer tipo, não somente de dados governamentais.

Os dados abertos governamentais são regidos por três leis propostas por David Eaves, ativista no movimento *open data* e especialista em políticas públicas (EAVES, 2009), mas que podem ser aplicadas de forma geral a qualquer tipo de dado aberto: (1) se o dado não pode ser encontrado e indexado na Web, ele não existe, (2) se não estiver aberto e disponível em formato comprehensível por máquina, ele não pode ser reaproveitado, e (3) se algum dispositivo legal não permitir sua replicação, ele não é útil.

Os dados abertos têm um grande potencial para exploração. Diversas áreas e atividades podem se beneficiar dessa disponibilidade, inclusive o governo. Existem exemplos de que os dados abertos estão gerando valor em um grande número de áreas e atividades. Transparência e controle democrático, inovação, participação popular, medição do impacto das políticas, conhecimento novo a partir da combinação de fontes de dados e padrões são algumas dessas áreas. Alguns exemplos de sucesso do uso de dados abertos podem ser vistos em (LABHACKER; NIC.BR; W3C BRASIL, 2011), como os projetos *Tax Tree* e *Where Does My Money Go*, da Finlândia e Grâ-Bretanha respectivamente, que mostram como os recursos dos impostos estão sendo gastos pelo governo. No Canadá, a abertura de dados permitiu que se economizassem 3.2 bilhões de dólares canadenses em fraudes fiscais de caridade (INTERNATIONAL, 2011). Há muitos outros exemplos, como mapa com a localização de banheiros públicos, serviço que permitem encontrar locais para morar de acordo com características indicadas, monitoramento da qualidade do ar, serviço que busca lugares onde é possível caminhar com um cachorro. São todos exemplos de serviços que usam dados abertos.

No Brasil, a partir de 2011 quando foi sancionada a Lei de Acesso a Informação Pública (Lei 12.527/2011), que regula o acesso a dados e informações retidas pelo governo, essa iniciativa de colaboração passou a ter mais atenção. No contexto de dados abertos governamentais, foi criado o **Portal Brasileiro de Dados Abertos**, que é a ferramenta construída pelo governo para centralizar a busca e o acesso dos dados e informações públicas. Os dados do portal são condizentes com normas técnicas e com as três leis de dados abertos. A Tabela 5 traz exem-

<sup>8</sup> The Open Definition. Disponível em: <<http://opendefinition.org/>>. Acessado em: 18 de Abril de 2016

plos de projetos brasileiros que utilizam dados abertos. Tanto a página do Portal Brasileiro de Dados Abertos quanto ([LABHACKER; NIC.BR; W3C BRASIL, 2011](#)) apresentam uma lista maior de exemplos. No entanto, muitos dos serviços não estão mais disponíveis para acesso ou apresentam erro durante o uso.

Tabela 5 – Exemplos de aplicativos e serviços que utilizam dados abertos (LABHACKER; NIC.BR; W3C BRASIL, 2011)

Nome	Descrição	URL	Dados utilizados
Aeroportos Brasil	Aplicativo que mostra o movimento de aeronaves e passageiros nos aeroportos administrados pela Infraero.	< <a href="http://ison.stratebi.es/aerobrasil">http://ison.stratebi.es/aerobrasil</a> >	Movimento dos Aeroportos Administrados pela Infraero
CMSP	Visualização das prestações de contas disponibilizadas no sítio da Câmara Municipal de São Paulo	< <a href="http://cmsp.topical.com.br">http://cmsp.topical.com.br</a> >	Dados provenientes da Câmara Municipal de São Paulo
Radar Parlamentar	Análise matemática sobre os dados de votações de projetos de lei na câmara para determinar as “semelhanças” entre partidos na atuação parlamentar.	< <a href="http://radarparlamentar.polignu.org">http://radarparlamentar.polignu.org</a> >	Votações dos Vereadores da Câmara Municipal de São Paulo
Reclamações Procon	É um serviço de busca e visualização intuitiva de dados abertos disponibilizados pelo PROCON relativos ao ano de 2011.	< <a href="http://reclamacoesprocon.com.br">http://reclamacoesprocon.com.br</a> >	Cadastro Nacional de Reclamações Fundamentadas
Reputação S.A.	Aplicativo que traz diversas informações sobre as empresas em formato ilustrativo e intuitivo, sendo de fácil utilização em smartphones e tablets devido ao layout vertical.	< <a href="http://reputacao-sa.org">http://reputacao-sa.org</a> >	Cadastro Nacional de Reclamações Fundamentadas
Siga seu vereador	O site tem como objetivo criar uma plataforma de linha do tempo com as ações realizadas pelos atuais vereadores, sendo possível que os usuários sigam os vereadores selecionados.	< <a href="http://www.vereadores.org">http://www.vereadores.org</a> >	Votações dos Vereadores da Câmara Municipal de São Paulo

O Brasil foi o décimo segundo colocado no Índice Global de Dados Abertos 2015 que é realizado pela *Open Knowledge International* em parceria com uma rede de especialistas e colaboradores. Esse ranking é baseado na disponibilidade e acessibilidade de informações

ligadas as áreas centrais que incluem gastos governamentais, resultados eleitorais, horários dos meios de transporte e níveis de poluição, entre outras (BRASIL, 2015).

Uma base de dados escolhida foi a do Comitê Nacional para os Refugiados (CONARE). Trata-se de um conjunto de dados extraídos dos registros efetuados pela Coordenação-Geral do CONARE, após recebimento dos processos de solicitações de refúgio enviados pelo Departamento de Polícia Federal (JUSTIÇA, 2016). Esta base é composta por 38782 registros, contendo atributos como sexo, estado civil, idade, país de origem, continente, cidade destino da solicitação. Os registros que apresentavam valores de atributos faltando ou valores desconhecidos foram removidos, resultando em um total de 34969 exemplos.

A partir de dados de (RIO2016, 2016a; RIO2016, 2016b; WIKIPEDIA, 2016) foi construído um banco de dados relacional, com os resultados dos jogos olímpicos 2016. A partir desse banco de dados, foi montado o conjunto de dados utilizado nos experimentos. A base contém dados como colocação final nos jogos, número de medalhas de ouro, prata e bronze, número de atletas participantes de cada país. Nos experimentos, foram utilizados apenas 87 registros, que correspondem ao número de países que conquistaram medalhas.

## 4.2 Métricas e avaliação de modelos

Uma etapa importante para as tarefas de aprendizagem de máquina é a avaliação. Esta seção traz dois métodos comumente usados para avaliar o desempenho de um classificador: *holdout* e validação cruzada.

No método *holdout*, os dados originais do *dataset* são divididos em dois conjuntos distintos. O primeiro é usado para o treinamento do modelo, chamado de conjunto de treinamento e o segundo é chamado de conjunto de teste, o qual é usado para avaliar o seu desempenho. A proporção geralmente usada é  $\frac{2}{3}$  para o treino e  $\frac{1}{3}$  para teste.

O segundo método é a validação cruzada. Nessa abordagem, cada registro é usado o mesmo número de vezes para treinamento e exatamente uma vez para teste. Denominada validação cruzada *k-fold*, o conjunto de dados é divido em *k* partições de tamanho igual. A partir disso, uma partição é utilizada para teste e as *k-1* restantes utilizadas para treinamento. Esse processo é realizado *k* vezes, alternando-se de forma circular o conjunto de teste. Assim, cada partição é usada para teste exatamente uma vez. Uma abordagem especial desse método, chamada de *leave-one-out* (deixe-um-fora), divide o conjunto de dados em *N* partes, onde *N* é o tamanho do *dataset*. Assim, cada partição é composta por apenas um registro. A vantagem é que todos os registros são usados no treinamento.

O desempenho de modelos de classificação é avaliado a partir do número de acertos e erros de classificação quando aplicado a um conjunto de teste. Esses números são tabulados em uma tabela conhecida como matriz de confusão (FRIZZARINI, 2013). A matriz de confusão é

formada pela contagem de registros de teste previstos correta e incorretamente pelo modelo de classificação. O desempenho do modelo é avaliado por meio dessa matriz. A Tabela 6 mostra a matriz de confusão para um problema de classificação binário. A diagonal principal representa o número de previsões corretas feita pelo modelo para cada classe. As demais entradas representam o número de previsões incorretas para cada classe. *VP* (verdadeiro positivo) significa o número de registros da classe positiva que foram classificados com positivo, *VN* (verdadeiro negativo) são registros pertencentes à classe negativa corretamente classificados como negativo, *FP* (falso positivo) é o número de registros classificados como positivo mas na verdade são da classe negativa e *FN* (falso negativo) é a quantidade de registros da classe positiva que foram classificados incorretamente como sendo da classe negativa.

Tabela 6 – Matriz de confusão para um problema de 2 classes

		Classe Preditada	
		Positivo	Negativo
Classe Verdadeira	Positivo	VP	FN
	Negativo	FP	VN

A partir dos dados da matriz de confusão, é possível calcular as métricas acurácia, sensibilidade, precisão e medida-F.

Acurácia, definida na Equação 4.1, representa a proporção de previsões corretas, sem discriminação do que é positivo ou negativo. A taxa de classificação incorreta, o erro, é usualmente medida como sendo o complemento da acurácia. Sensibilidade é a proporção de verdadeiros positivos que foram corretamente classificados como verdadeiros positivos e está definida na Equação 4.2. Precisão é a proporção de verdadeiros positivos em relação a todas as previsões positivas. A Equação 4.3 a define. Finalmente, a medida-f está definida na Equação 4.4 e representa a média harmônica ponderada da precisão e da sensibilidade. O *w* na equação é um peso que determina a importância da precisão e da sensibilidade. Para *w* = 1, a precisão e a sensibilidade possuem igual relevância.

$$Acurácia = \frac{VP + VN}{N} \quad (4.1)$$

$$Sensibilidade = \frac{VP}{VP + FN} \quad (4.2)$$

$$Precisão = \frac{VP}{VP + FP} \quad (4.3)$$

$$Medida - F = \frac{(w+1) \times sensibilidade \times precisão}{sensibilidade + (w * precisão)} \quad (4.4)$$

Onde *N* é o número de elementos do conjunto de teste.

Para um problema de classificação com mais de duas classes, como ilustrado na Tabela 7, os valores de  $VP$ ,  $FP$  e  $FN$  são calculados de forma individual para cada classe da seguinte maneira:  $VP$  é igual à soma da diagonal principal,  $FP$  é igual à soma da coluna da classe  $j$  menos o valor de  $p_{jj}$ ,  $FN$  é a soma da linha  $i$  menos o valor de  $p_{ii}$  (SOKOLOVA; LAPALME, 2009).

Tabela 7 – Matriz de confusão para um problema não binário

		Classe Predita				
		Classe 1	Classe 2	Classe 3	...	Classe n
Classe Verdadeira	Classe 1	$p_{11}$	$p_{12}$	$p_{13}$	...	$p_{1n}$
	Classe 2	$p_{21}$	$p_{22}$	$p_{23}$	...	$p_{2n}$
	Classe 3	$p_{31}$	$p_{32}$	$p_{33}$	...	$p_{3n}$
	:	:	:	:	..	...
	Classe n	$p_{n1}$	$p_{n2}$	$p_{n3}$	...	$p_{nn}$

Para avaliar o desempenho dos modelos de classificação, foram utilizados os métodos *holdout* e validação cruzada. Segundo (TAN; STEINBACH; KUMAR, 2009), é útil fazer a comparação de diferentes classificadores para descobrir qual atua melhor em um determinado conjunto de dados. Uma das maneiras, é através da avaliação do intervalo de confiança quanto à acurácia. Para essa abordagem, utilizou-se a Equação 4.5.

$$\frac{2 \times N \times acc + Z_{\alpha/2}^2 \pm Z_{\alpha/2} \sqrt{Z_{\alpha/2}^2 + 4Nacc - 4Nacc^2}}{2(N + Z_{\alpha/2}^2)} \quad (4.5)$$

Onde  $Z_{\alpha/2}$  é obtido a partir de uma distribuição normal em nível de confiança  $(1 - \alpha)$ ,  $\alpha$  representa a significância,  $N$  é a quantidade de registros de teste e  $acc$  a acurácia.

### 4.2.1 Avaliação de agrupamentos

Uma medida tradicional de separação entre grupos, quando a proximidade é medida pela distância Euclidiana, é a soma de quadrados do grupo intermediário (SSB), que é a soma da distância quadrada de um centroide de grupo,  $c_i$ , até o centroide geral de todos os pontos de dados  $c$ . A soma de todos os valores de SSB, resulta na SSB Total, definida na Equação 4.6, onde  $c_i$  é o centroide de índice  $i$ ,  $c$  é o centroide geral do conjunto de dados e  $m_i$  é o número de objetos do *cluster* de índice  $i$  (TAN; STEINBACH; KUMAR, 2009).

$$SSB \text{ Total} = \sum_{i=1}^K m_i dist(c_i, c)^2 \quad (4.6)$$

Quanto maior for a SSB Total de um agrupamento, maior a separação *intercluster* dos grupos.

## 4.3 Os experimentos realizados

Para fins de demonstração do funcionamento do ml.js foram realizados alguns experimentos com os *datasets* apresentados na seção 4.1. Para alguns conjuntos de dados, a exemplo do *dataset GPS Trajectories*, foram aplicados mais de um modelo de AM.

### 4.3.1 k-NN

Na Tabela 8 é possível ver os resultados obtidos pelo modelo k-NN aplicado ao *dataset GPS Trajectories*. O experimento consistiu em dez execuções do modelo com variação do valor de  $k$  de 1 até 19. O melhor resultado obtido foi para  $k = 13$ , conseguindo classificar corretamente 41 das 53 amostras de teste. A curva de evolução do valor da acurácia é mostrada na Figura 39.

Tabela 8 – Métricas obtidas pela aplicação do modelo k-NN para diferentes valores de  $k$  com a base GPS Trajectories (autoria própria)

<b>k</b>	<b>Acurácia</b>	<b>Medida-F</b>	<b>Precisão</b>	<b>Sensibilidade</b>
1	0.5660	0.5660	0.5769	0.5556
3	0.5283	0.4681	0.5000	0.4400
5	0.5094	0.5517	0.4706	0.6667
7	0.6038	0.5532	0.5200	0.5909
9	0.6981	0.6923	0.6667	<b>0.7200</b>
11	0.6415	0.5778	0.5417	0.6190
<b>13</b>	<b>0.7736</b>	<b>0.7273</b>	<b>0.7619</b>	0.6957
15	0.6038	0.5333	0.7500	0.4138
17	0.6226	0.5455	0.7059	0.4444
19	0.6415	0.6275	0.6400	0.6154
Média	0.6189	0.5843	0.6134	0.5761
Variância	0.0055	0.0054	0.0101	0.0109

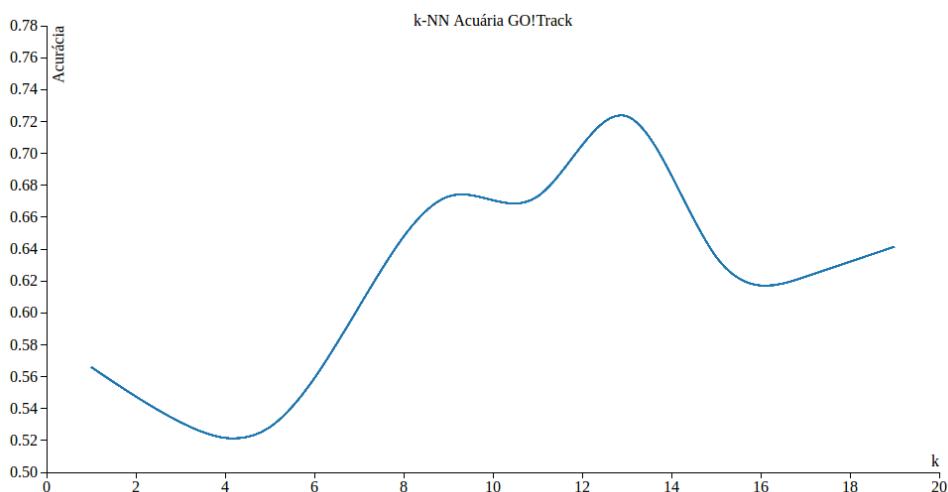


Figura 39 – Gráfico  $k \times$  Acurácia para os valores da Tabela 8 (autoria própria)

Em outro experimento, também com o conjunto de dados GPS Trajectories, uma execução separada do modelo utilizando a técnica *holdout*, resultou em uma acurácia de 88% quando avaliado sobre 53 registros de teste, para  $k = 13$ . Com esse resultado, foi obtido um intervalo de confiança para a acurácia verdadeira em nível de confiança de 95% entre 76,6 e 92,9.

