ESCOLA SUPERIOR DE PROPAGANDA E MARKETING

PROJETO DE GRADUAÇÃO EM TECNOLOGIA
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

Em decorrência do aumento de operações fraudulentas e, com o intuito de se protegerem e também de protegerem seus clientes contra fraudes, as empresas têm criado ou aprimorado suas áreas de análise de fraude. Dentre várias técnicas que podem ser utilizadas, a criação de modelos estatísticos que analisam os dados das transações e definem um *score* baseado nos metadados de cada transação têm sido bastante empregadas. Para cada transação realizada e, caso o *score* de uma transação ultrapasse um *treshold*, previamente definido, a transação é marcada para que um analista humano a revise e tome as decisões.

Em síntese, o objetivo deste trabalho é entender os *datasets* de fraudes financeiras e, com isso, criar modelos preditivos, utilizando técnicas de Aprendizado de Máquina, que apresentem um desempenho satisfatório – modelos com maior taxa de assertividade e que minimizem o número de falsos negativos (transações fraudulentas classificadas como não fraudulentas) – erro inaceitável neste problema de classificação.

Palavras–chave: Classificação, Reconhecimento de padrões, Fraudes Financeiras.

Abstract

As a result of the increase in fraudulent operations and, in order to protect themselves and also to protect their customers against fraud, companies have created or improved their areas of fraud analysis. Among several techniques that can be used, the creation of statistical models that analyze the data of the transactions and define a score based on the metadata of each transaction has been widely used. For each transaction carried out and, if the score of a transaction exceeds a previously defined threshold, the transaction is scheduled for a human analyst to review and make decisions.

In summary, the objective of this work is to understand the datasets of financial fraud and, with that, to create predictive models, using Machine Learning techniques, that present a satisfactory performance - models with a higher rate of assertiveness and that minimize the number of false negatives (fraudulent transactions classified as non-fraudulent) - unacceptable error in this classification problem.

Keywords: Classification, Pattern Recognition, Financial Fraud.

Lista de Figuras

Sumário

1 Introdução

Como parte importante de uma contínua evolução, a tecnologia e a internet têm proporcionado profundas mudanças no mundo – alterando o modo e a velocidade que tais negócios são realizados e firmados. Como consequência direta desta evolução novas tecnologias e campos de estudo surgem – Aprendizagem de máquina, por exemplo. Em contraponto os avanços tecnológicos favoreceram também que algumas situações desagradáveis surgissem e passassem à ocorrer com mais frequência – causando transtornos à indivíduos – como, por exemplo, a maior ocorrência de transações fraudulentas – não somente em instituições financeiras mas também em empresas e negócios online.

Com a evolução tecnológica, novas técnicas de inteligência artificial vêm sendo desenvolvidas para identificar se uma transação é fraudulenta, e, além disso, tornar os sistemas mais seguros, buscando evitar fraudes e quaisquer tipos de problemas que impactam os negócios e os clientes.

Atualmente tem-se aplicado Aprendizagem de Máquina para entender e analisar os dados de uma forma mais simples, e fornecer diversos serviços por meio delas. Mas para isso é necessário entender o que é Aprendizagem de Máquina e como é utilizada para identificar uma fraude. Aprendizagem de Máquina é uma área de inteligência artificial, onde seu principal objetivo é desenvolver técnicas computacionais sobre aprendizado bem como desenvolver sistemas que possam adquirir conhecimento (REZENDE, 2003). Sistemas que estão aprendendo são programas computacionais que tomam as decisões baseando-se em situações ou problemas anteriormente explorados (Weiss & Kulikowski, 1991). É uma técnica que vem sendo utilizada para identificar anomalias em documentos, boletos, e nas formas de pagamentos atuais com o objetivo de evitar fraudes em documentações dos clientes, evitar duplicidade de pagamento e garantir a segurança de uma transação.

