**ESCOLA SUPERIOR DE PROPAGANDA E MARKETING**

**PROJETO DE GRADUAÇÃO EM TECNOLOGIA**

**PLANO DE PESQUISA ACADÊMICA**

BRUNO PAES

FERNANDO SINIGAGLIA

GUILHERME HEITZMANN

LEONARDO BRIOTTO

LEONARDO MESSIAS

**MOONCAKE:**

Melhorando o desempenho de preditores para detecção de transações fraudulentas

**São Paulo**

**2020**

BRUNO PAES

FERNANDO SINIGAGLIA

GUILHERME HEITZMANN

LEONARDO BRIOTTO

LEONARDO MESSIAS

**MOONCAKE:**

Melhorando o desempenho de preditores para detecção de transações fraudulentas

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Sistemas de Informação pela Escola Superior de Propaganda e Marketing –

ESPM.

Orientador: Prof. Dr. Antonio Marcos Selmini

Coorientador: Prof. Dr. Humberto Sandmann

**São Paulo**

**2020**

**Resumo**

Com o advento da tecnologia, as pessoas passaram a realizar transações online, o que por sua vez, surgiu e tem aumento o número de fraudes financeiras em transações online. Este trabalho tem como finalidade, entender a fundamentação teórica das RNAs, bem como, o seu processo de concepção e treinamento, com o propósito de analisar e reconhecer os padrões de transações *online* de caráter fraudulento.

**Palavras–chave:** Redes Neuras Artificiais, Inteligência, Classificação, Reconhecimento de padrões, Fraudes Financeiras.

**Lista de Figuras**

**Figura 1 –** Arquitetura da plataforma da ClearSale 10

**Figura 2 –** Modelo de um neurônio biológico 11

**Figura 3 –** Modelo de um neurônio artificial com uma camada oculta 12

**Figura 4 –** Gráfico da função sigmoide e sua respectiva derivada 13

**Figura 5 –** Gráfico da função tanh e sua respectiva derivada 14

**Figura 6 –** Gráfico da função ReLU e sua respectiva derivada 15

**Figura 7 –** Taxa de aprendizado pequena 16

**Figura 8 –** Taxa de aprendizado grande 17

**Figura 9 –** Arquitetura do Perceptron 18

**Figura 10 –** Arquitetura do MLP 19

**Figura 11 –** *Headers* do *dataset* 21

**Figura 12 –** *Boxplot* da variável *amount* 21

**Figura 13 –** Arquitetura do *TensorFlow* 23

**Figura 14 –** Arquitetura do *PyTorch* 24

**Figura 15 –** *Interest over time in tensorflow* 25

**Sumário**

**1. Introdução 6**

2. Revisão Literária 8

2.1. Definição de Fraude 8

2.1.1. Técnicas Antifraude 8

2.2. Aprendizado de Máquina 8

2.2.1. Tree Classifiers 8

2.2.2. Random Forests Classifiers 8

2.2.3. Percepton 8

2.2.4. MultiLayer Percepton 8

2.2.5. Support Vector Machine 8

2.3. Sampling 8

2.3.1. Random Sampling 8

2.3.2. SMOTe 8

3. Metodologia 20

3.1. Base de Dados 20

3.1.1. Breve descritivo & aquisição de dados 20

3.1.2. Análise Exploratória 20

3.2. Redução de Dimensionalidade 20

3.3. Resultados Preliminares 20

3.3.1. Overfitting e Falsos Negativos 20

3.4. Resultados Otimizados 20

3.4.1. Otimização dos modelos 20

4. Conclusão 20

4.1. Trabalhos Futuros 20

**1 Introdução**

**2 Revisão Literária**

Este capítulo tem como objetivo revisar a fundamentação teórica que serviu de embasamento para o desenvolvimento deste trabalho. Este capítulo está dividido em três seções. A primeira seção, baseando-se na Constituição Federal de 1988, busca definir fraudes, bem como, a importância de técnicas antifraudes – que utilizam aprendizado de máquina ou não – e estas estão sendo usadas para minimizar ou até eliminar os impactos das transações fraudulentas. A segunda seção explica a fundamentação base dos algoritmos de classificação mais comuns utilizados por equipes de *Data Science* espalhadas pelos mais diversos segmentos do mercado. A terceira seção, por sua vez, explica algumas técnicas utilizadas para minimizar os efeitos do *dataset imbalance* – como o *Overfitting* e o *Underfitting*.