### 4.3.2 ID3

A Figura 40 mostra a árvore de decisão obtida a partir da execução usando a técnica de validação cruzada com 10-folds. Os resultados obtidos podem ser vistos na Tabela 9. O melhor modelo apresentou 88,2% de acurácia e está representado na Figura 40.

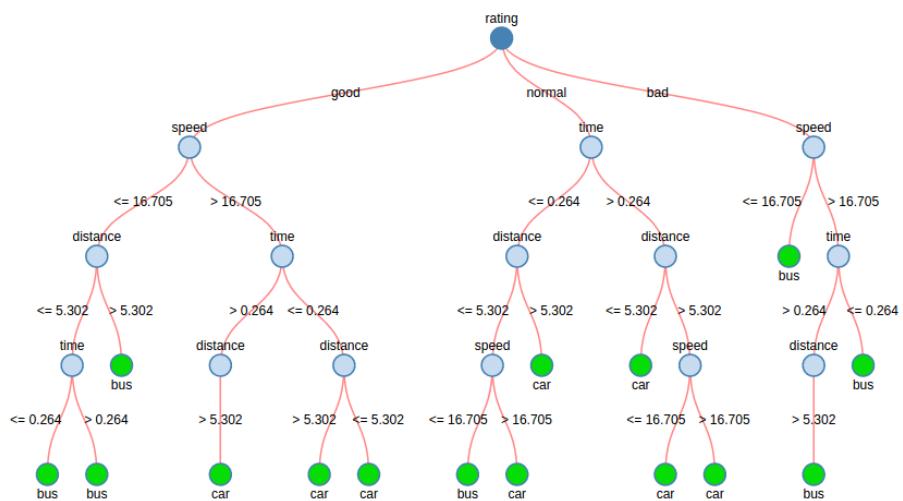


Figura 40 – Árvore de decisão gerada pelo modelo ID3 aplicado à base GPS Trajectories (autoria própria)

Tabela 9 – Métricas obtidas pela aplicação do ID3 com validação cruzada usando 10 partições à base GPS Trajectories (autoria própria)

Partição	Acurácia	Medida-F	Precisão	Sensibilidade
1	0.7647	0.8000	0.8000	0.8000
2	0.5882	0.6316	0.5455	0.7500
3	0.7059	0.7619	0.8000	0.7273
4	0.5882	0.5882	0.4545	0.8333
5	0.7647	0.7500	<b>0.8571</b>	0.6667
6	0.8750	<b>0.8571</b>	0.8571	0.8571
7	0.7059	0.7059	0.8571	0.6000
8	<b>0.8824</b>	0.8333	0.7143	<b>1.0000</b>
9	0.7647	0.7143	0.7143	0.7143
10	0.7000	0.4000	0.5000	0.3333
Média	0.7340	0.7042	0.7100	0.7282
Variância	0.0090	0.0166	0.0218	0.0284

Utilizando o mesmo conjunto de dados e a técnica *holdout*, o ID3 conseguiu uma acurácia de 86,0% quando avaliado sobre um conjunto de 53 registros de teste e 110 para treinamento do *dataset GPS Trajectories*, resultando em um intervalo de confiança entre 74,2 e 92,9 para um nível de confiança de 95%.

No experimento com a base de refugiados, utilizando a técnica *holdout*, os resultados obtidos são mostrados na Tabela 10. Com esses resultados, o ID3 obteve um intervalo de confiança com um nível de 95% entre 0,35 e 0,37. Um intervalo bastante estreito em comparação com os experimentos anteriores.

Tabela 10 – Resultados experimento usando o ID3 com a base de refugiados

Classificados corretamente	4236	36.344%
Classificados incorretamente	7405	63.534%
Não classificados	14	0.120%
Total exemplos de teste	11655	100%

A Figura 41 mostra uma árvore de decisão construída pelo ID3 para um conjunto de registro da base de refugiados. Dado a lista de atributos Sexo, Faixa etária, Estado Civil e Continente, o modelo classifica para qual estado foi feito o pedido de refúgio.

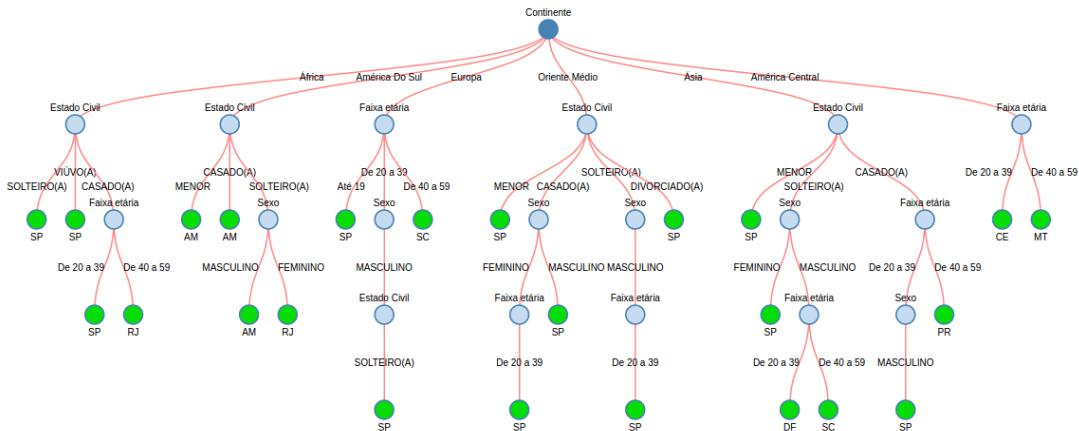


Figura 41 – Árvore de decisão gerada pelo modelo ID3 aplicado à base de imigrantes (autoria própria)

### 4.3.3 k-Means

Na Figura 42, é possível ver os grupos encontrados pela aplicação do k-Means ao *dataset t4.8k*. Como foi citado na Subseção 2.3.3, o k-Means apresenta dificuldade ao se deparar com bases que apresentam forma não circular, o que notoriamente pode ser visto na Figura 42. A Tabela 11 mostra os valores para a métrica SSBTotal do experimento. O menor valor encontrado foi para  $k = 28$  que corresponde a Figura 42d. A escolha dos centroides iniciais feita de maneira aleatória.

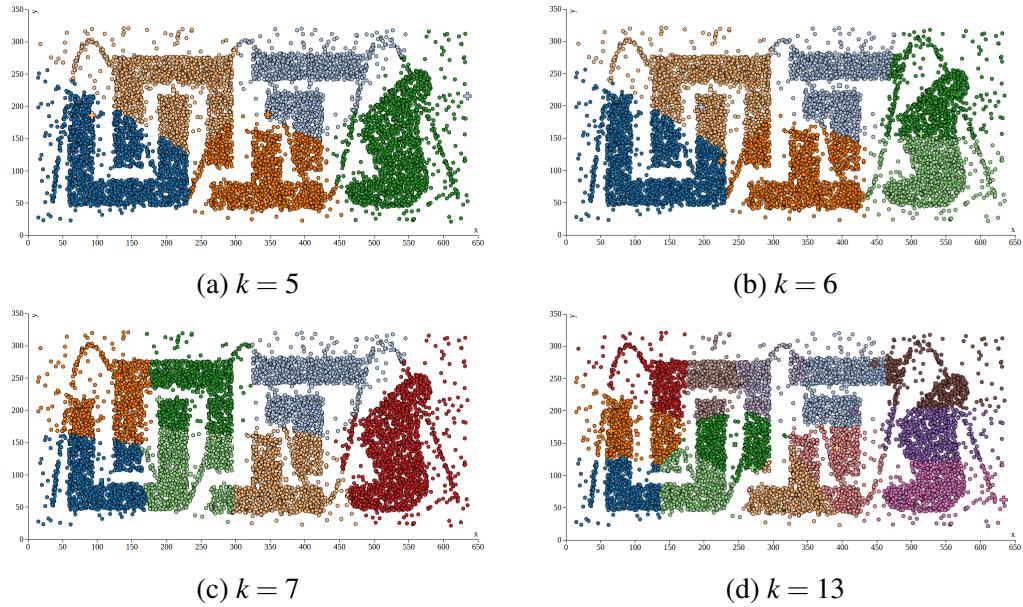


Figura 42 – Demonstração do k-Means aplicado ao *dataset* t4.8k para diferentes valores de  $k$  (autoria própria)

Tabela 11 – Métricas obtida pelo k-Means para diferentes valores de  $k$  aplicado ao *dataset* t4.8k

<b>k</b>	<b>Número de iterações</b>	<b>SSBTotal</b>
5	39	2.24
6	26	2.71
7	28	2.15
13	68	4.59

A Figura 43 mostra o resultado da aplicação do k-Means aos conjuntos de dados apresentados na Tabela 4. Nota-se que os grupos encontrados são diferentes dos reais grupos dos conjuntos de dados.

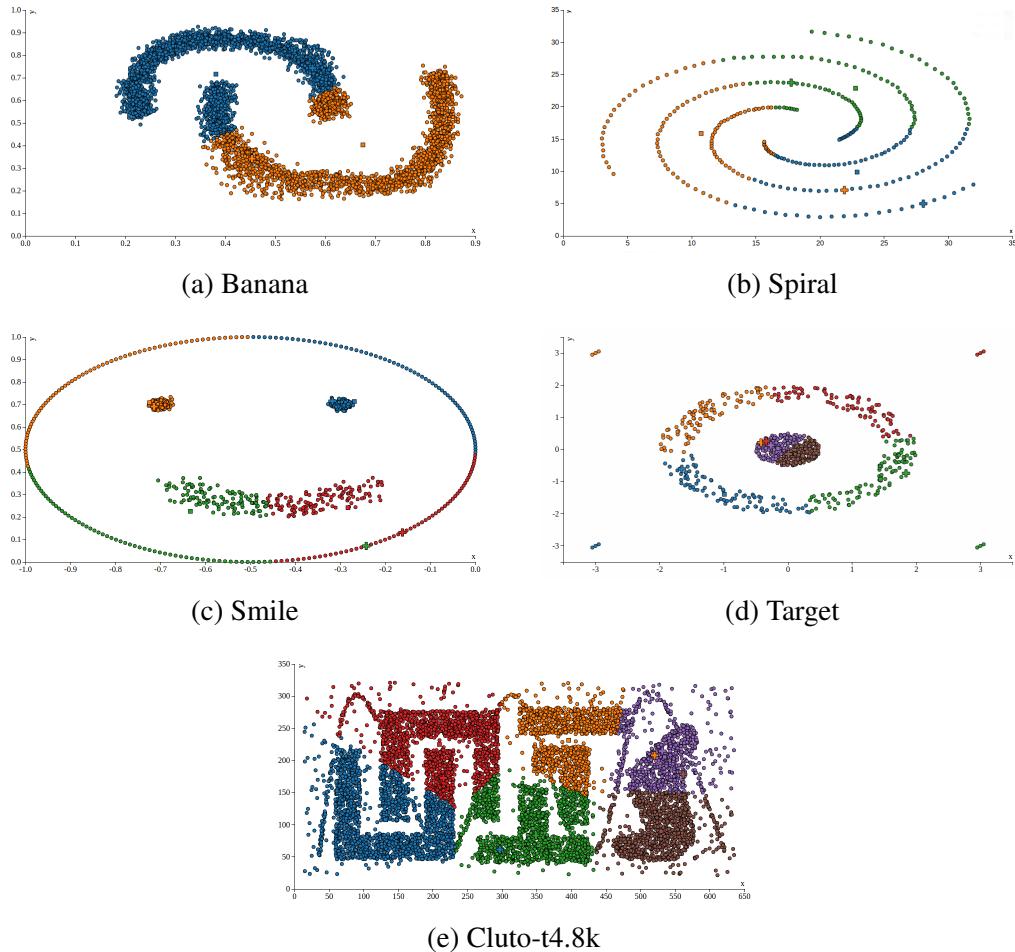


Figura 43 – Demonstração do k-Means aplicado aos *datasets* da Tabela 4 para o valor de  $k$  igual ao seu número de classes (autoria própria)

#### 4.3.4 DBSCAN

Um dos experimentos realizados com o DBSCAN foi sua aplicação ao *dataset* t4.8k, um dos *benchmarks* para atividades de agrupamento. Os grupos resultantes são mostrados na Figura 44. O resultado que mais se aproximou do agrupamento visivelmente natural, foi obtido com os valores 22 e 10 para *minPts* e *Eps* respectivamente. A Figura 44d ilustra esse resultado.

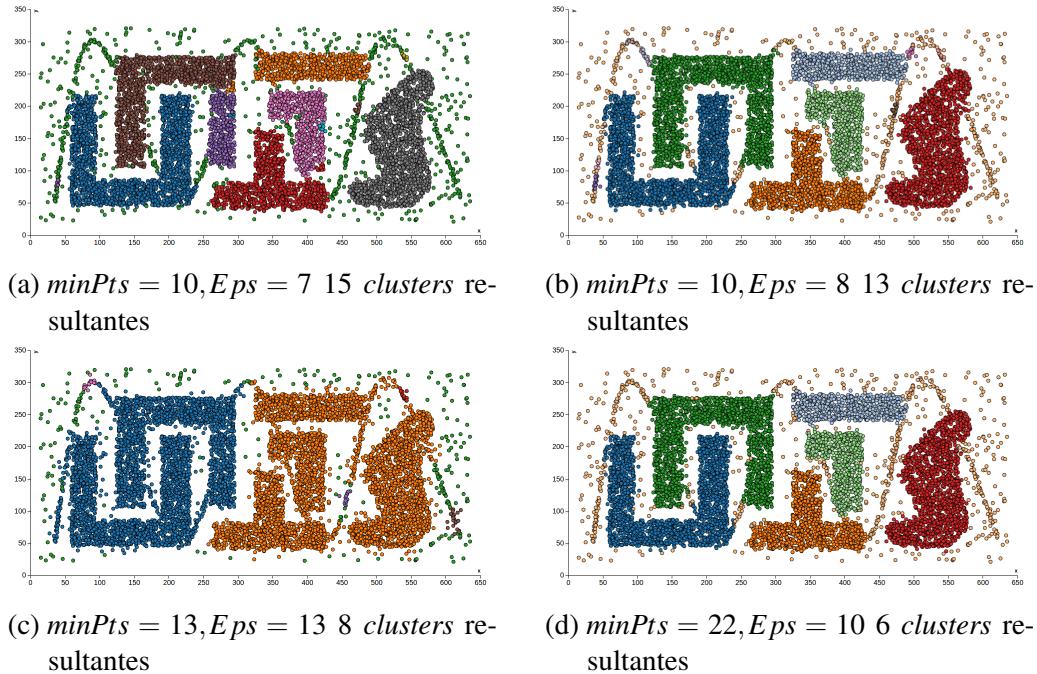


Figura 44 – Demonstração do algoritmo DBSCAN aplicado ao *dataset* t4.8k para diferentes valores de  $minPts$  e  $Eps$  (autoria própria)

Os resultados obtidos por DBSCAN quando aplicado às bases apresentadas na Tabela 4 são mostrados na Figura 45. Percebe-se que o modelo conseguiu encontrar os grupos esperados em cada base.