1.1 Objetivos e Justificativas

Os avanços tecnológicos, trazidos por anos de pesquisa e desenvolvimento, trouxeram à sociedade mais dinamismo no modo como as tarefas cotidianas são realizadas. Tais avanços, embora muito benéficos, também trouxeram certas complicações às empresas e pessoas. Com a popularização de tecnologias como o *smartphone*, cartões de crédito e a própria internet – com *e-commerces* e o *internet banking* – operações fraudulentas, pela maior facilidade de acesso à vítimas potenciais – que agora podem ser alcançadas por meio de *e-mails*, mensagens e telefonemas – se popularizaram. Em decorrência do aumento de operações fraudulentas e,

com o intuito de se protegerem e também de protegerem seus clientes contra fraudes, as empresas têm criado ou aprimorado suas áreas de análise de fraude. Dentre várias técnicas que podem ser utilizadas, a criação de modelos estatísticos que analisam os dados das transações e definem um *score* baseado nos metadados de cada transação têm sido bastante empregadas. Para cada transação realizada e, caso o *score* de uma transação ultrapasse um *treshold*, previamente definido, a transação é marcada para que um analista humano a revise e tome as decisões.

Em síntese, o objetivo deste trabalho é entender os *datasets* de fraudes financeiras e, com isso, criar modelos preditivos, utilizando técnicas de Aprendizado de Máquina, que apresentem um desempenho satisfatório – modelos com maior taxa de assertividade e que minimizem o número de falsos negativos (transações fraudulentas classificadas como não fraudulentas) – erro inaceitável neste problema de classificação.

1.1 Organização do Trabalho

Esta seção tem por objetivo descrever a organização deste trabalho.

Capítulo 2 – Revisão Literária:

- Baseado na legislação brasileira define o que é uma fraude e introduz às tecnicas antifraudes usadas por diversas empresas em seus produtos e serviços;
- Explicar o funcionamento de todos os modelos preditivos – aprendizagem de máquina – utilizados neste trabalho;
- Explicar algumas técnicas de *Data Science* que buscam resolver o problema de *class imbalance* – *datasets* desbalanceados compostos por duas ou mais classes.

Capítulo 3 – Metodologia:

- Descrever o processo de aquisição do *dataset* bem como demonstrar o comportamento dos dados – por meio de *plots* e estatística descritiva;
- Descrever o processamento dos dados – Redução de dimensionalidade, normalização e limpeza dos dados;
- Demonstrar o desempenho dos modelos construídos bem como descrever o processo de otimização dos dados e, por consequência, dos modelos preditivos.

Capítulo 4 – Conclusão:

- Demonstração e comparação dos resultados obtidos;
- Conclusão e trabalhos futuros.

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 define fraude 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 Nº 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 operações fraudulentas 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 estatísticas – 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 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 Supervisionado[1] (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 se iniciam 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.

[1] Aprendizado de Máquina Supervisionado é um método de aprendizagem computacional que consiste 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.

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 dataset*[2].

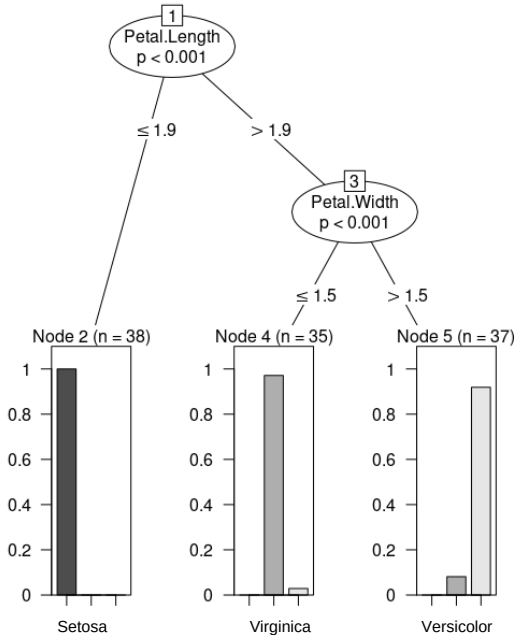

Figura 1 – Árvore de decisão do *Iris dataset.*
Fonte: Autor.