**2.1 Definição de Fraude**

Fraudes financeiras não são questões relacionadas somente à bancos e instituições financeiras. Produtos ou serviços negociados através de sistemas digitais – mesmo aqueles em que não há a troca de explícita de dinheiro – são alvos de fraudes. As fraudes financeiras causam enormes problemas – não só geram prejuízos para as empresas ou clientes, mas também podem impactar a imagem das empresas negativamente (LIMA, Isaque. 2017).

A lei federal de Nº 8.137 de 27 de dezembro de 1990 é definida como qualquer ato ardiloso, enganoso, de má fé, com o intuito de lesar ou ludibriar outrem. Ou seja, no âmbito de transações financeiras, fraude caracteriza-se pelo ato intencional de manipulação de transações (Lei federal 8.137. 1990).

**2.1.1 Técnicas Antifraude**

Devido ao aumento de transações fraudulentas, novas técnicas computacionais passaram a ser usadas na detecção de transações fraudulentas como, por exemplo, o aprendizado de máquina. Os sistemas baseados em aprendizado de máquina são mais capazes de cruzar um grande volume de dados e classificar transações que fogem à normalidade do que um departamento de *compliance* operado por seres humanos.

Os sistemas antifraude são executados após uma transação e tentam comprovar a identidade da pessoa que efetuou a transação. Soluções comuns utilizam regras de negócio e regras estatísticas definidas por seres humanos – como, por exemplo, a definição de valores *outliers* – para classificar se uma transação é fraudulenta ou não. As soluções que utilizam de aprendizado de máquina, após cada transação, analisam os dados fornecidos pelo cliente durante a transação – localização, valor, estabelecimento/*website* – para identificar padrões e detectar os perfis que fogem à regra e, caso a fraude seja confirmada ou não o sistema retroalimenta a base de treinamento com a nova transação e se reajusta automaticamente – fazendo com que novas técnicas de fraudes sejam detectadas mais rapidamente (LIMA, Isaque. 2017).

Um dos grandes diferenciais de sistemas que utilizam o aprendizado de máquina quando comparados com sistemas antifraude comuns – que utilizam de estratificações e análises estátisticas – é a sua maior assertividade. A maior assertividade se dá pelo fato de quando a máquina passa a aprender com suas próprias avaliações a granularidade é muito maior do que quando o departamento de *compliance* cria um grupo de regras de negócios genéricas baseada na média dos perfis. Outra grande vantagem no uso de aprendizado de máquina na classificação de operações fraudulentas é o uso do *big data* – com cada a transação gerando um volume de dados gigantesco o sistema terá uma base de dados robusta e que permitirá um processo de treinamento mais refinado e por consequência uma maior taxa de assertividade (BUGHIN, Jacques; CHUI, Michael; HENKE, Nicolaus. 2016).

**2.2 Aprendizado de Máquina**

Esta seção, dividida em cinco partes, busca fundamentar a base teórica de cinco algoritmos de aprendizado de máquina utilizados amplamente em problemas de classificação e regressão. Segundo Andrew NG (NG, Andrew. 2018. p6) – Pesquisador chefe do Google Brain até 2012 – aprendizado de máquina é a ciência que faz com que computadores executem determinadas tarefas sem que sejam, para isso, explicitamente programados.

In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it (NG, Andrew. 2018. p6).

**2.2.1 Tree Classifiers**

*Tree Classifiers* ou *Decision Trees* são um dos métodos de Aprendizado de Máquina Supervisionado1 (J.R, Quinlan. 1985. p1). Estas árvores, de modo geral, são estruturas de dados formadas por um conjunto de elementos que armazenam informações em nós. As árvores sempre iniciam-se na raiz – um nó que está no topo da hierarquia – e dividem-se, por meio de ligações com nós filhos (filhos que podem possuir filhos que por sua vez podem possuir os seus). O nó que não possui filhos é conhecido como nó folha ou terminal (CAMPOS, Raphael. 2017). Em uma árvore de decisão, uma decisão é feia através do caminho percorrido a partir do nó raiz até o nó folha. A figura 1 ilustra a árvore de decisão com gerada com o *iris dataset2*.