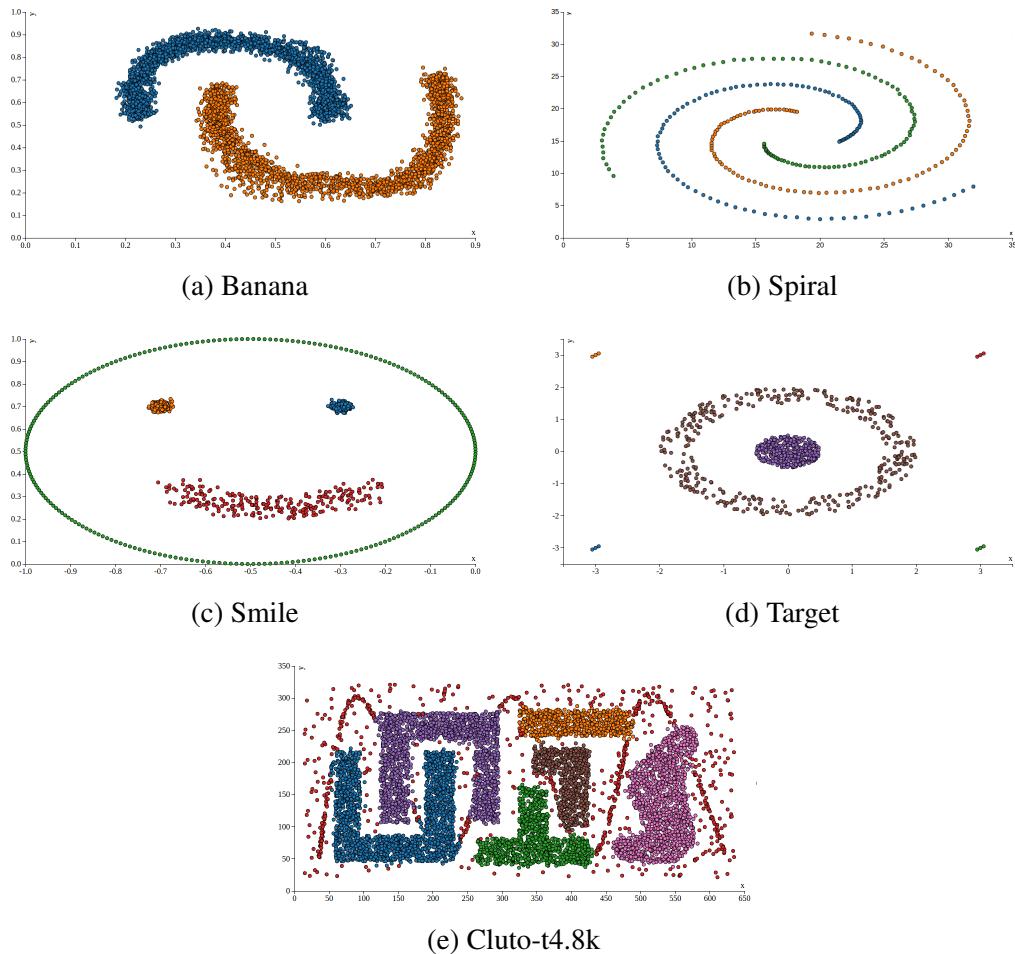


Figura 45 – Demonstração do DBSCAN aplicado aos *datasets* da Tabela 4 (autoria própria)

### 4.3.5 Gradiente descendente

A partir do conjunto de dados do quadro de medalhas e número de atletas de cada país, participante das olimpíadas 2016, foi realizado um experimento com o Gradiente descendente. A Equação 4.7 foi a resultante do treinamento do modelo. Segundo o modelo, o Brasil teria conquistado 52.14 medalhas contra as 19 realmente conquistadas, um número bastante diferente. Para os Estados Unidos o modelo previu 64.84 medalhas, contra as 121 conquistadas.

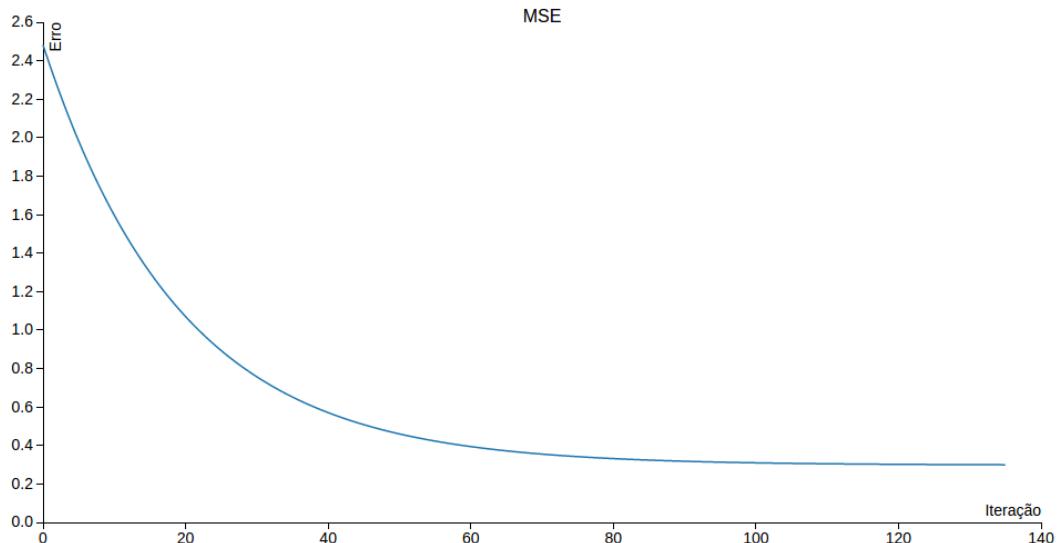
$$\text{medalhas} = 0.5502752 * \text{atletas} + 0.1343021 \quad (4.7)$$

O mesmo experimento foi realizado com o *software* WEKA, para comparação dos resultados por causa da grande diferença nos valores encontrados. A Tabela 12 traz mais resultados desse experimento. As colunas ml.js e WEKA representam o número de medalhas preditas por cada ferramenta respectivamente. A Figura 46 ilustra a evolução do erro médio quadrático para o experimento ao longo das iterações, o valor do erro final foi 0.299.

Tabela 12 – Predições de medalhas para regressão linear resultantes do ml.js e WEKA

País	Atletas	ml.js	WEKA	Valor real
Brasil	465	52.14	53.90	19
China	413	45.89	47.50	70
Estados Unidos	554	62.82	64.84	121
Grã-Bretanha	366	40.25	41.72	67
Jamaica	68	4.47	5.00	11
Japão	338	36.89	38.27	41

Vale ressaltar que o experimento levou em conta o número total de atletas de cada país. Os esportes coletivos contam apenas com uma medalha e existem atletas que conquistaram mais de uma medalha. Esses fatos interferem nos resultados. Outra observação é que os valores da base foram escalados para diminuir a diferença nos intervalos dos valores.

Figura 46 – Gráfico da evolução do MSE para experimento com Gradiente descendente e *dataset* número de medalhas e atletas olímpicas 2016 (autoria própria)

# 5

## Conclusão

Este trabalho descreve um *framework* para uso de modelos de aprendizagem de máquina na linguagem JavaScript. Uma interface gráfica que possibilita o uso dos modelos implementados de forma simples e direta sem a necessidade de programação também foi desenvolvida.

Foram selecionadas diferentes bases de dados para avaliar a aplicabilidade do *framework* e da ferramenta gráfica. Resultados confirmaram a corretude de ambos.

Espera-se que os resultados obtidos e a ferramenta desenvolvida nesse trabalho possam contribuir com o processo de ensino-aprendizagem da disciplina Aprendizado de Máquina e afins, auxiliando professores e alunos como fonte de pesquisa e ferramenta didática. Além disso, apresentou-se uma breve discussão sobre *Open Data* e experimentos com bases de dados do mesmo, mostrando que é possível trabalhar com dados de diferentes domínios com o ml.js. Com isso espera-se também que o interesse pelo assunto seja despertado.

Uma limitação presente quanto ao uso do modelo Gradiente descendente está relacionada ao tamanho dos dados de entrada. É preferível que sejam usados atributos com pequeno intervalo de valores. Como o *framework* usa valores de 32-bits, para alguns valores grandes dos atributos do conjunto de dados pode ocorrer um *overflow* numérico ocasionando assim, erros ao avaliar a condição de convergência do modelo. Uma possível solução para esse problema seria utilizar números de 64-bits ao invés de 32-bits assim com, o Google fez com o contador de visualizações do YouTube, quando o vídeo do cantor Psy esgotou a quantidade de números possíveis para um inteiro de 32-bits (SAVOV, 2014). A Figura 47 mostra um gráfico número de iterações × valor do MSE. Nota-se que a curva está no lado contrário. O erro cresce ao longo das iterações. Para o exemplo em questão foram usados dados de entrada com valores em torno de 50. Outra forma de contornar essa limitação é utilizando uma escala para tornar os valores menores. Por exemplo, uma escala *min-max* com intervalo entre 0 e 1.

Outro fator importantíssimo que se deve atentar é o navegador a ser utilizado. Levando

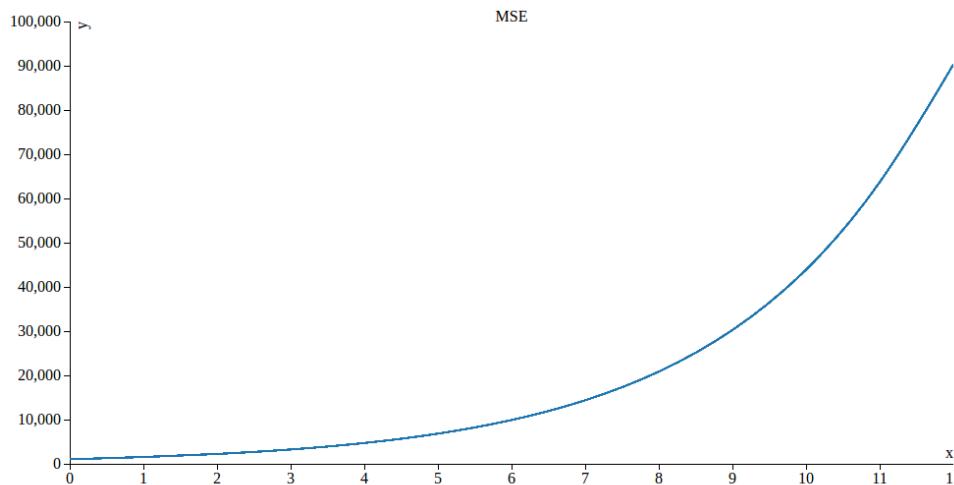


Figura 47 – Exemplo gráfico MSE para um conjunto de dados com valores muito altos (autoria própria)

em consideração que todo o processo de desenvolvimento foi realizado utilizando Google Chrome, este é o mais recomendado. Demais navegadores e versão que dão suporte as tecnologias utilizadas são listados na Tabela 13.

Tabela 13 – Navegadores e versão mínima para uso do ml.js

Navegador	Firefox	IE	Opera	Safari	Edge
Versão mínima	10	10	12	9	13

Sugestões para trabalhos futuros incluem: (i) adição de outros modelos de aprendizado de máquina, (ii) inclusão de método para tratamento de exemplos com atributos ausentes, (iii) opção para escalar atributos, (iv) opção para remoção de exemplos duplicados. A nível de interface gráfica, é interessante (i) a possibilidade de usar conjunto de dados de teste a partir de um arquivo diferente do *dataset* de treinamento, (ii) separação da interface gráfica em módulos possibilitando que seja montada de forma personalizada, (iii) adicionar opções de tratamento do *dataset*, (iv) realizar um estudo de caso com a utilização da ferramenta em uma disciplina ligada à aprendizagem de máquina, (v) desenvolver e adicionar ao *framework* uma forma de acompanhar a evolução dos modelos durante o treinamento e um *debugger*.

# Referências

- AGHA, M. E.; ASHOUR, W. M. Efficient and fast initialization algorithm for k-means clustering. *International Journal of Intelligent Systems and Applications*, Modern Education and Computer Science Press, v. 4, n. 1, p. 21, 2012. (Acessado em 17/09/2016). Disponível em: <<http://www.mecs-press.org/ijisa/ijisa-v4-n1/IJISA-V4-N1-3.pdf>>. Citado na página 43.
- AGRELA, L. *Apple apresenta relógio inteligente para natação*. 2016. (Acessado em 17/09/2016). Disponível em: <<http://exame.abril.com.br/tecnologia/noticias/apple-apresenta-relogio-inteligente-para-natacao>>. Citado na página 15.
- AL-HIARY, H. et al. Fast and accurate detection and classification of plant diseases. *Machine learning*, Citeseer, v. 14, p. 5, 2011. Citado na página 15.
- ALSHEIKH, M. A. et al. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 4, p. 1996–2018, 2014. Citado na página 15.
- ANGULAR.JS. *angular.js: HTML enhanced for web apps*. 2016. (Acessado em 17/09/2016). Disponível em: <<https://github.com/angular/angular.js>>. Citado na página 16.
- ATKESON, C. G.; MOORE, A. W.; SCHAAL, S. Locally weighted learning for control. In: *Lazy learning*. [S.l.]: Springer, 1997. p. 75–113. Citado na página 28.
- AUER, S. et al. Dbpedia: A nucleus for a web of open data. In: *The semantic web*. [S.l.]: Springer, 2007. p. 722–735. Citado na página 62.
- BARTON, T. *Clustering benchmarks*. 2016. <<https://github.com/deric/clustering-benchmark>>. (Acessado em 29/06/2016). Citado 2 vezes nas páginas 57 e 63.
- BENOIT, K. *Ordinary Least Squares Regression*. 2010. <[http://www.kenbenoit.net/courses/quant1/Quant1\\_Week8\\_OLS.pdf](http://www.kenbenoit.net/courses/quant1/Quant1_Week8_OLS.pdf)>. (Acessado em 11/09/2016). Citado na página 28.
- BERKHIN, P. A survey of clustering data mining techniques. In: *Grouping multidimensional data*. Springer, 2006. p. 25–71. (Acessado em 15/09/2016). Disponível em: <<https://pdfs.semanticscholar.org/26f1/78dbb00630ce19ccb9840ea12dbe31801be.pdf>>. Citado na página 29.
- BIDELMAN, E. *Reading files in JavaScript using the File APIs*. 2010. <<http://www.html5rocks.com/pt/tutorials/file/dndfiles/>>. (Acessado em 02/03/2016). Citado na página 54.
- BOOTSTRAP. *Bootstrap ü The world's most popular mobile-first and responsive front-end framework*. 2015. (Acessado em 16/05/2015). Disponível em: <<http://getbootstrap.com/>>. Citado na página 59.
- BOOTSTRAP. *Dashboard Template for Bootstrap*. 2015. (Acessado em 16/05/2015). Disponível em: <<http://getbootstrap.com/examples/dashboard/>>. Citado na página 59.
- BORGES, C. *RankBrain: aprenda tudo sobre o novo algoritmo do Google*. 2016. (Accessed on 09/11/2016). Disponível em: <<http://marketingdeconteudo.com/rankbrain/>>. Citado na página 15.