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 se verifica a largura da pétala.
    a. 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.*
    b. Caso contrário encaminha-se para o **nó folha à direita** e a planta é classificada como *Iris Versicolor*.

2.2.2 Random Forests Classifiers

Um outro algoritmo de classificação são os *Random* Forests que nada mais são do que diversas árvores de decisão operando como um conjunto único. Surgiram em decorrência do baixo desempenho apresentado em problemas de classificação mais complexos – problemas normalmente não linearmente separáveis ou *datasets* compostos por muitos atributos. *Random Forests* são compostos por inúmeras árvores de decisão operando como um *ensemble*[3].

[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).

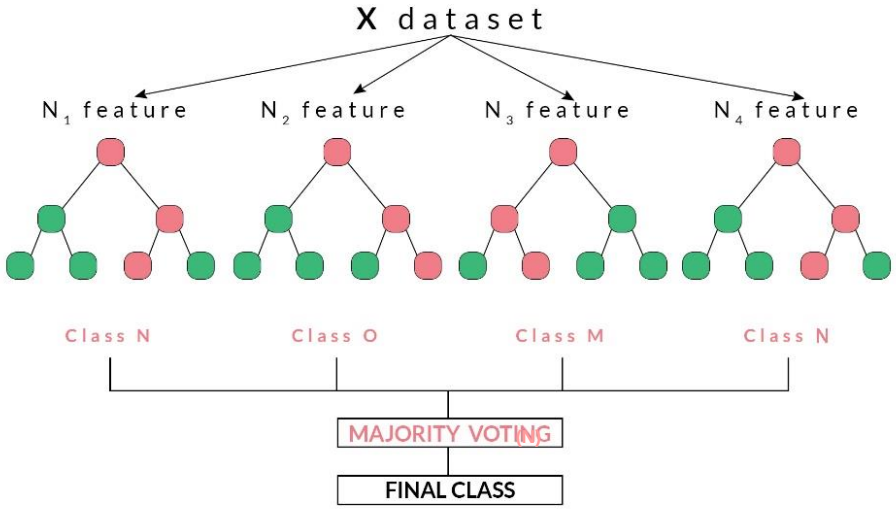

Figura 2 – *Random Forests.*
Fonte: Yiu, Anthony. **Understanding Random Forests**.

O funcionamento de um *Random Forest* é apresentado na figura 2. 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 se encontram 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 ($\boldsymbol{x}_1$, $\boldsymbol{x}_2$...$\boldsymbol{x}_n$), conectada, por meio das conexões sinápticas ($\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$... $\boldsymbol{\omega}_n$), a uma camada de neurônios ($\boldsymbol{n}_1$, $\boldsymbol{n}_2$ ... $\boldsymbol{n}_n$) que redireciona a uma camada de saída ($\boldsymbol{s}_1$, $\boldsymbol{s}_2$ ... $\boldsymbol{s}_n$).

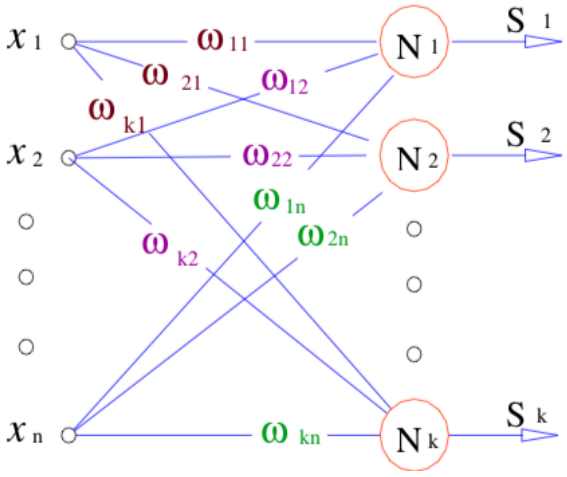

Figura 3 – Arquitetura do Perceptron.
Fonte: PUC – Rio: Maxwell Biblioteca Digital. p41.