1 Aprendizado de Máquina Supervisionado é um método de aprendizagem computacional que consiste em que o treinamento seja feito com a tutoria do *erro*, ou seja, o conjunto de treinamento é rotulado – sabe-se que um grupamento de atributos corresponde a uma determinada classe – e pode-se, por isso, verificar e ajustar o aprendizado do sistema.



Setosa Virginica Versicolor

**Figura 1** – Árvore de decisão do *Iris dataset.*

**Fonte:** Autor.

A Figura 1 apresenta um grafo que representa a árvore de decisão gerada pela base de dados das plantas *Iris* – Esse grafo foi gerado na linguagem de programação R **(Apêndice A)**:

1. Caso o comprimento da pétala seja menor que 1,9 cm encaminha-se para o **nó folha à esquerda** e a planta é classificada como *Iris Setosa*.
2. Caso contrário encaminha-se para o **nó filho à direita** em que verifica-se a largura da pétala.
   1. Caso a largura da pétala seja maior que 1,5 cm encaminha-se para o **nó folha à esquerda** e a planta é classificada como *Iris Virginica.*
   2. Caso contrário encaminha-se para o **nó folha à direita** e a planta é classificada como *Iris Versicolor.*

**2.2.2 Random Forests Classifiers**

Como o prórpio nome dá a entender *Random Forests* são compostos por inúmeras árvores de decisão operando como um *ensemble3.* A Figura 2 ilustra o funcionamento do *Random Forest*.

2 O *Iris dataset* é uma base de dados que foi coletada em 1936 e digitalizada em 1988 pela universidade da Califórnia. A base é composta por 150 elementos – cada um com 4 atributos – que representam medidas de três espécies da família de plantas *Iris* (*Iris Setosa, Iris Versicolor e Iris Vírginica*).

3 Ensemble Learning é um método que utiliza multiplos algoritmos de aprendizado num único modelo preditivo. Cada algoritmo funciona de forma independente e, após o treinamento, uma espécie de comitê escolhe os melhores preditores. Seu uso foca em alcançar melhores performances preditivas – o que não poderia ser alcançado por nenhum dos modelos constituintes caso classificassem sozinhos (Opitz, D; Maclin, R. 1999).



**(N)**

**Figura 2** – *Random Forests*

**Fonte:** Yiu, Anthony. **Understanding Random Forests**.

Como pode-se notar, o modelo apresentado na Figura 2 é composto por quatro árvores de decisão que treinam de forma independente uma da outra. Cada árvore recebe um conjunto aleatório de atributos do *dataset* de treinamento e, partir de então, o treinamento de cada árvore se dá de forma independente. Após o treinamento, uma espécie de comitê verifica a classe que, durante o treinamento, mais obteve “votos” e a escolhe como a classe finalista para determinados *inputs* (Yiu, Anthony. 2019)*.*

A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models... The reason for this wonderful effect is that the trees protect each other from their individual errors – as long all of them constantly err in the same direction (Yiu, Anthony. 2019).

**2.2.3 Perceptron**

O *Perceptron* é a forma mais simples de Redes Neurais Artificiais usada para classificação de padrões linearmente separáveis (padrões que encontram-se em lados opostos de um hiperplano) (HAYKIN, Simon 2008. p143). O modelo foi proposto por Frank Rosenblatt no ano de 1958 e introduz o conceito de aprendizagem supervisionada.

Sua arquitetura, como pode ser observada na Figura 3, não possui camadas intermediárias (camadas ocultas) e consiste apenas de uma camada de entrada (***x****1,* ***x****2*...**x**n), conectada, por meio das conexões sinápticas (********,********...*******n*), a uma camada de neurônios (***n****1*, ***n****2* ... ***n****n*) que redireciona a uma camada de saída (***s****1*,***s****2* ...***s****n*).

­­­

**Figura 3 –** Arquitetura do Perceptron

**Fonte:** PUC – Rio: Maxwell Biblioteca Digital. p41.

Durante o processo de treinamento do *Perceptron*, busca-se encontrar um conjunto de pesos sinápticos que determine uma reta que separe as diferentes classes, de maneira que a rede possa classificar corretamente as entradas apresentadas (PUC – Rio: Maxwell Biblioteca Digital. p42).