- BRAGA, L. P. V. B. *Introdução à Mineração de Dados-2a edição: Edição ampliada e revisada.* [S.l.]: Editora E-papers, 2005. Citado na página 48.
- BRASIL. *Brasil alcança 12º lugar em ranking mundial de dados abertos.* Portal Brasileiro de Dados Abertos, 2015. (Acessado em 03/03/2016). Disponível em: <<http://dados.gov.br/noticia/brasil-alanca-12o-lugar-em-ranking-mundial-de-dados-abertos/>>. Citado na página 69.
- BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. (Acessado em 17/09/2016). Disponível em: <<https://www.cs.princeton.edu/courses/archive/fall07/cos402/readings/bagging.pdf>>. Citado na página 30.
- BREIMAN, L. et al. *Classification and regression trees.* 1984. The Wadsworth Statistics/Probability Series. Belmont, California: Wadsworth International Group, a Division of Wadsworth, Inc. X, 358 p. \$ 29.25; \$ 18.95 (1984). Citado na página 29.
- BROWNLEE, J. *A Tour of Machine Learning Algorithms.* 2013. (Acessado em 02/20/2016). Disponível em: <<http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/>>. Citado 3 vezes nas páginas 8, 28 e 31.
- BROWNLEE, J. *K-Nearest Neighbors for Machine Learning - Machine Learning Mastery.* 2016. <<http://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/>>. (Acessado em 13/09/2016). Citado na página 34.
- CASSIANO, K. M. *ANÁLISE DE SÉRIES TEMPORAIS USANDO ANÁLISE ESPECTRAL SINGULAR (SSA) E CLUSTERIZAÇÃO DE SUAS COMPONENTES BASEADA EM DENSIDADE.* Tese (Doutorado) — PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO, 2014. Disponível em: <<http://dx.doi.org/10.17771/PUCRio.acad.24787>>. Citado 3 vezes nas páginas 8, 44 e 46.
- CÁSSIO, E. *Desenvolva jogos com HTML5 Canvas e JavaScript.* [S.l.]: Casa do Código, 2014. ISBN 978-85-66250-38-1. Citado na página 53.
- CHANG, H.; YEUNG, D.-Y. Robust path-based spectral clustering. *Pattern Recognition*, v. 41, n. 1, p. 191 – 203, 2008. ISSN 0031-3203. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0031320307002038>>. Citado na página 63.
- CHIGNARD, S. *A brief history of Open Data.* Paris Tech Review, 2013. (Acessado em 03/01/2016). Disponível em: <<http://www.paristechreview.com/2013/03/29/brief-history-open-data/>>. Citado na página 66.
- CLEOPHAS, T.; ZWINDERMAN, A. *Machine Learning in Medicine.* Springer Netherlands, 2013. (Machine Learning in Medicine, v. 1). ISBN 9789400758247. Disponível em: <<https://books.google.com.br/books?id=MQhAAAAAQBAJ>>. Citado na página 15.
- COLTON, S. *Lecture 11 Decision Tree Learning.* 2004. Disponível em: <<http://www.doc.ic.ac.uk/~sgc/teaching/pre2012/v231/lecture11.html>>. Citado 2 vezes nas páginas 11 e 39.
- CORTEZ, P. et al. Lamb meat quality assessment by support vector machines. *Neural Processing Letters*, Springer, v. 24, n. 1, p. 41–51, 2006. Citado na página 15.
- CRUZ, M. O. et al. Grouping similar trajectories for carpooling purposes. In: IEEE. *2015 Brazilian Conference on Intelligent Systems (BRACIS).* [S.l.], 2015. p. 234–239. Citado na página 63.

- DELBEM, A. C.; CARVALHO, A. P. L. de; BRETAS, N. Main chain representation for evolutionary algorithms applied to distribution system reconfiguration. *IEEE Transactions on Power Systems*, IEEE, v. 20, n. 1, p. 425–436, 2005. Citado na página 15.
- DIAZ, D.; HARMES, R. *Pro JavaScript Design Patterns: The Essentials of Object-Oriented JavaScript Programming*. [S.l.]: Apress, 2007. ISBN 159059908X. Citado na página 59.
- DREISEITL, S.; OHNO-MACHADO, L. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, Elsevier, v. 35, n. 5, p. 352–359, 2002. Citado na página 28.
- EAVES, D. *The Three Laws of Open Government Data | eaves.ca*. 2009. (Acessado em 02/11/2016). Disponível em: <<http://eaves.ca/2009/09/30/three-law-of-open-government-data/>>. Citado na página 66.
- ECMA, I. *ECMAScript 2015 Language Specification ECMA-262 6th Edition*. 2015. <<http://www.ecma-international.org/ecma-262/6.0/index.html>>. (Acessado em 27/05/2016). Citado na página 53.
- ERTÖZ, L.; STEINBACH, M.; KUMAR, V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: SIAM. *SDM*. [S.l.], 2003. p. 47–58. Citado na página 43.
- ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: . [S.l.]: AAAI Press, 1996. p. 226–231. Citado 3 vezes nas páginas 29, 44 e 46.
- FACELI, K. et al. *Inteligencia artificial: Uma Abordagem de Aprendizado de Máquina*. [S.l.]: LTC, 2011. ISBN 9788521618805. Citado 6 vezes nas páginas 8, 20, 23, 27, 33 e 36.
- FERRAIOLLO, J. *Scalable Vector Graphics (SVG) 1.0 Specification*. [S.l.], 2001. (Acessado em 28/03/2016). Disponível em: <<http://www.w3.org/TR/2001/REC-SVG-20010904>>. Citado na página 55.
- FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, Wiley Online Library, v. 7, n. 2, p. 179–188, 1936. Citado na página 63.
- FREUND, Y.; SCHAPIRE, R. E. *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*. 1995. (Acessado em 17/09/2016). Disponível em: <<http://www.math.tau.ac.il/~mansour/ml-course/ada-boost.ps.gz>>. Citado na página 30.
- FREUND, Y.; SCHAPIRE, R. E. Large margin classification using the perceptron algorithm. *Machine Learning*, v. 37, n. 3, p. 277–296, 1999. ISSN 1573-0565. Disponível em: <<http://dx.doi.org/10.1023/A:1007662407062>>. Citado na página 29.
- FRIEDMAN, J. H. Multivariate adaptive regression splines. *The annals of statistics*, JSTOR, p. 1–67, 1991. Citado na página 28.
- FRIZZARINI, C. *Algoritmo para indução de árvores de classificação para dados desbalanceados*. Tese (Doutorado) — Universidade de São Paulo, 2013. Citado na página 69.
- GABASOVA, E. *Spice up your website with machine learning!* 2016. (Acessado em 17/09/2016). Disponível em: <<https://www.infoq.com/br/presentations/apimente-seu-site-com-machine-learning>>. Citado na página 15.

- GAMA, J. Árvores de decisão. *Palestra ministrada no Núcleo da Ciência de Computação da Universidade do Porto, Porto*, 2002. (Acessado em 11/09/2016). Disponível em: <[http://www.dcc.fc.up.pt/~ines/aulas/MIM/arvores\\_de\\_decisao.pdf](http://www.dcc.fc.up.pt/~ines/aulas/MIM/arvores_de_decisao.pdf)>. Citado na página 35.
- GARCIA, S. C. *O uso de árvores de decisão na descoberta de conhecimento na área da saúde*. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, 2003. Citado 2 vezes nas páginas 35 e 65.
- GOODFELLOW, I. J. et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013. Citado na página 15.
- GO!TRACK. *Go!Track*. 2015. Disponível em: <<https://play.google.com/store/apps/details?id=com.go.router>>. Citado na página 63.
- GRINSTEAD, B. *filereader.js - A Lightweight FileReader Wrapper*. 2010. <<http://bgrins.github.io/filereader.js/>>. Citado na página 54.
- GUPTA, M. R.; CHEN, Y. *Theory and use of the EM algorithm*. Now Publishers Inc, 2011. (Acessado em 13/09/2016). Disponível em: <<http://mayagupta.org/publications/EMbookGuptaChen2010.pdf>>. Citado na página 29.
- HALL, P.; PARK, B. U.; SAMWORTH, R. J. Choice of neighbor order in nearest-neighbor classification. *The Annals of Statistics*, JSTOR, p. 2135–2152, 2008. (Acessado em 11/09/2016). Disponível em: <<http://projecteuclid.org/euclid.ao/1223908087>>. Citado na página 33.
- HANDL, J.; KNOWLES, J. Multiobjective clustering with automatic determination of the number of clusters. *UMIST, Manchester, Tech. Rep. TR-COMPSYSBIO-2004-02*, 2004. Citado na página 63.
- HEATON, J. *JavaScript Machine Learning and Neural Networks with Encog*. 2012. <<http://www.codeproject.com/Articles/477689/JavaScript-Machine-Learning-and-Neural-Networks-wi>>. (Acessado em 28/05/2016). Citado 3 vezes nas páginas 8, 16 e 17.
- HECKERMAN, D.; GEIGER, D.; CHICKERING, D. M. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, v. 20, n. 3, p. 197–243, 1995. ISSN 1573-0565. Disponível em: <<http://dx.doi.org/10.1007/BF00994016>>. Citado na página 29.
- HO, T. K. Random decision forests. In: IEEE. *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. [S.I.], 1995. v. 1, p. 278–282. Citado na página 30.
- HOTHORN, T.; HORNIK, K.; ZEILEIS, A. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, Taylor & Francis, v. 15, n. 3, p. 651–674, 2006. Citado na página 29.
- HTIKE, Z. Z.; WIN, S. L. Recognition of promoters in dna sequences using weightily averaged one-dependence estimators. *Procedia Computer Science*, v. 23, p. 60 – 67, 2013. ISSN 1877-0509. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1877050913011447>>. Citado na página 15.

- IBA, W.; LANGUAGE, P. Induction of one-level decision trees. In: *Proceedings of the ninth international conference on machine learning*. [S.l.: s.n.], 1992. p. 233–240. Citado na página 29.
- INTERNATIONAL, O. K. *Why Open Data?* 2011. <<http://opendatahandbook.org/guide/en/why-open-data/>>. (Acessado em 16/09/2016). Citado na página 66.
- JACOBY, W. G. Loess: a nonparametric, graphical tool for depicting relationships between variables. *Electoral Studies*, Elsevier, v. 19, n. 4, p. 577–613, 2000. Disponível em: <<http://www.med.upenn.edu/beat/docs/Jacoby2000.pdf>>. Citado na página 28.
- JAIN, A. K. Data clustering: 50 years beyond k-means. *ECML/PKDD (1)*, v. 5211, p. 3–4, 2008. (Acessado em 17/09/2016). Disponível em: <<http://ce.sharif.edu/courses/89-90/2/ce725-1/resources/root/Readings/JainDataClusteringPRL09.pdf>>. Citado na página 29.
- JAIN, A. K. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, Elsevier, v. 31, n. 8, p. 651–666, 2010. (Acessado em 15/09/2016). Disponível em: <[http://www.inf.ufpr.br/lesoliveira/padroes/data\\_clustering.pdf](http://www.inf.ufpr.br/lesoliveira/padroes/data_clustering.pdf)>. Citado na página 43.
- JAITLEY, N. et al. Application of pretrained deep neural networks to large vocabulary speech recognition. In: *Proceedings of Interspeech 2012*. [S.l.: s.n.], 2012. Citado na página 15.
- JOHNSON, J. *K-Nearest Neighbors*. 2013. <<https://shapeofdata.wordpress.com/2013/05/07/k-nearest-neighbors/>>. (Acessado em 12/09/2016). Citado na página 33.
- JQUERY. *jQuery write less, do more*. 2015. (Acessado em 16/05/2015). Disponível em: <<https://jquery.com/>>. Citado na página 59.
- JUNQUEIRA, M. *Smart offices: otimizando o uso de salas com sensores conectados e Machine Learning*. 2016. (Acessado em 03/09/2016). Disponível em: <<https://www.infoq.com/br/presentations/smart-offices-otimizando-o-uso-de-salas-com-sensores>>. Citado na página 15.
- JUSTIÇA, M. da. *Comitê Nacional para os Refugiados*. 2016. (Acessado em 11/06/2016). Disponível em: <<http://dados.mj.gov.br/dataset/comite-nacional-para-os-refugiados>>. Citado na página 69.
- KANBER, B. *Machine Learning in Javascript: Introduction*. 2012. <<https://www.burakkanber.com/blog/machine-learning-in-other-languages-introduction/>>. (Acessado em 28/05/2016). Citado na página 17.
- KARPATY, A. *ConvNetJS: Deep Learning in your browser*. 2014. <<http://cs.stanford.edu/people/karpathy/convnetjs/index.html>>. (Acessado em 28/05/2016). Citado na página 16.
- KARYPIS, G.; HAN, E.-H.; KUMAR, V. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, IEEE, v. 32, n. 8, p. 68–75, 1999. Citado na página 63.
- KOHONEN, T. Learning vector quantization. In: \_\_\_\_\_. *Self-Organizing Maps*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997. p. 203–217. ISBN 978-3-642-97966-8. Disponível em: <[http://dx.doi.org/10.1007/978-3-642-97966-8\\_6](http://dx.doi.org/10.1007/978-3-642-97966-8_6)>. Citado na página 28.
- KRIEG, M. L. A tutorial on bayesian belief networks. 2001. Citado na página 29.

- LABHACKER, L. B. d. C. D.; NIC.BR, N. d. I. e. C. d. P. B.; W3C BRASIL, W. W. W. C. E. B. *Manual dos dados abertos: governo*. 2011. Disponível em: <[http://www.w3c.br/pub/Materiais/PublicacoesW3C/Manual\\_Dados\\_Abertos\\_WEB.pdf](http://www.w3c.br/pub/Materiais/PublicacoesW3C/Manual_Dados_Abertos_WEB.pdf)>. Citado 4 vezes nas páginas 11, 66, 67 e 68.
- LANDAU, D. M. *Inteligência artificial e aprendizado de máquina: como os computadores aprendem - Intel iQ Brasil*. 2016. <<https://iq.intel.com.br/inteligencia-artificial-e-aprendizado-de-maquina-como-os-computadores-aprendem/>>. (Acessado em 06/09/2016). Citado na página 15.
- LEITE, P. T.; CARNEIRO, A. A. F. M.; CARVALHO, A. Energetic operation planning using genetic algorithms. *IEEE Transactions on Power Systems*, IEEE, v. 17, n. 1, p. 173–179, 2002. Citado na página 15.
- LICHMAN, M. *UCI Machine Learning Repository*. 2013. Disponível em: <<http://archive.ics.uci.edu/ml>>. Citado 2 vezes nas páginas 55 e 63.
- LOWD, D.; DOMINGOS, P. *Naive Bayes models for probability estimation*. 2005. (Acessado em 17/09/2016). Disponível em: <<http://homes.cs.washington.edu/~pedrod/papers/mlc05b.pdf>>. Citado na página 29.
- MACEDO, H. *Notas de aulas em Aprendizado de Máquina*. 2014. Citado 4 vezes nas páginas 8, 21, 36 e 44.
- MACLEAN, M. *D3 Tips and Tricks - Interactive Data Visualization in a Web Browser*. [S.l.]: Leanpub, 2015. <<http://leanpub.com/D3-Tips-and-Tricks>>. Citado na página 55.
- MARKOFF, J. Google cars drive themselves, in traffic. *The New York Times*, v. 10, n. A1, p. 9, 2010. Citado na página 15.
- MATTEUCCI, M. *Clustering - Hierarchical*. 2003. (Acessado em 16/09/2016). Disponível em: <[http://home.deib.polimi.it/matteucci/Clustering/tutorial\\_html/hierarchical.html](http://home.deib.polimi.it/matteucci/Clustering/tutorial_html/hierarchical.html)>. Citado na página 29.
- MILIDIÚ, R. L. *Aprendizado de Máquina para o Problema de Sentiment Classification*. Tese (Doutorado) — PUC-Rio, 2006. Citado na página 29.
- MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. Citado na página 21.
- MOTTA, T. *Recomendação de ponta a ponta na Globo.com*. 2016. (Acessado em 17/09/2016). Disponível em: <<https://www.infoq.com/br/presentations/recomendacao-de-ponta-a-ponta-na-globo>>. Citado na página 15.
- NEDRICH, M. *An Introduction to Gradient Descent and Linear Regression*. 2014. <<https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/>>. (Acessado em 17/09/2016). Citado na página 49.
- NG, A. *Machine learning: Lecture i. Supervised Learning*. [S.l.]: Stanford University, Coursera platform, 2012. <<https://class.coursera.org/ml-005/lecture/3>>. (Acessado em 21/05/2016). Citado 5 vezes nas páginas 8, 9, 24, 48 e 49.
- NG, A. *Machine learning: Lecture iv. Linear Regression with Multiple Variables*. [S.l.]: Stanford University, Coursera platform, 2012. <<https://class.coursera.org/ml-005/lecture/20>>. (Acessado em 21/05/2016). Citado 5 vezes nas páginas 9, 11, 28, 50 e 51.