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 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.

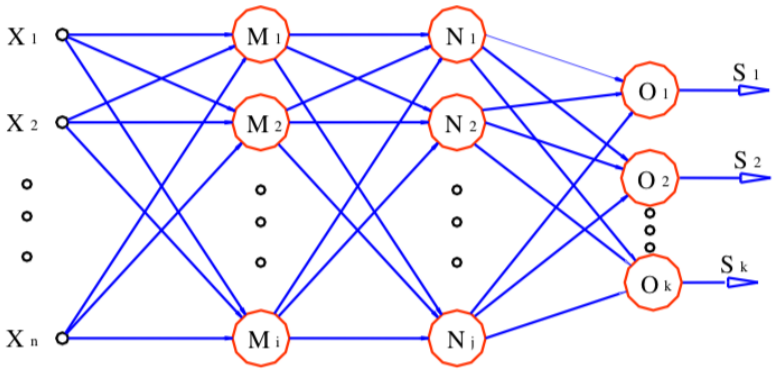

Figura 4 – Arquitetura do MLP.
Fonte: PUC – Rio: Maxwell Biblioteca Digital. p43.

2.2.5 *Support Vector Machines*

Support Vector[4] *Machines* (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, são 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 constroem um hiperplano[5] num espaço N-dimensional – sendo N o número de atributos – que permite a separação de duas ou mais 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*.

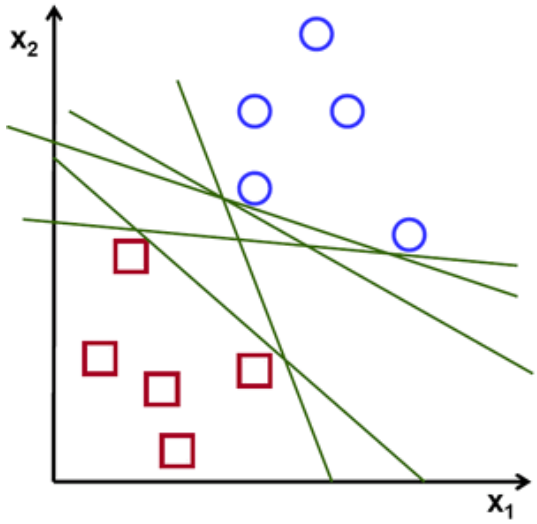

Figura 5 – Possíveis Hiperplanos.
Fonte: Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

Como pode-se notar na Figura 5, para separar duas classes de *datapoints*, existem diversos possíveis hiperplanos que poderiam ser eleitos 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)

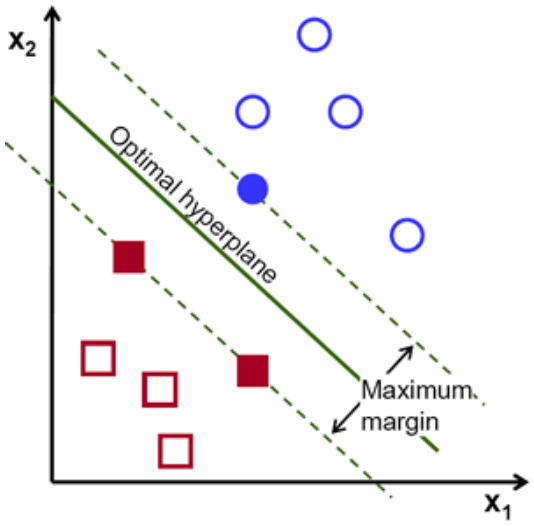

Figura 6 – Hiperplano Ótimo.
Fonte: Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

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 excluir 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 se baseia 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 gerados nas retas entre os pontos da classe minoritária.