**2.2.4 MultiLayer Perceptron**

O *Perceptron* de Múltiplas Camadas (MLP) é considerado como um dos mais importantes modelos de RNAs. Esse tipo de modelo tem sido aplicado em diversos problemas de classificação e regressão, como por exemplo, no reconhecimento de voz e imagens (PUC – Rio: Maxwell Biblioteca Digital. p43).

Sua arquitetura, derivada do modelo original do Perceptron, além de ser constituída por uma camada de entrada (***x****1,* ***x****2*...**x**n) e uma camada de saída (***s****1*,***s****2* ...***s****n*), também consiste na adição de uma ou mais camadas ocultas (HAYKIN, Simon. 2008. p186). A Figura 4 ilustra a arquitetura de um MLP.

­­­

**Figura 4 –** Arquitetura do MLP

**Fonte:** PUC – Rio: Maxwell Biblioteca Digital. p43.

**2.2.5 Support Vector Machine**

*Support Vector*4 *Machine* (SVM) por se tratar de um dos algoritmos de *machine learning* que melhor gerencia o *trade off* entre custo computacional e assertividade é um dos preferidos em equipes de *data science* (GANDHI, Rohith. 2018). *Support Vector Machines* podem ser utilizadas tanto em problemas de classificação quanto de regressão, entretanto, é em problemas de classificação que ela possui mais destaque (GANDHI, Rohith. 2018).

A Máquina de Vetor de Suporte é uma *máquina linear* com algumas propriedades interessantes...A principal ideia de uma Máquina de Vetor de Suporte é construir um hiperplano como superfície de decisão de tal forma que a margem de separação entre exemplos positivos e negativos seja máxima (HAYKIN, Simon. 2008. p349).

As SVM constróem um hiperplano5 num espaço N-dimensional – sendo N o número de atributos – que permite a separação de duas classes de *datapoints* (HAYKIN, Simon. 2008. p351)*.* A Figura 5 ilustra alguns dos possíveis hiperplanos que uma SVM construiria para separar duas classes de *datapoints*.



**Figura 5 –** Possíveis Hiperplanos

**Fonte:** Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

Como pode-se notar na Figura 5, para separar duas classes de *datapoints*, existem diversos possíveis hiperplanos que poderiam ser elegidos como hiperplano ótimo. Entretanto, mesmo com inúmeros possíveis hiperplanos, o hiperplano ótimo é aquele que consegue maximizar a distância marginal entre os *datapoints* das duas classes (HAYKIN, Symon. 2008. p353). A Figura 6 ilustra a configuração de um hiperplano ótimo.

4 Support Vectors são os *datapoints* que estão mais próximos ao hiperplano e, por consequência, influenciam a posição e orientação do hiperplano (HAYKIN, Simon. 2008. p355). Ao se utilizar destes vetores de suporte a margem do classificador é maximizada que, por consequência, faz com que futuros *datapoints* sejam classificados com mais confidência (GANDHI, Rohith. 2018). (Anexo A)

5 Hiperplanos são fronteiras de decisão que auxiliam na classificação de *datapoints*. A dimensão de um hiperplano está relacionada diretamente com o número de atributos do *dataset* (HAYKIN, Simon. 2008. p114). Se o número de atributos do *dataset* for 2, então o hiperplano é somente uma linha. Se o número for 3, então o hiperplano passa a ser um plano bidimensional (GHANDI, Rohith. 2018). (Anexo B)



**Figura 6 –** Hiperplano Ótimo

**Fonte:** Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

**2.3 Sampling**

Esta seção, dividida em duas subseções, busca fundamentar três técnicas de *sampling* que buscam resolver o *class imbalance*, ou seja, buscam balancear – igualar a quantidade de elementos de classes minoritárias com a classe majoritária – *datasets* para que possam ser utilizados no treinamento dos mais diversos modelos preditivos. *Imbalanced datasets* são caracterizados por uma severa inclinação na distribuição de suas classes (e.g. *datasets* com distribuição de 1:1000). Tal desproporcionalidade pode trazer, ao modelo preditor, viés – dar peso desproporcional a uma classe – e fazer com que o mesmo ignore por completo a classe minoritária – tornando-se um problema quando as predições corretas da classe minoritária são as mais relevantes para o modelo (BRANCO, Paula; TORGO, Luís; RIBEIRO, Rita. 2015).