- NG, A. *Gradient Descent For Linear Regression*. Stanford University, Coursera platform, 2016. (Acessado em 17/09/2016). Disponível em: <<https://pt.coursera.org/learn/machine-learning/lecture/kCvQc/gradient-descent-for-linear-regression>>. Citado 3 vezes nas páginas 9, 48 e 49.
- NODE.JS. *About | Node.js*. 2016. (Acessado em 17/09/2016). Disponível em: <<https://nodejs.org/en/about/>>. Citado na página 16.
- OLIVEIRA, P. H. M. A. *Detecção de fraudes em cartões: um classificador baseado em regras de associação e regressão logística*. Tese (Doutorado) — Universidade de São Paulo, 2016. Citado na página 15.
- PETERSON, L. E. K-nearest neighbor. *Scholarpedia*, v. 4, n. 2, p. 1883, 2009. Citado na página 28.
- POZZER, C. T. *Aprendizado por Árvores de Decisão*. 2006. <[http://www-usr.inf.ufsm.br/~pozzer/disciplinas/pj3d\\_decisionTrees.pdf](http://www-usr.inf.ufsm.br/~pozzer/disciplinas/pj3d_decisionTrees.pdf)>. (Acessado em 11/09/2016). Citado na página 35.
- QUINLAN, J. R. Induction of decision trees. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 1, n. 1, p. 81–106, mar. 1986. ISSN 0885-6125. Disponível em: <<http://dx.doi.org/10.1023/A:1022643204877>>. Citado 2 vezes nas páginas 29 e 34.
- QUINLAN, J. R. *C4. 5: programs for machine learning*. [S.l.]: Elsevier, 2014. Citado na página 29.
- RADIS. Novidade no diagnóstico de tb. *RADIS Comunicação e Saúde*, n. 103, p. 7, mar. 2011. (Acessado em 17/09/2016). Disponível em: <[http://www6.ensp.fiocruz.br/radis/sites/default/files/103/radis\\_103.pdf](http://www6.ensp.fiocruz.br/radis/sites/default/files/103/radis_103.pdf)>. Citado na página 15.
- RANGANATHAN, A.; SICKING, J. W3C Working Draft, *File API*. 2015. <<https://www.w3.org/TR/FileAPI/>>. (Acessado em 27/05/2016). Citado na página 54.
- RESIG, J.; BIBEAULT, B. *Segredos do Ninja JavaScript*. [S.l.]: Novatec Editora, 2013. Citado na página 54.
- RIO2016. *Atletas Olímpicos*. 2016. (Acessado em 23/08/2016). Disponível em: <<http://www.rio2016.com/atletas>>. Citado na página 69.
- RIO2016. *Medalhas Olímpicas*. 2016. (Acessado em 23/08/2016). Disponível em: <<http://www.rio2016.com/quadro-de-medalhas-paises>>. Citado na página 69.
- RUSSELL, S.; NORVIG, P. *Inteligência Artificial (Em Portuguese do Brasil)*. [S.l.]: CAMPUS - GRUPO ELSEVIER, 2013. ISBN 8535237011. Citado na página 27.
- SANTOS, E. B. dos et al. Bayesian networks for obstacle classification in agricultural environments. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. [S.l.: s.n.], 2013. p. 1416–1421. ISSN 2153-0009. Citado na página 15.
- SATHASIVAM, S.; ABDULLAH, W. A. T. W. Logic learning in hopfield networks. *arXiv preprint arXiv:0804.4075*, 2008. Citado na página 29.
- SAVOV, V. *Gangnam Style broke YouTube's view counter*. 2014. <<http://www.theverge.com/2014/12/3/7325819/gangnam-style-broke-youtube-view-counter>>. (Acessado em 29/05/2016). Citado na página 80.

- SCHAPIRE, R. E. The boosting approach to machine learning an overview. In: *MSRI Workshop on Nonlinear Estimation and Classification*. [s.n.], 2002. Disponível em: <[https://www.cs.princeton.edu/picasso/mats/schapire02boosting\\_schapire.pdf](https://www.cs.princeton.edu/picasso/mats/schapire02boosting_schapire.pdf)>. Citado na página 30.
- SCHMIDHUBER, J. Deep Learning. *Scholarpedia*, v. 10, n. 11, p. 32832, 2015. revision 152272. Citado na página 29.
- SCHWENKER, F.; KESTLER, H. A.; PALM, G. Three learning phases for radial-basis-function networks. *Neural networks*, Elsevier, v. 14, n. 4, p. 439–458, 2001. Citado na página 29.
- SHAHINFAR, S. et al. Prediction of insemination outcomes in holstein dairy cattle using alternative machine learning algorithms. *Journal of Dairy Science*, v. 97, n. 2, p. 731 – 742, 2014. ISSN 0022-0302. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0022030213008059>>. Citado na página 15.
- SIMEONE, F.; FERREIRA, I. A. Árvore de decisão com algoritmo ID3 - Trabalho desenvolvido para a disciplina Mineração de dados - Mestrado em Ciência da Computação - DCC - UFLA. 2014. <<http://www.fernandosimeone.com.br/id3/>>. (Acessado em 28/05/2016). Citado 4 vezes nas páginas 8, 17, 18 e 58.
- SIMON, P. *Too Big to Ignore: The Business Case for Big Data*. Wiley, 2013. (Wiley and SAS Business Series). ISBN 9781118642108. Disponível em: <<https://books.google.com.br/books?id=Dn-Gdoh66sgC>>. Citado na página 20.
- SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 45, n. 4, p. 427–437, jul. 2009. ISSN 0306-4573. (Acessado em 01/04/16). Disponível em: <<http://dx.doi.org/10.1016/j.ipm.2009.03.002>>. Citado na página 71.
- SOL, A. *Real-World Machine Learning: How Kinect Gesture Recognition Works*. 2013. (Acessado em 11/09/2016). Disponível em: <<https://channel9.msdn.com/Events/Build/2013/3-704>>. Citado na página 15.
- SPANGLER, S. et al. Automated hypothesis generation based on mining scientific literature. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2014. (KDD '14), p. 1877–1886. ISBN 978-1-4503-2956-9. Disponível em: <<http://doi.acm.org/10.1145/2623330.2623667>>. Citado na página 15.
- STACKOVERFLOW. *Stack Overflow Developer Survey 2016 Results*. 2016. <<http://stack overflow.com/research/developer-survey-2016>>. (Acessado em 27/05/2016). Citado na página 16.
- TAN, P. ning; STEINBACH, M.; KUMAR, V. *Introdução ao datamining : mineracao de dados*. Rio de Janeiro (RJ: Ciencia Moderna, 2009. ISBN 978-85-7393-761-9. Citado 5 vezes nas páginas 8, 33, 42, 45 e 71.
- TELECOMMUNICATIONS, V. M.; METSIS, V. Spam filtering with naive bayes – which naive bayes? In: *Third Conference on Email and Anti-Spam (CEAS. [S.l.: s.n.], 2006. Citado na página 29.*

- THRUN, S. et al. Winning the darpa grand challenge. *Journal of Field Robotics*, 2006. Citado na página 15.
- TIOBE. *TIOBE Index for May 2016*. 2016. <[http://www.tiobe.com/tiobe\\_index](http://www.tiobe.com/tiobe_index)>. (Acessado em 27/05/2016). Citado na página 16.
- TRIOLA, M. F. et al. *Introdução à estatística*. [S.l.]: Ltc Rio de Janeiro, 2005. v. 10. Citado na página 51.
- ULTSCH, A. Clustering wih som: U\* c. In: *Proceedings of the 5th Workshop on Self-Organizing Maps*. [S.l.: s.n.], 2005. v. 2, p. 75–82. Citado na página 63.
- ULTSCH, A. *Fundamental Clustering Problem Suite*. 2016. (Acessado em 29/06/2016). Disponível em: <<http://www.uni-marburg.de/fb12/datenbionik/data>>. Citado na página 63.
- VALE, M. N. do. *Agrupamentos de dados: Avaliação de Métodos e Desenvolvimento de Aplicativo para Análise de Grupos*. Tese (Doutorado) — PUC-Rio, 2005. (Acessado em 11/09/2016). Disponível em: <<http://doi.org/10.17771/PUCRio.acad.7975>>. Citado na página 34.
- WEBB, I. G.; BOUGHTON, R. J.; WANG, Z. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, v. 58, n. 1, p. 5–24, 2005. ISSN 1573-0565. Disponível em: <<http://dx.doi.org/10.1007/s10994-005-4258-6>>. Citado na página 29.
- WEKA. *Weka 3: Data Mining Software in Java*. 2015. (Acessado em 18/04/2015). Disponível em: <<http://www.uni-marburg.de/fb12/datenbionik/data>>. Citado na página 15.
- WIECKI, T. *GLM: Linear regression*. 2015. (Acessado em 08/12/2015). Disponível em: <<https://pymc-devs.github.io/pymc3/notebooks/GLM-linear.html>>. Citado 2 vezes nas páginas 8 e 26.
- WIKIPEDIA. *2016 Summer Olympics — Wikipedia, The Free Encyclopedia*. 2016. (Acessado em 23/08/2016). Disponível em: <[https://en.wikipedia.org/w/index.php?title=2016\\_Summer\\_Olympics&oldid=738812550](https://en.wikipedia.org/w/index.php?title=2016_Summer_Olympics&oldid=738812550)>. Citado na página 69.
- WILKINSON, L. Tree structured data analysis: Aid, chaid and cart. *Retrieved February*, v. 1, p. 2008, 1992. Citado na página 29.
- WILSON, D. R.; MARTINEZ, T. R. Improved heterogeneous distance functions. *Journal of artificial intelligence research*, v. 6, p. 1–34, 1997. Citado na página 33.
- ZHANG, H. The optimality of naive bayes. *AA*, v. 1, n. 2, p. 3, 2004. (Acessado em 15/09/2016). Disponível em: <<http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf>>. Citado na página 29.

# **Apêndices**

# APÊNDICE A – Manual do ml.js

Para a utilização do ml.js é necessário a inclusão dos arquivos ao seu projeto como mostrado no exemplo de Código 4.

Código 4 – Exemplo de inclusão do ml.js a um projeto

```

1  <!-- dependências externas ml.js -->
2  <!-- geração dos gráficos e leitura de arquivos-->
3  <script src="mljs/lib/d3.min.js"></script>
4  <script src="mljs/lib/filereader.js"></script>
5
6  <!-- dependências ml.js classe e funções auxiliares-->
7  <script src="mljs/utils.js"></script>
8  <script src="mljs/dataSet.js"></script>
9  <script src="mljs/confusionMatrix.js"></script>
10
11 <!-- modelos -->
12 <script src="mljs/ml.models.js"></script>
```

Caso não queira utilizar todos os modelos, adicione separadamente os necessários como no exemplo de Código 5.

Código 5 – Exemplo de inclusão dos modelos do ml.js de forma separada

```

1  <!-- inclusão dos modelos de forma separada -->
2  <script src="mljs/ml.kmeans.js"></script>
3  <script src="mljs/ml.dbSCAN.js"></script>
4  <script src="mljs/ml.knn.js"></script>
5  <script src="mljs/ml.ID3.js"></script>
6  <script src="mljs/ml.linearRegressionGradientDescent.js"></script>
```

Nas seções que seguem são apresentados exemplos de uso para os modelos implementados.

## A.1 k-Means

Uma forma de uso simplificado do k-Means está ilustrada no trecho de Código 6.

## Código 6 – Exemplo de uso do k-Means

---

```

1 // Exemplo de uso do modelo kMeans
2 var model = new kMeans();
3 // O resultado é um array de clusters
4 clusterResult = model.buildClusterer({
5   "k": kValue, // valor do K
6   "data": myDataset.data // dados
7 });
8 // Retornando o valor do SSE
9 var sse = model.getSSE();

```

---

Onde *myDataset* é uma instância da classe *dataSet*, *idem* para os demais exemplos de código.

## A.2 DBSCAN

No trecho de Código 7 o uso do DBSCAN é ilustrado de forma simples.

## Código 7 – Exemplo de uso do DBSCAN

---

```

1 // Exemplo de uso do modelo DBSCAN
2 var model = new DBSCAN();
3 // O resultado é um array de clusters
4 clusterResult = model.buildClusterer({
5   "eps": epsValue, // valor do raio eps
6   "minPts": minPtsValue, // número mínimo de pontos
7   "data": myDataset.data // dados
8 });

```

---

## A.3 k-NN

O Código 8 trás um exemplo simples do uso do k-NN.

## Código 8 – Exemplo de uso do k-NN

---

```

1 var model = new kNN();
2 // O kNN tem como retorna uma matriz de confusão
3 var confusionMatrix = model.buildClassifier({
4   "k": i, // valor do K
5   "data": myDataset.data // dados
6 });

```

---

O trecho de Código 9 exemplifica a forma de definir uma função de distância personalizada, quando ela é omitida o modelo usa a distância Euclidiana. A função *myDist* definida na linha 2, pode ser usada para comparação de instância de um conjunto de dados não numérico, quando os valores de dois atributos comparados são exatamente iguais, nada é acrescentado à distância, caso contrário soma uma unidade. A função deve receber dois parâmetros e retorna um valor numérico que representa a distância entre os dos objetos. O k-Means e DBSCAN também aceitam esse parâmetro para a função de distância.

Código 9 – Exemplo de uso do k-NN com função de distância personalizada

---

```

1 // Exemplo de função para calcular distância
2 function myDist (x1, x2) {
3     var l = x1.length;
4     var distance = 0;
5     for(var i=0; i<l; i++) {
6         var dx = x1[i] === x2[i] ? 0 : 1;
7         distance += dx;
8     }
9     return distance;
10}
11
12 var model = new kNN();
13 //O kNN tem como retorna uma matriz de confusão
14 var confusionMatrix = model.buildClassifier({
15     "k": 1, // valor do K
16     "data": myDataset.data // dados
17     "distance": myDist // função de distância personalizada
18 });

```

---

## A.4 ID3

No Código 10 está ilustrado o uso do ID3 e como gerar uma visualização da árvore resultante do treinamento do modelo.

## Código 10 – Exemplo de uso do ID3

```

1 var model = new ID3();
2 //O ID3 tem como retorna uma matriz de confusão
3 var confusionMatrix = model.buildClassifier({
4     "data": myDataset.data, // dados
5     "attributesNames": myDataset.getAttributesNames() // nomes dos atributos do
    ↵ conjunto de dados, obtidos pelo método getAttributesNames
6     "targetAttribute": 1 // coluna do atributo alvo, se não for passado considera o
    ↵ último
7 });
8
9 //Retornando os dados da árvore resultante
10 var treeData = model.getTree();
11
12 //Método para gerar a visualização da árvore
13 drawTree({
14     "data": treeData, // objeto com os dados da árvore
15     "containerPlot": "tree-visualization" // id do local para o gráfico da árvore
16 });

```

## A.5 Gradiente descendente

Para a regressão existem duas classes, uma para regressão univariada e outra para a multivariada. Os trechos de Códigos 11 e 12 exemplificam o uso respectivamente.