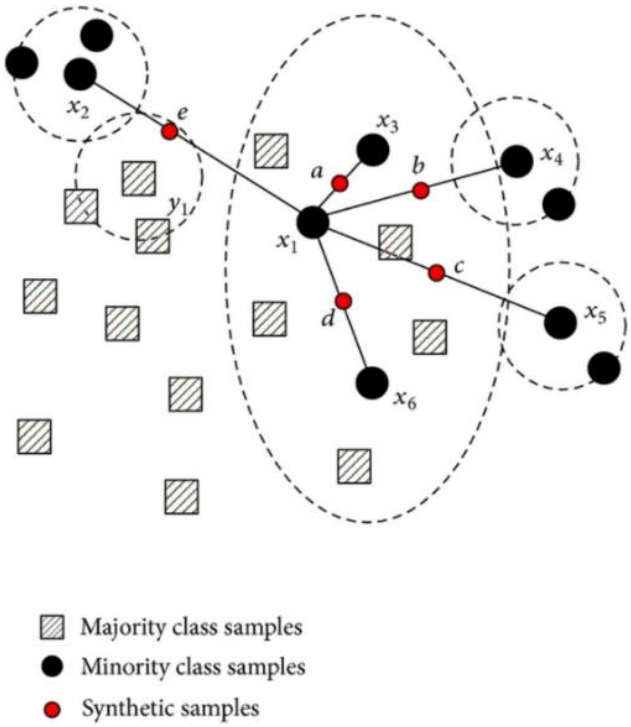

Figura 7 – Funcionamento do SMOTe.
Fonte: Battacharyya, Indresh. **SMOTe and ADASYN (Handling Imbalanced datasets).**

3 Metodologia

Este capítulo tem por objetivo descrever os recursos metodológicos utilizados para o desenvolvimento deste trabalho. O mesmo é subdividido em quatro seções. Sendo a primeira uma descrição detalhada da base de dados utilizada, a segunda uma descrição dos métodos utilizados no pré-processamento dos dados, a terceira e a quarta seção demonstram, respectivamente, os resultados obtidos pelos modelos treinados com as bases de dados, não otimizadas e otimizadas.

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 dos atributos 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, nomeada de *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 atributos e possui 6.354.407 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 8.

This is a sample of 1 row with headers explanation:

1,PAYMENT,1060.31,C429214117,1089.0,28.69,M1591654462,0.0,0.0,0,0

Figura 8 – *Headers* do *dataset.*
Fonte: E. A. Lopez-Rojas, A. Elmir, and S. Axelsson. 2016.

Como já mencionado anteriormente, o *dataset* possui 11 atributos – numéricos ou não – e cada atributo é explicado 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 dos atributos 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 9.

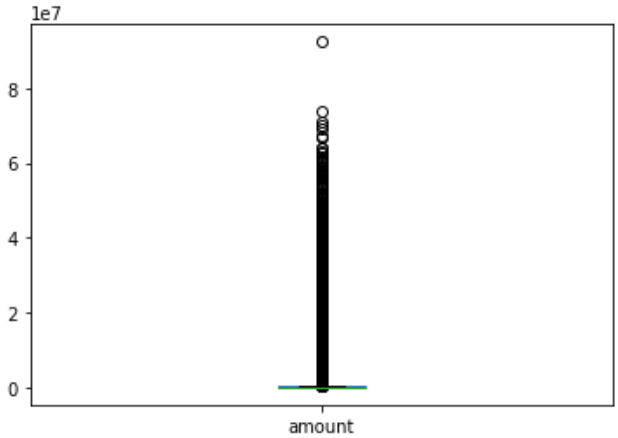

Figura 9 – *Boxplot* da variável *amount.*
Fonte: Autor.

Ao se observar a figura 9, 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. O valor médio é de cerca 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* – 6.354.407 *datapoints* – apenas 0,12% das transações são caracterizadas como fraudes, ou seja, dos pouco mais de 6 milhões de *datapoints*, apenas 8213 são caracterizadas como fraudulentas, caracterizando assim, o *class imbalance* – que pode vir a trazer problemas durante o treinamento dos modelos preditivos.