**2.3.1 Random Sampling**

Em cenários de *class imbalance*, uma das abordagens mais comuns é amostrar aleatoriamente o *dataset* de treinamento (BRANCO, Paula; TORGO, Luís; RIBEIRO, Rita. 2015). Dentro do *Random* Sampling, existem duas abordagens principais para se balancear *datasets* que possuam *class imbalance*, dentre elas:

* O *Random Undersampling* (RUS) consiste em deletar aleatoriamente exemplos da classe majoritária, ou seja, o RUS seleciona aleatoriamente um exemplo da classe majoritáia e o deleta do *dataset* de treinamemento.
* O *Random Oversampling* (ROS) consiste em duplicar aleatoriamente exemplos da classe minoritária, ou seja, o ROS seleciona aleatoriamente um exemplo da classe minoritária e o duplica no *dataset* de treinamento.

Ambas, por não usarem heurísticas e não assumirem nada sobre os dados – padrões, *outliers* e outros – são referenciadas como *Naive Resampling methods*. A execução destes métodos, por conta disso, é rápida – algo muito desejado em *datasets* de proporções grandes e mais complexas (BRANCO, Paula; TORGO, Luís; RIBEIRO, Rita. 2015). Estes métodos podem ser utilizados em *datasets* de classificação binária bem como em classificações multi-classes – com uma ou mais classes minoritárias ou majoritárias. Vale a pena ressaltar que estas mudanças são aplicadas somente aos *datasets* de treinamento – visto que a intenção é influenciar positivamente no treinamento dos modelos (BRANCO, Paula; TORGO, Luís; RIBEIRO, Rita. 2015).

Applying re-sampling strategies to obtain a more balanced data distribution is an effective solution to the imbalance problem…although that depends on the specifics of the dataset and models involved (BRANCO, Paula; TORGO, Luís; RIBEIRO, Rita. 2015).

**2.3.2 SMOTe**

*Synthetic Minority OverSampling Technique* (SMOTe) é uma técnica de *sampling* que baseia-se no algoritmo *k-nn* *(k-nearest neighbors)* –que calcula a distância euclidiana dos diversos *datapoints* de uma classe (CHAWLA, Nitesh; BOWYER, Kevin; HALL, Lawrence; KEGELMEYER, W. Philip. 2002).

Como pode-se notar na Figura 7, o algoritmo SMOTe, para cada exemplo da classe minoritária, encontra o k-vizinho mais próximo e, a partir de então, traça uma reta entre os vizinhos – distância euclidiana – e gera novos registros (Battacharyya, Indresh. 2018). Nota-se também que os exemplos sintéticos são geradas nas retas entre os pontos da classe minoritária.



**Figura 7 –** Funcionamento do SMOTe

**Fonte:** Battacharyya, Indresh. **SMOTe and ADASYN (Handling Imbalanced datasets).**

**3 Metodologia**

Este capítulo tem por objetivo descrever os recursos metodológicos utilizados para o desenvolvimento deste plano de pesquisa acadêmica. O mesmo é subdividido em duas seções. Sendo a primeira uma descrição detalhada da base de dados utilizada e a segunda um detalhamento de duas bibliotecas de Redes Neurais para a linguagem de programação *python*.

**3.1 Base de dados**

Este subcapítulo, subdividido em duas seções, descreve a base de dados, seu processo de aquisição e ilustra, por meio de gráficos e figuras, a análise exploratória das variáveis do *dataset*.

**3.1.1 Breve descritivo e aquisição de dados**

Adquirida na página do *kaggle* da *Nowergian University of Science and Technology,* a base de dados, nomeadade *Synthetic Financial Datasets For Fraud Detection* é uma base de dados sintética criada com o objetivo de suprir a falta de *datasets* públicos de cunho financeiro disponíveis pela *web*.

There is a lack of public available datasets on financial services and specially in the emerging mobile money transactions domain. Financial datasets are important to many researchers and in particular to us performing research in the domain of fraud detection. Part of the problem is the intrinsically private nature of financial transactions, that leads to no publicly available datasets (E. A. Lopez-Rojas; A. Elmir, and S. Axelsson. 2016).

Este *dataset* simula transações financeiras feitas por meio de aparelhos *mobile*. A base de dados, extraída de uma empresa de serviços financeiros é apenas uma amostra – diminuída em cerca de 4 vezes o seu tamanho original – dos registros cedidos pela empresa. Os registros deste *dataset* apresenta cerca de 31 dias de transações financeiras.