## Código 11 – Exemplo de uso do Gradiente descendente para regressão univariada

```

1 var model = new univariateLinearRegressionGradientDescent();
2 model.train({
3     "data": myDataset.data, // dados
4     "alpha": alphaValue, // taxa de aprendizagem
5     "precision": precisionValue, // valor da precisão em relação ao erro
6     "maxIterations": max
7 });

```

### Código 12 – Exemplo de uso do Gradiente descendente para regressão multivariada

```

1 var model = new linearRegressionGradientDescent();
2 model.train({
3   "data": myDataset.data // dados
4   "y": yCol, // coluna alvo, se omitida será considerada a ultima posição do vetor
5   "alpha": alphaValue, // taxa de aprendizagem
6   "precision": precisionValue, // valor da precisão em relação ao erro
7   "maxIterations": max, // número máximo de iterações
8 });

```

O trecho de Código 13 ilustra a forma de obtenção de métricas para o Gradiente descendente para os dois casos.

### Código 13 – Exemplo de obtenção de resultados para o Gradiente descendente

```

1 // Retornando o número de iterações até a convergência
2 var numIterations = model.numIterations;
3 // Retornando o erro médio quadrático
4 var mse = model.mse;
5 // Retornando o vetor de pesos
6 var theta = model.theta;

```

## A.6 Classes auxiliares

Esta seção trás exemplos de uso das principais classes auxiliares do ml.js. A primeira delas é a classe para leitura e manipulação do conjunto de dados, o trecho de Código 14 mostra como fazer o carregamento de um arquivo.

### Código 14 – Exemplo de leitura de arquivo e execução de *callback* após o carregamento

```

1 var myDataset = new dataSet();
2 myDataset.loadFile({
3   "loadend": function() {
4     // função executada após o carregamento do arquivo
5     console.log('Arquivo carregado: ', myDataset.file.name);
6     console.log('Tamanho: ', myDataset.file.extra.prettySize);
7   }
8 });

```

Além do carregamento do *dataset* por *input* do tipo *file*, onde o usuário seleciona o arquivo, também é possível fazer o carregamento através de dados no formato de JSON, o que torna o uso da classe *dataset* mais flexível, por exemplo, fazer o uso de dados de alguma API

que forneça um retorno nesse formato ou a partir de um banco de dados. O Código 15 ilustra exemplifica o carregamento pelo meio de um objeto JSON que deve ser na forma apresentada no trecho de exemplo.

Código 15 – Exemplo carregamento conjunto de dados a partir de JSON

```
1 var dataJSON = {  
2     "relation": "Car fast", // nome do dataset  
3     "attributes": ["engine", "turbo", "weight", "fuel-eco", "fast"], // atributos  
4     "data": [  
5         ["small", "no", "average", "good", "no"],  
6         ["small", "no", "light", "average", "no"],  
7         ["small", "yes", "average", "bad", "yes"],  
8         ["medium", "no", "heavy", "bad", "yes"],  
9         ["large", "no", "average", "bad", "yes"],  
10        ["medium", "no", "light", "bad", "no"],  
11        ["large", "yes", "heavy", "good", "no"],  
12        ["large", "no", "heavy", "bad", "no"],  
13        ["medium", "yes", "light", "bad", "yes"]  
14    ] // dados  
15};  
16  
17 var myDataset = new dataSet();  
18 // Carregando dados a partir de objeto JSON  
19 myDataset.loadJSON(dataJSON);
```

O trecho de Código 16 exemplifica o uso da classe dataSet para modificações no conjunto de dados.

Código 16 – Trecho de código para transformações aplicadas ao *dataset GPS Trajectories*


---

```

1 // Função para trocar o nome da classe
2 function carOrBus (x) {
3     return x == "1" ? "car" : "bus"
4 };
5
6 // Função para trocar o nome da avaliação da viagem
7 function rating2str(x) {
8     return x == "3" ? "good" : (x == "2" ? "normal" : "bad");
9 }
10
11 var myDataset = new dataSet();
12 myDataset.loadFile({
13     // Após o carregamento do arquivo execute as seguintes operações
14     "loadend": function() {
15         // Remova as colunas 0,1,6,7,9
16         myDataset.removeAttribute([0,1,6,7,9])
17         // Converta as colunas 0,1,2 para forma nominal
18         .numericToNominalAttribute([0,1,2])
19         // Troque o nome dos atributos das colunas 3 e 4 de acordo com as funções
20         → rating2str e carOrBus
21         .replaceAttribute([3,4], [rating2str, carOrBus]);
22     }
23 });

```

---

No exemplo de Código 17 está ilustrado o uso da classe plot. Ela possui métodos para a geração de gráficos em SVG, no exemplo é feito o uso do k-Means para exemplificar como pode ser feito a visualização de um agrupamento.

### Código 17 – Exemplo de geração de visualização para resultados de agrupamentos

```
1 var modelKmeans = new kMeans();  
2 var clusterResult = modelKmeans.buildClusterer({  
3     "k": 5,  
4     "data": myDataset.data  
5});  
6  
7 var plotDataSet = new plot({  
8     "containerPlot": "svg-container", // id do local onde deve ser gerado o gráfico  
9     "title": "Exemplo plotClusters" // título do gráfico  
10 }).plotClusters({  
11     "data": clusterResult.clusters, // pontos do dataset em seus respectivos grupos  
12     "centroids": clusterResult.centroids, // array com os centroides finais  
13     "initialCentroids": clusterResult.initialCentroids, // array com os centroides  
14     ← iniciais  
15 });
```

---

Para avaliação dos modelos de classificação, foi implementada uma classe para representação da matriz de confusão, que possui métodos para cálculos de métricas e sua representação em forma de tabela em HTML. O Código 18 ilustra o seu uso.

## Código 18 – Exemplo de uso da classe ConfusionMatrix

```
1 // Classes do conjunto de dados
2 var labels = ["+", "-"];
3 // Criando uma instância de ConfusionMatrix
4 var cm = new ConfusionMatrix(labels);
5
6 // Informando os valores
7 cm.matrix["+"]["+"] = 8;
8 cm.matrix["+"]["-"] = 2;
9 cm.matrix["-"]["+"] = 16;
10 cm.matrix["-"]["-"] = 974;
11
12 // Calculando métricas
13 cm.calculate();
14
15 // Retornando representaçāo em tabela da matriz de confusāo e métricas
16 var str = cm.toString();
17
18 // Retornando o valor da acurácia
19 var acc = cm.accuracy;
20
21 // Retornando o valor da classificações corretas
22 var correct = cm.correctClassifications;
23
24 // Retornando o valor da classificações incorretas
25 var incorrect = cm.incorrectClassifications;
```

Outra classe de grande importância é a classe utils. Ela possui uma grande quantidade de métodos auxiliares, bastantes úteis para o *framework*, o trecho de Código 19 trás exemplos de alguns métodos.

## Código 19 – Exemplo de uso das funções da classe utils

```
1 var pointA = [1,0];
2 var pointB = [5,5];
3
4 // Distância Euclidiana
5 var distance = utils.math.euclidean(pointA, pointB);
6
7 // Distância Manhattan
8 var distance = utils.math.manhattan(pointA, pointB);
9
10 var data = [[1,2,3,10],[0,5,7,13],[8,3,1,4],[4,9,0,5]];
11
12 // Retornando a coluna 1 da matriz data
13 var col = utils.getCol(data, 1); // [2,5,3,9]
14
15 // Removendo a coluna 0 da matriz data
16 var data = utils.removeCol(data, 0);
17
18 // Retornando min,max,avg,median do array col
19 var statistics = utils.statistics(col);
20 // retorna um objeto na forma {min: 2, max: 9, average: 4.75, median: 4, stdDev:
→ 2.680951323690902}
```

# APÊNDICE B – Manual da interface gráfica

A tela inicial da interface gráfica é mostrada da Figura 48 e está dividida em cinco abas *Dashboard*, *Dataset*, Agrupamento, Classificação e Regressão. A aba *Dashboard* é destinada a apenas mostrar informações sobre a aplicação.

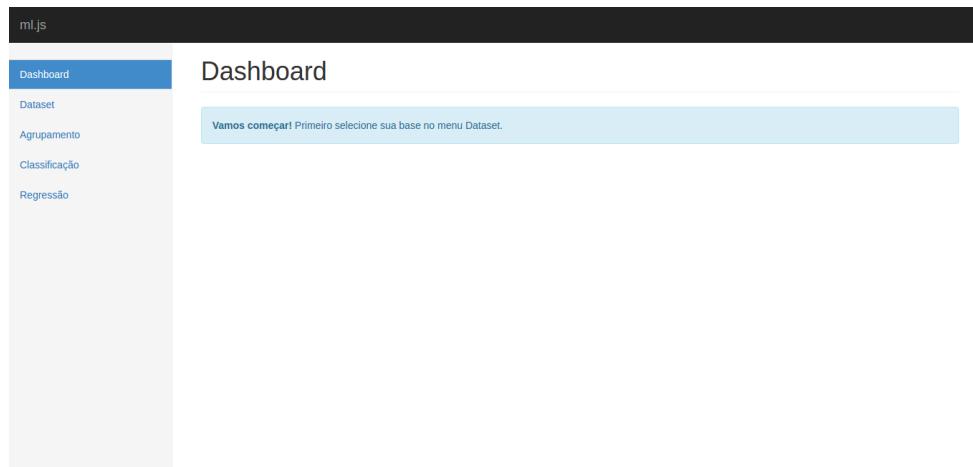


Figura 48 – Tela inicial da GUI

## B.1 Dataset

Nesta aba estão as opções para carregamento e visualização do arquivo que será usado como *dataset* por um dos modelos disponíveis.

Para mostrar a parte destinada a leitura e visualização de arquivo selecione o local destacado na Figura 49.

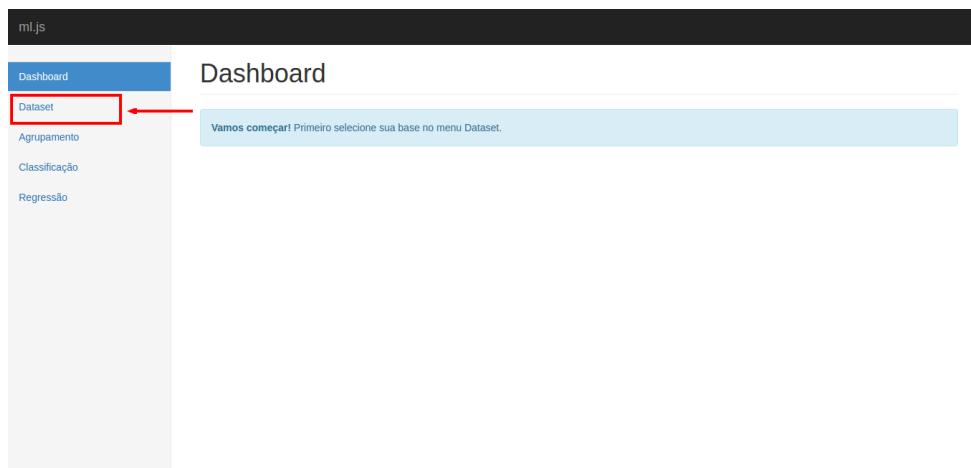


Figura 49 – Destaque menu

Em seguida será exibida a tela como na Figura 50, nela estão as opções de leitura, visualização e informações do arquivo que será carregado. Para fazer o carregamento, arraste e solte o arquivo no local indicado destacado na Figura 50 ou clique para selecionar manualmente, será exibido uma nova janela onde você deve selecionar o arquivo que deseja utilizar, como mostrado na Figura 51.

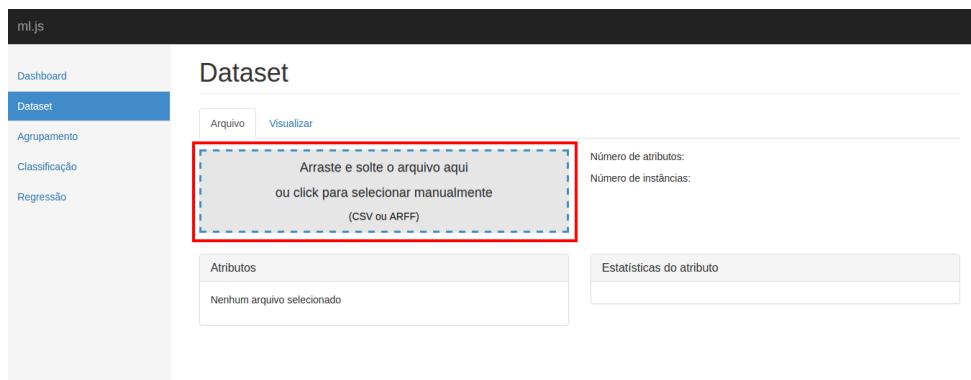


Figura 50 – Tela Dataset

Os formatos de arquivos suportados pela GUI são arquivos *arff* e *csv*, para o uso de arquivos de outros formatos deve-se usar a API.

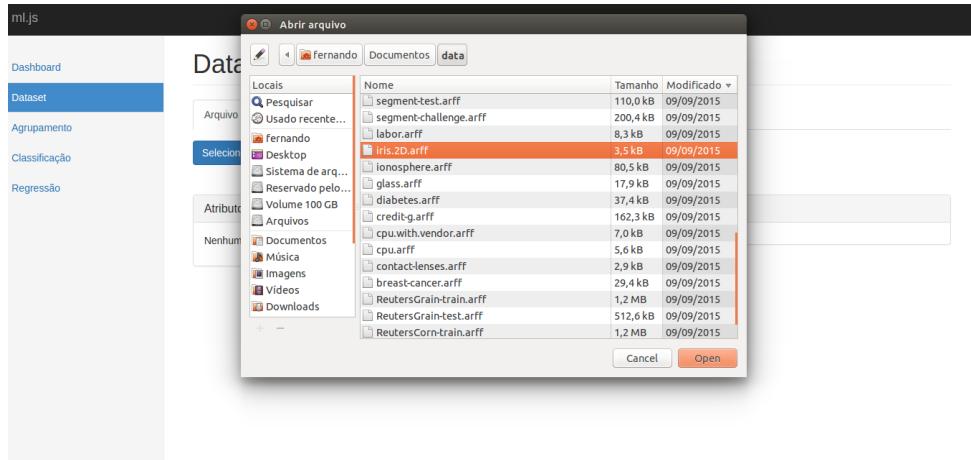


Figura 51 – Tela selecionando arquivo

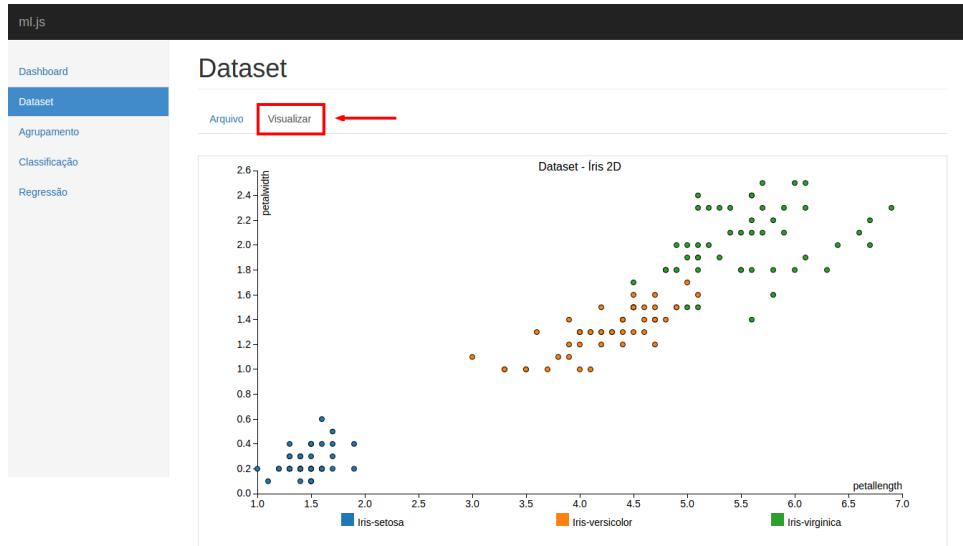
Após selecionado o arquivo, algumas informações serão mostradas da tela como pode ser observado na Figura 52. Quantidade de exemplos do arquivo, quantidade e lista dos atributos. Clicando no botão destacado da Figura 52, é possível ver mais detalhes de cada um dos atributos presentes no *dataset*.