3.2 Pré-processamento

O pré-processamento dos dados é, em qualquer projeto de *Data Science*, um dos principais pilares para o desenvolvimento de bons preditores. Esta etapa consiste na preparação, organização e estruturação dos dados além de ser o momento ideal para se escolher quais atributos fazem sentido em permanecer como parte integrante do *dataset* utilizado pelos modelos preditivos. O pré-processamento dos dados, geralmente, inicia-se com a limpeza ou normalização de anomalias na base de dados, ou seja, busca-se, por exemplo, a remoção ou o preenchimento de valores faltantes (N.A. *values*), *outliers* – valores maiores que 3 desvio-padrões da média – e outras anomalias estatísticas. Para a verificação de valores N.A utilizou-se o Apêndice B.

A variável *"type"* – tipo de transação – era composta por *strings* que identificavam os diferentes tipos de transação que poderiam ocorrer. Como alguns dos modelos preditivos utilizados neste trabalho só conseguem trabalhar com dados de tipagem numérica, a conversão para valores numéricos fez-se necessária. O quadro 2 ilustra a alteração da variável *"type"* pelo Apêndice C.

Exemplo Original – *type (string)*	Conversão – *type (int)*
Cash_In	0
Cash_Out	1
Debit	2
Payment	3
Transfer	4

Quadro 2 – Processamento da variável *type.*
Fonte: Autor.

Como pode-se notar no quadro 2, todos exemplos da variável *type* que se correspondiam como *Cash_In* – Depósito – passaram a equivaler à 0. Como já mencionado, essa mudança foi

feita para que alguns modelos – incapazes de trabalhar com dados não numéricos – pudessem ser utilizados.

Com o intuito de se otimizar o desempenho dos preditores – menor custo computacional, mas mantendo-se com baixos falsos negativos – alguns atributos (estatisticamente menos relevantes) teriam de ser eliminados da base. Para isso um modelo de regressão linear múltipla foi criado - que treinou baseado nos 11 atributos - e por meio de correlação e outras técnicas estatísticas definiu quais variáveis independentes (x) mais impactavam na variável dependente (Fraude). O Apêndice D ilustra tal operação.

Outras técnicas mais robustas poderiam ser utilizadas - como, por exemplo, o *Principal Component Analysis* (PCA). Entretanto, optou-se, por conta de sua simplicidade e desempenho, utilizar a regressão linear para reduzir a dimensionalidade do *dataset* - que diminuiu de 10 atributos para somente 6 – *type, amount, oldBalanceOrig, newBalanceOrig, oldBalanceDest* e *newBalanceDest*.

Além da limpeza e da redução de dimensionalidade, a normalização – característica fundamental do pré-processamento, também é necessária. No momento das análises descritivas da base de dados, pôde-se observar que os valores mínimos e máximos – de diversos atributos – possuíam um alto grau de amplitude, ou seja, os valores mínimos e máximos de um atributo encontravam-se distantes uns dos outros. Para otimizar, os atributos de classificação foram normalizados utilizando o *MinMaxScaler* (Apêndice E). Esse algoritmo realiza o quociente de todos elementos de cada atributo pelo maior valor de cada atributo - resultando assim em um *dataset* de baixa amplitude com o valor mínimo sendo 0 e o maior sendo 1.