**3.1.2 Análise Exploratória**

O *dataset* é composto por 11 variáveis e possui 1.048.586 registros. O mesmo, após o *download*, é apresentado como um arquivo *csv* – *comma separated values –* e a primeira linha do *dataset* é apresentada na Figura 11.



**Figura 11 –** *Headers* do *dataset*

**Fonte:** E. A. Lopez-Rojas, A. Elmir, and S. Axelsson. 2016.

Como já mencionado anteriormente, o *dataset* possui 11 variáveis – numéricas ou não – e cada variável é explicada no Quadro 1.

|  |  |  |
| --- | --- | --- |
| ***Headers*** | **Explicação** | **Exemplo** |
| ***step:*** | Unidade de tempo. 1 *step* equivale à 1 hora. | 1 |
| ***type:*** | Tipo de transação. | PAYMENT |
| ***amount:*** | Unidade monetária. Valor total da transação. | 1060.31 |
| ***nameOrig:*** | ID da conta que iniciou a transação. | C429214117 |
| ***oldBalanceOrig:*** | Balanço inicial antes a transação - conta de origem. | 1089.0 |
| ***newBalanceOrig:*** | Balanço final após a transação - conta de origem. | 28.69 |
| ***nameDest:*** | ID da conta que recebeu a transação. | M1591654462 |
| ***oldBalanceDest:*** | Balanço inicial antes a transação - conta de destino. | 0.0 |
| ***newBalanceDest:*** | Balanço final após a transação - conta de destino. | 0.0 |
| ***isFraud:*** | A transação é uma fraude? | 0 |
| ***isFlaggedFraud:*** | A transação foi *flagada* como fraude pelo sistema da universidade? | 0 |

**Quadro 1 –** Explicação das variáveis do *dataset.*

**Fonte:** Autor

Ao se analisar o comportamento da variável *amount* (valor total da transação – como pode ser observado no quadro 1), nota-se que esta variável se comporta de forma não normalizada. O valor mínimo de uma transação foi de $ 0,00, ao passo que o valor máximo de uma transação foi de $ 92.445.516,64 e o seu *boxplot* é representado na figura 12.



**Figura 12 –** *Boxplot* da variável *amount*

**Fonte:** Autor

Ao se observar a figura 12, nota-se que a variável *amount*, em sua grande parte, é composta por valores *outliers*. O desvio padrão desta variável é de $ 603.858,23 – este valor indica o quão distante os *datapoints* encontram-se da média. A média, por sua vez, é de $ 179.861,90. A variável *isFraud*, por sua vez, é composta por dois valores únicos – 0 (não fraude) e 1 (fraude). Do total do *dataset*, apenas 0,12% das transações são fraudes em contrapartida, o sistema de classificação da universidade de ciência e tecnologia da Noruega considerou como transações fraudulentas apenas 0,00025% do *dataset*. Destas o modelo construído obteve uma taxa de assertividade de 99,89% – 1.047.433 valores classificados corretamente.

**Referências Bibliográficas**

BISHOP, Christopher Michael. **Neural Networks for Pattern Recognition**. 1a ed. Editora: Oxford University Press, 1996.

BRANCO, Paula; TORGO, Luís; RIBEIRO, Rita. **A Survey of Predictive Modelling under Imbalanced Distributions**.2015. Disponível em: <https://arxiv.org/pdf/1505.01658.pdf>. Acesso em: 17 de abril de 2020.

BRASIL. **Lei federal de Nº 8.137 de 27 de dezembro de 1990**.

Disponível em: <http://www.planalto.gov.br/ccivil\_03/leis/L8137.htm>. Acesso em 28 de abril de 2020.

BUGHIN, Jacques; CHUI, Michael; HENKE, Nicolaus. **The age of analytics: Competing in a data-driven world – McKinsey Global Institute**. 2016.

Disponível em <https://www.mckinsey.com/the-age-of-analytics-competing-in-a-data-driven-world>. Acesso em 15 de setembro de 2019.

BATTACHARYYA, Indresh. **SMOTE and ADASYN (Handling Imbalanced Datasets).** 2018. Disponível em: <https://medium.com/smote-and-adasyn-handling-imbalancedata>. Acesso em: 16 de abril de 2020.