Estatística	Valor
Minímo	1
Máximo	6.9
Média	3.7587
Mediana	4.35
Desvio padrão	1.7585

Figura 52 – Tela informações do *dataset*

Com o arquivo carregado, é possível visualizar seus valores em uma representação gráfica 2D clicando em Visualizar como está destacado na Figura 53.

No gráfico é mostrado os valores de cada atributo colocar o curso do *mouse* sobre os pontos. Abaixo do gráfico está a legenda com as classes do conjunto de dados selecionado. Clicando nos quadrados coloridos da legenda, é mostrado ou escondido os ponto de dados de cada uma das classes.

Figura 53 – Tela visualização do *dataset*

## B.2 Agrupamento

Para realizar a atividade de agrupamento, selecione a aba Agrupamento, então será exibida a tela como a Figura 54. Nela são mostrados os modelos disponíveis para a atividade, basta selecionar o desejado no local indicado na Figura 54.

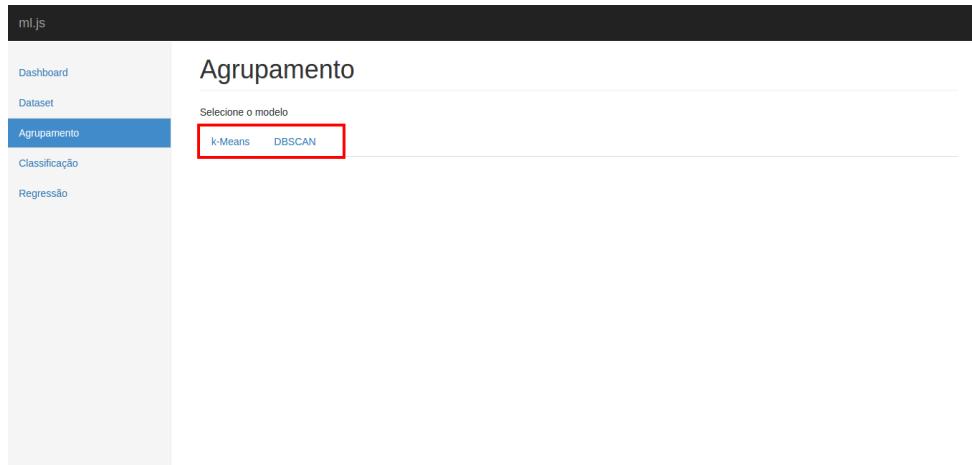


Figura 54 – Tela inicial para atividade de agrupamento

### B.2.1 k-Means

Seleciona a opção k-Means, preencha os parâmetros mostrados na Figura 55 valor de K e número máximo de iterações caso queira alterar o valor padrão do parâmetro e clique no botão Executar para realizar o agrupamento.

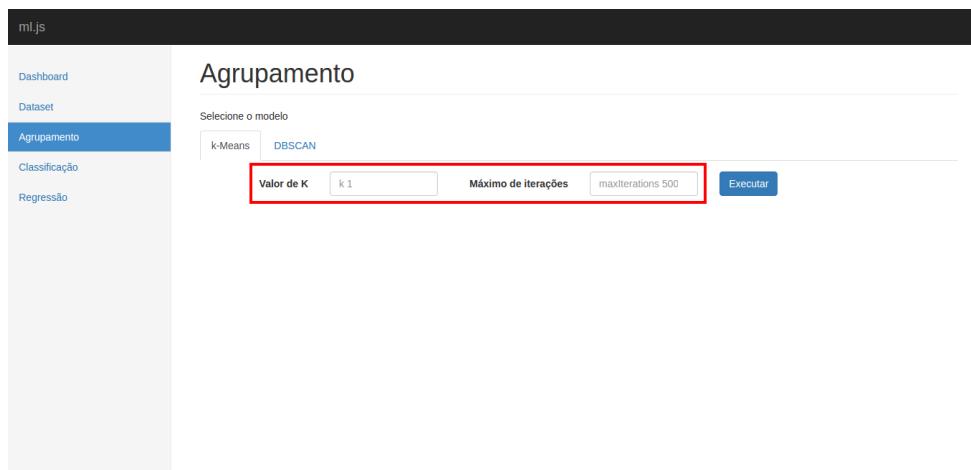


Figura 55 – Tela parâmetros k-Means

As métricas resultantes para o agrupamento do k-Means são mostradas na Figura 56, onde pode ser visto número de iterações realizadas, soma do erro quadrático e SSB total.

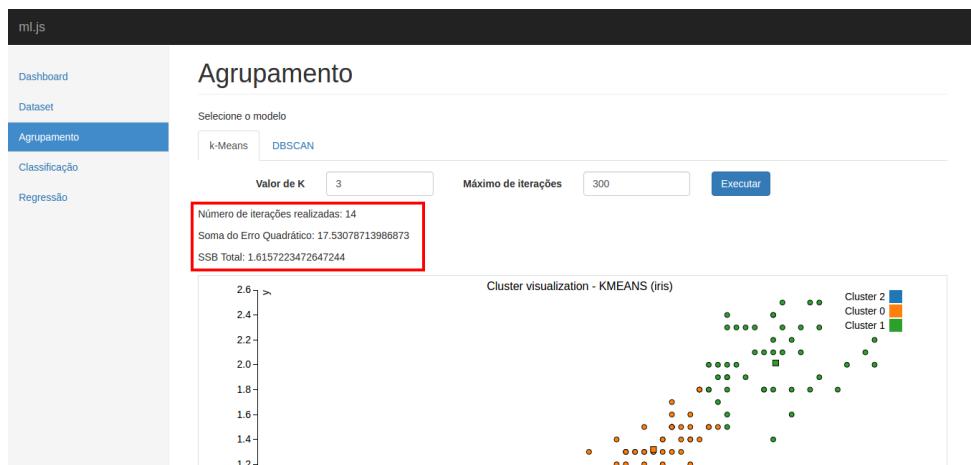


Figura 56 – Métricas para o modelo k-Means

A Figura 57 ilustra um resultado de agrupamento encontrado pelo modelo k-Means. Na legenda do lado direito da imagem estão listados os grupos e na legenda inferior a representação para o centroide final de cada grupo e o centroide inicial respectivamente.

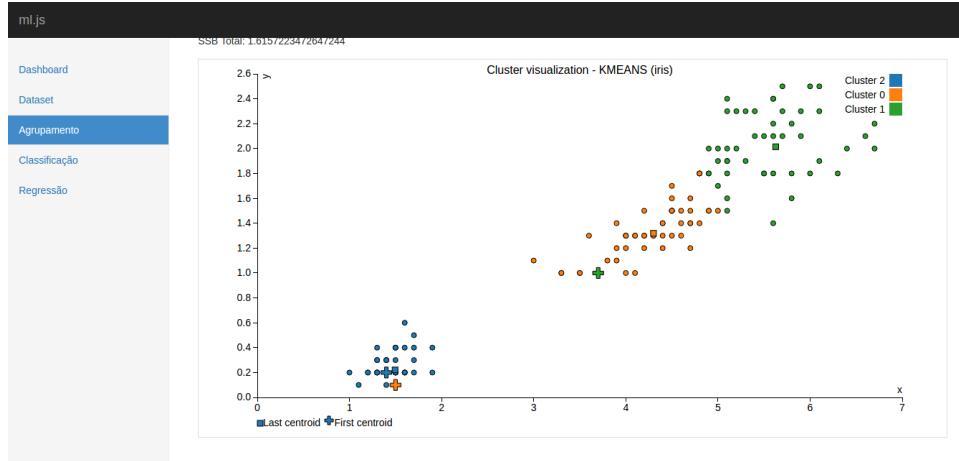


Figura 57 – Exemplo resultado de agrupamento modelo k-Means

### B.2.2 DBSCAN

Selecione a opção DBSCAN, preencha os parâmetros mostrados na Figura 58 número mínimo de pontos e o valor de épsilon e clique no botão Executar para realizar o agrupamento.

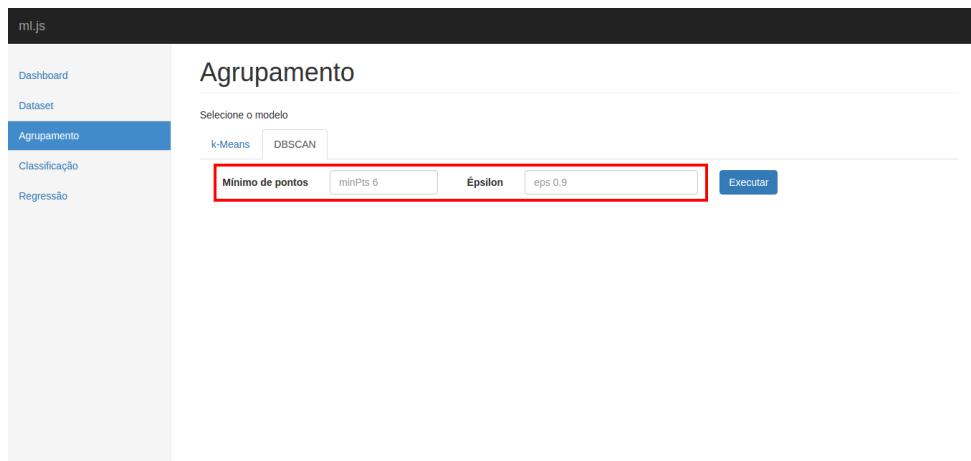


Figura 58 – Tela parâmetros DBSCAN

A Figura 59 ilustra um resultado de agrupamento encontrado pelo modelo DBSCAN. Na legenda do lado direito da imagem estão listados os grupos encontrados no agrupamento, caso algum ponto do conjunto de dados fossem considerado como ruído, seria mostrada mais uma linha na legenda com o rótulo *noise*.

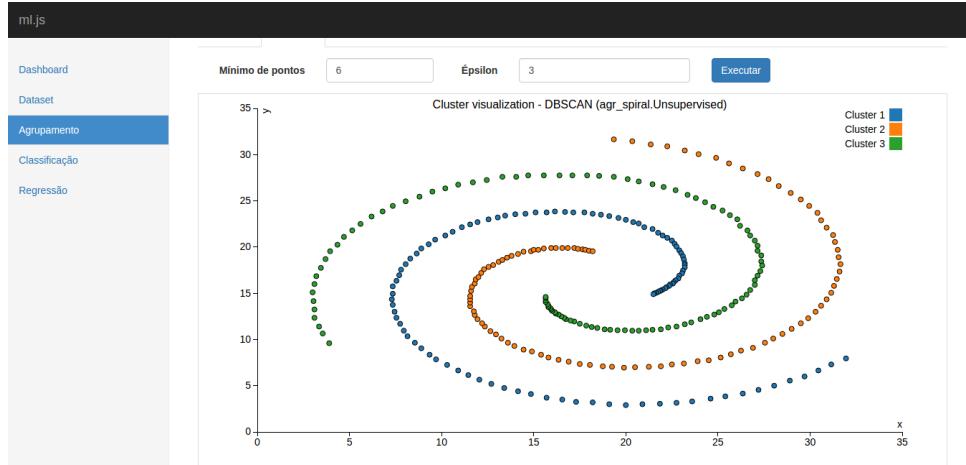


Figura 59 – Exemplo resultado de agrupamento modelo DBSCAN

## B.3 Classificação

Para realizar a atividade de classificação selecione a opção Classificação, em seguida será exibida na tela opções como mostrada na Figura 60. Seguindo o mesmo caminho da Subseção B.2 selecione o modelo que deseja utilizar.

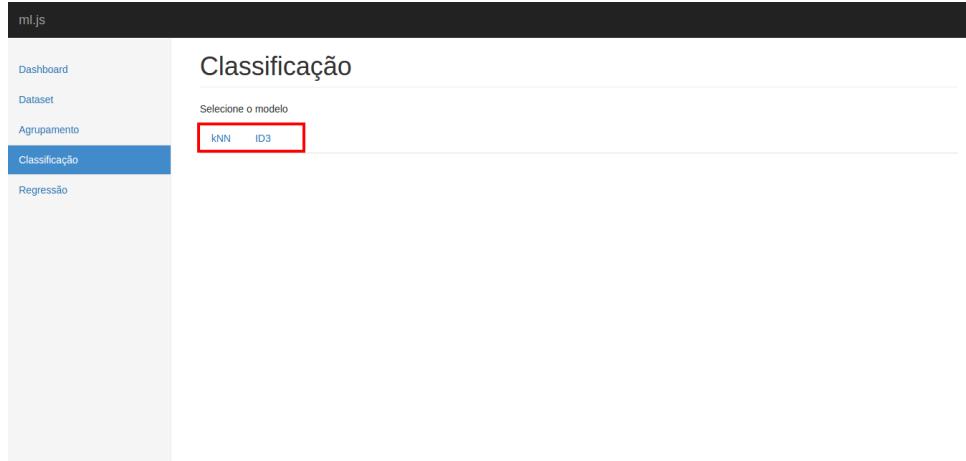


Figura 60 – Tela inicial para atividade de classificação

Os modelos de classificação, além dos seus próprios parâmetros, pode ser informado ainda a forma que será realizado o teste do modelo, a Figura 61 mostra essa opção. O teste do modelo pode ser realizado das seguintes formas:

**Porcentagem de treinamento** Nessa opção deve ser informado a porcentagem do conjunto de dados que deverá ser usada para fazer o treinamento do modelo, o restante será usado como dados de teste. A Figura 61a mostra essa opção, o valor padrão é 66 que corresponde a aproximadamente a  $2/3$  do dataset.

**Holdout** Para essa opção  $\frac{2}{3}$  do *dataset* serão usados como treinamento e  $\frac{1}{3}$  como teste, a Figura 61b ilustra essa opção.

**Validação cruzada** O modelo será avaliado usando validação cruzada, o número de partições padrão é 10 e pode ser alterado como mostrado na Figura 61c.

The screenshot shows the 'ml.js' interface with the 'Classificação' tab selected. In the 'Opções para teste' section, the radio button for 'Porcentagem de treinamento' is selected, and the input field contains the value '66'. The 'Executar' button is visible at the bottom right.

(a) Porcentagem de treinamento

The screenshot shows the 'ml.js' interface with the 'Classificação' tab selected. In the 'Opções para teste' section, the radio button for 'Holdout' is selected. The 'Executar' button is visible at the bottom right.

(b) Holdout

The screenshot shows the 'ml.js' interface with the 'Classificação' tab selected. In the 'Opções para teste' section, the radio button for 'Validação cruzada' is selected, and the input field for 'Número de Partições' contains the value '10'. The 'Executar' button is visible at the bottom right.