CAMPOS, Raphael. **Árvores de Decisão**. 2017.

Disponível em: <medium.com/machine-learning-beyond-deep-learning/arvores-de-decisao>. Acesso em: 15 de abril de 2020.

CHAWLA, Nitesh; BOWYER, Kevin; HALL, Lawrence; KEGELMEYER, W. Philip. **SMOTE: Synthetic Minority Over-Sampling Technique.** 2002.

Disponível em: <https://arxiv.org/pdf/1106.1813.pdf>. Acesso em: 13 de março de 2020.

DOUGHERTY, Geoff. **Pattern Recognition and Classification: An Introduction**. Editora Springer. Edição: Softcover reprint of the original 1st ed. 2013.

E. A. Lopez-Rojas, A. Elmir, and S. Axelsson. **PaySim: A financial mobile money simulator for fraud detection**. 2016.

Disponível em: <https://www.kaggle.com/ntnu-testimon/paysim1>. Acesso em: 15 de outubro de 2019.

FISHER, Ronald; MARSHALL, Michael. **Iris Dataset**. 1936 – 1988. Disponível em: <https://archive.ics.uci.edu/ml/datasets/iris/>. Acesso em 15 de setembro de 2017.

GANDHI, Rohith. **Support Vector Machine – Introduction to ML Algorithms.** 2018. Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Acesso em: 16 de abril de 2020.

HAYKIN, Simon. **Redes Neurais – Princípios e práticas**. 3a ed. Porto Alegre, RS: Artmed, 2008.

J.R, Quinlan. **Induction of Decision Trees**. 1985.

Disponível em: <https://link.springer.com/content/pdf/10.1007/BF00116251.pdf>. Acesso em: 15 de abril de 2020.

LIMA, Isaque. **Inteligência Artificial chega aos sistemas antifraude com aprendizado de máquina**. 2017.

Disponível em: <https://canaltech.com.br/ia-chega-aos-sistemas-antifraude>. Acesso em 16 de setembro de 2019.

MAAS, Andrew; HANNUN, Awni. **Rectifier Nonlinearities Improve Neural Network Acoustic Models**. 2013.

Disponível em: <http://web.stanford.edu/~awni/papers/relu\_hybrid\_icml2013\_final.pdf/>. Acesso em 09 de janeiro de 2018.

McCULLOCH, Warren Sturgis; PITTS, Walter. **A logical calculus of the ideas immanent in nervous activity**. Bulletin of Mathematical Biophysics, 5, 115-137.

NG, Andrew. **Machine Learning Yearning**. 2018.

Disponível em: <https://github.com/ajaymache/machine-learning-yearning/>. Acesso em: 13 de abril de 2020.

OPITZ, D; MACLIN, R. **Popular Ensemble Methods: An Empirical Study.** Journal of Artificial Intelligence Research. Volume 11. 1999.

Disponível em: <https://jair.org/index.php/jair/article/view/10239>. Acesso em: 12 de abril de 2020

PUC – Rio: Maxwell Biblioteca Digital. **Redes Neurais.**

Disponível em: <https://www.maxwell.vrac.puc-rio.br/32823/32823\_4.PDF/>. Acesso em 29 de junho de 2018.

RIPLEY, Brian David. **Pattern Recognition and Neural Networks**. eBook Kindle. 1a ed. Editora: Cambridge University Press, 2008.

RUSSEL, Stuart Jonathan; NORVIG, Peter. **Inteligência Artificial**. 3a ed. Rio de Janeiro: Elsevier, 2013

YIU, Anthony. **Understanding Random Forest**.2019.

Disponível em: <https://towardsdatascience.com/understanding-random-forest>. Acesso em: 15 de abril de 2020

**Apêndice A – *Plotting* Tree graph**

[1] library(RWeka)

[2] data(iris)

[3] ind = sample(2, nrow(iris), replace=TRUE, prob=c(0.8, 0.2))

[4] train = iris[ind == 1, ]

[5] test = iris[ind == 2, ]

[6] plot(J48(Species ~ ., data=train))

**Anexo A – Support Vectors**



**Figura xx –** Support Vectors

**Fonte:** Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

**Anexo B – Hiperplanos**



**Figura xx –** Hiperplanos

**Fonte:** Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**