(c) Validação cruzada

Figura 61 – Opções de teste para modelos classificadores

### B.3.1 k-NN

Selecione a opção k-NN, escolha a opção de teste e informe o valor de K como mostrado na Figura 62. Clique no botão Executar para ver o resultado classificação.

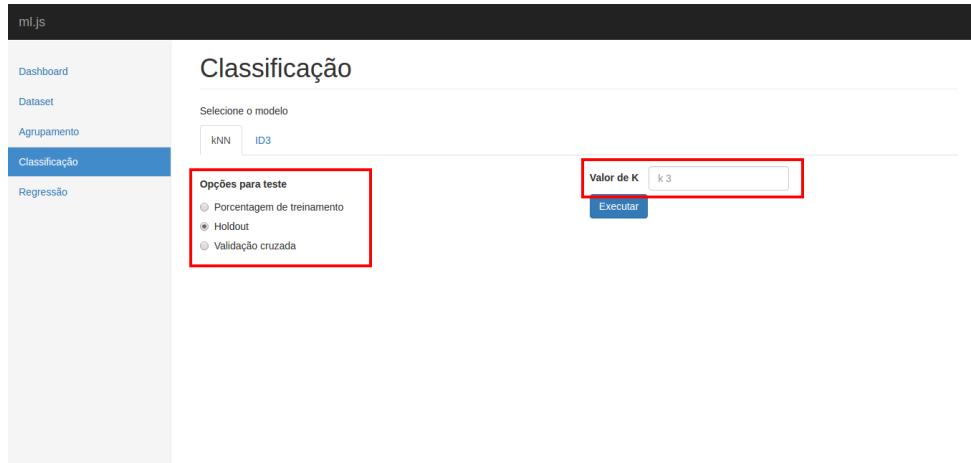


Figura 62 – Tela parâmetros k-NN

Exemplos de métricas obtidas pelo modelo são mostradas na Figura 63. São mostradas a Acurácia, Instâncias corretamente classificadas, Instâncias incorretamente classificadas, Total instâncias de teste, Precisão, Sensibilidade e Medida-F detalhados por classe e a matriz de confusão.

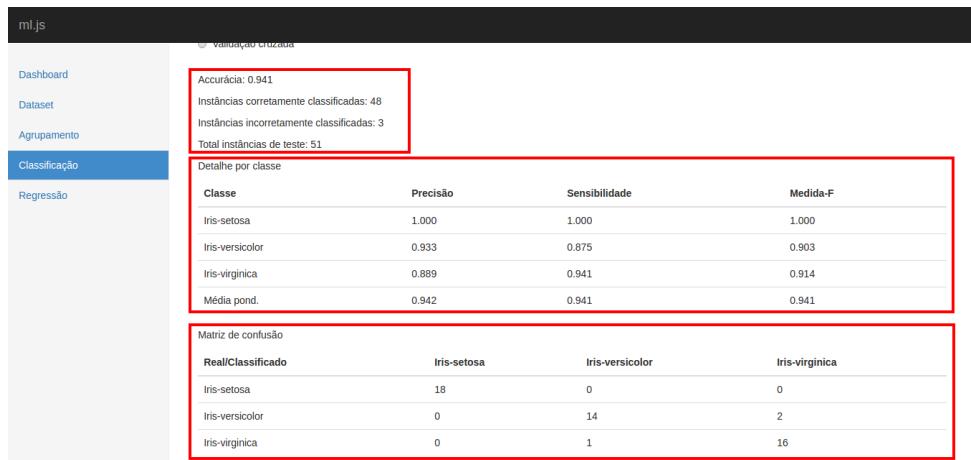


Figura 63 – Exemplo de métricas para modelo k-NN

Além das métricas citadas anteriormente, um gráfico com os ponto de dados que foram utilizados para o teste do modelo é mostrado, como ilustrado a Figura 64.

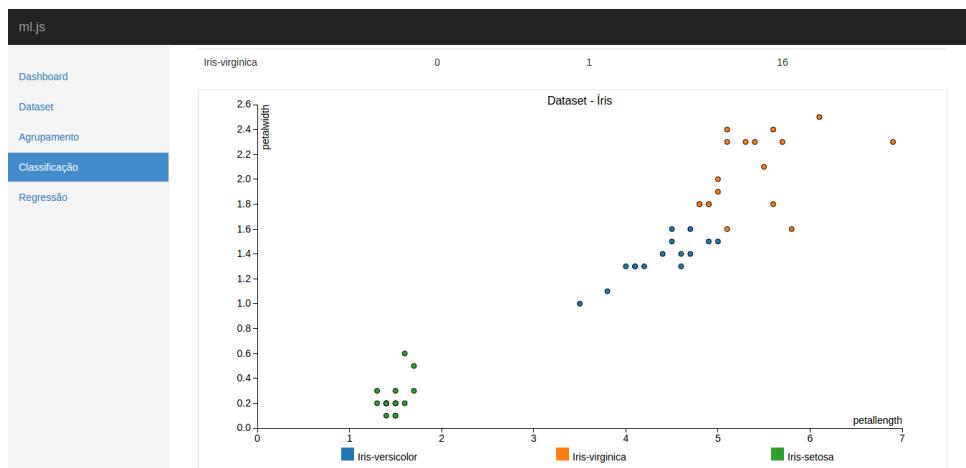


Figura 64 – Gráfico com ponto usados na avaliação do modelo k-NN

### B.3.2 ID3

Selecione a opção ID3, escolha a opção de teste e clique no botão Executar como mostrado na Figura 65.

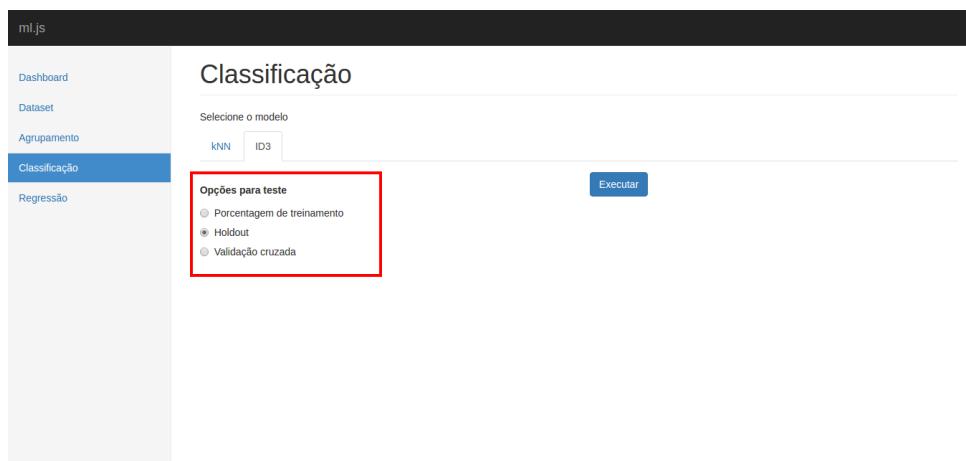


Figura 65 – Tela parâmetros ID3

Resultado é mostrado na Figura 66 que ilustra uma árvore de decisão gerada pelo modelo a partir do conjunto de dados selecionado.

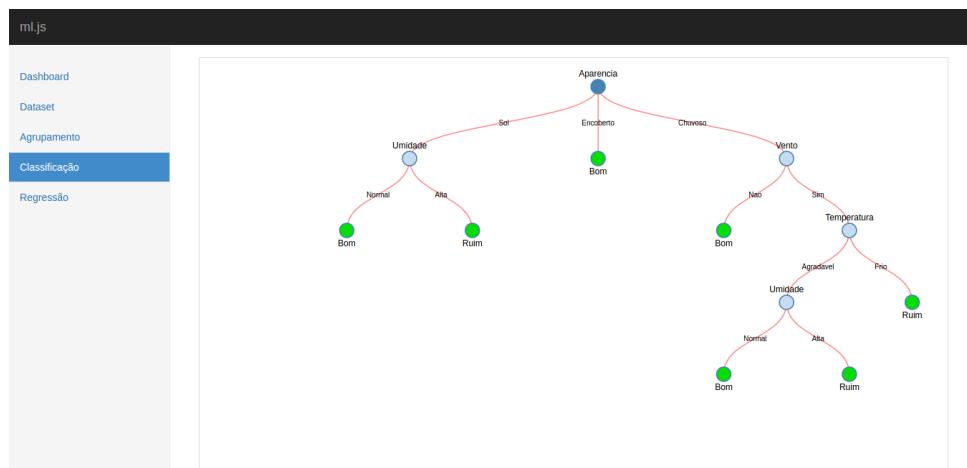


Figura 66 – Árvore de decisão gerada como resultado do ID3

A Figura 67 mostra um exemplo de métricas para o modelo ID3.

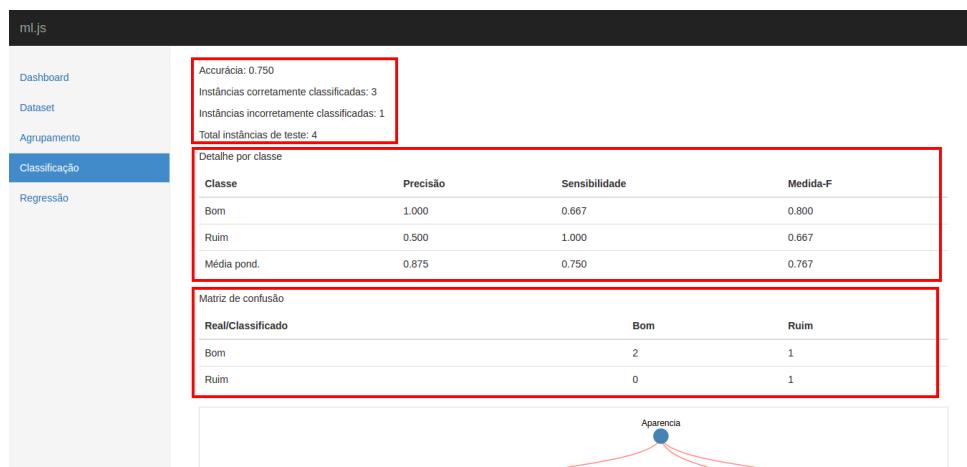


Figura 67 – Matriz de confusão e métricas para modelo de classificação ID3

## B.4 Regressão

A Figura 68 mostra a tela inicial para a atividade de regressão, seguindo a mesma ideia da Subseção B.2 selecione o modelo desejado.

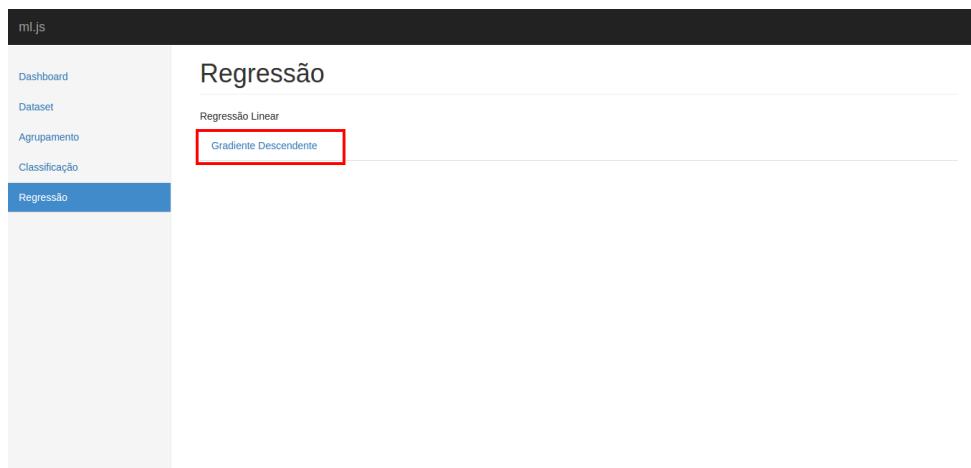


Figura 68 – Tela inicial para atividade de regressão

#### B.4.1 Gradiente descendente

Selecione a opção Gradiente descendente e preencha os valores dos parâmetros mostrados na Figura 69. Os parâmetros são Taxa de aprendizagem, Precisão e Máximo de iterações. Para treinar o modelo clique no botão Executar.

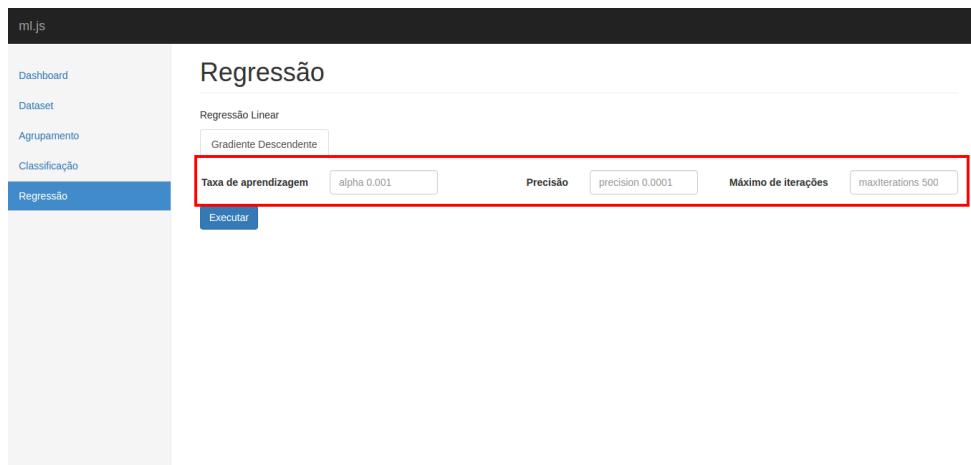


Figura 69 – Tela parâmetros Gradiente descendente

Um exemplo de métrica para o modelo Gradiente descendente é mostrado na Figura 70, onde pode ser visto a quantidade de iterações até atingir a convergência, o erro médio quadrático e o vetor de pesos.



Figura 70 – Exemplo métricas para modelo Gradiente descendente

A Figura 71 mostra a curva do erro médio quadrático para o modelo Gradiente descendente, nela pode ser visto o decaimento do erro ao longo das iterações até que o modelo atinja seu ponto de convergência de acordo com os valores dos parâmetros que foram informados.

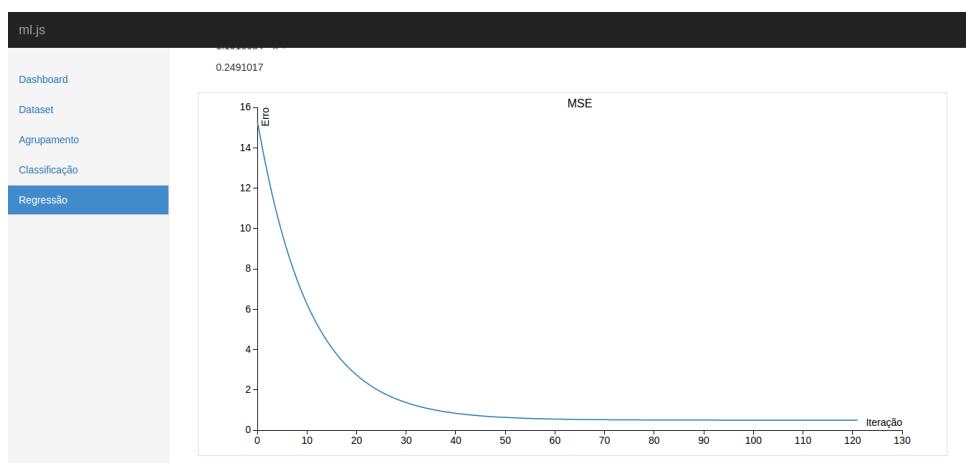


Figura 71 – Gráfico MSE × iteração para o modelo Gradiente descendente