Um dos últimos passos do pré-processamento é a separação do *dataset*. Algumas bibliografias indicam a proporção de 80/20 – 80% do material destinado ao treinamento e os 20% restantes para testes de performance e assertividade dos modelos – outras (NG, Andrew. 2018), por sua vez, indicam 80/10/10 – 80% do material destinado ao treinamento, 10% destinada à testes e os 10% restantes à validação dos modelos – no entanto, independentemente da bibliografia, do autor e, principalmente, da época em que foi escrita (sendo que as mais modernas optam pela proporção de 80/10/10), a ideia de se fatiar o *dataset* tem o propósito de fornecer material suficiente para o treinamento do modelo e ainda ter dados suficientes para verificação de desempenho antes que os modelos sejam colocados em produção. Neste cenário, em específico, a proporção de 80/10/10 foi escolhida. A figura 10 ilustra a divisão do *dataset*.

sample	fraud_proportion	non_fraud_proportion	fraud_examples	non_fraud_examples	total
train	0.001291	0.998709	6570	5083525	5090095
test	0.001292	0.998708	822	635442	636264
validation	0.001290	0.998710	821	635440	636261

Figura 10 – Divisão do *dataset*.
Fonte: Autor.

Ao se observar a figura 10 nota-se que após a divisão do *dataset* original – treinamento, teste e validação – a proporcionalidade original de exemplos que correspondiam à fraudes e não fraudes manteve-se, ou seja, cada *dataset* possui os mesmos 0,12% de registros correspondentes à fraudes.

3.3 *Overfitting* e Falsos Negativos

Um dos principais problemas encontrados em cenários de classificação ou regressão é o *class imbalance*, que pode trazer viés aos modelos preditivos – os fazendo ignorar por completo a classe minoritária. Neste cenário, por conter apenas duas classes – Fraude e Não-Fraude, correspondendo, respectivamente à 1 e 0 – esperava-se, inicialmente, que o *dataset* contenha proporções próximas à 50% para cada classe, o que, como visto nas seções anteriores, não ocorreu e que pode acarretar em *overfitting*[6] dos modelos preditivos. O *overfitting*, por sua vez, pode induzir os modelos preditivos à dois tipos de erros:

- **Falsos Negativos:** classificar uma transação fraudulenta como não-fraude;
- **Falsos Positivos:** classificar uma transação não fraudulenta como fraude.

Levando em consideração o cenário de classificação, os Falsos Negativos são erros com menos tolerância – visto que na prática enviar uma transação à um analista humano é um cenário menos ruim do que não considerar fraude uma transação fraudulenta.

3.3.1 *Benchmarking* Preliminar

Esta seção tem, por objetivo, descrever o processo de treinamento e validação dos modelos preditivos utilizados neste trabalho. Os algoritmos – *Tree Classifiers*, *Random Forests* e SVM – usados neste trabalho foram escolhidos por sua eficiência e popularidade em diversos problemas de classifcação. Os *datasets*, anteriormente separados, foram utilizados para o treinamento, teste e validação dos modelos, entretanto, um quarto *dataset* (Apêndice F) – *fraud* – composto apenas por transações fraudulentas (aglomeradas dos *datasets* de teste e de validação), também foi utilizado para validar o desempenho dos modelos – sem que houvesse,

[6] *Overfitting* ocorre quando, no treinamento, o modelo apresenta um desempenho excelente, no entanto, quando os dados de teste ou validação são apresentados, o seu desempenho é ruim. Isso acontece pois o modelo "decorou" as relações existentes e, ao aplicá-las nos *datasets* de teste as regras decoradas não funcionam bem ocasionando um desemepho ruim (Anexo C).

assim, a distorção causada pelo massivo volume de registros não fraudulentos. A figura 11 demonstra o desempenho dos três modelos utilizados.

	train	test	validation	training time
tree classifier	0.999	0.999	0.999	0:00:05.459905
random forest	1.000	1.000	1.000	0:07:41.670989
svm	0.999	0.999	0.999	0:03:38.943760

Figura 11 – Resultados dos modelos preditivos – *dataset* de treinamento desbalanceado.
Fonte: Autor.

Como pode-se notar, os três modelos – *Tree Classifier* (Apêndice G.1), *Random Forest* (Apêndice G.2) e SVM (Apêndice G.3) – mesmo sem hiperparametrização obtiveram taxas de assertividades elevadas (beirando os 100%) nos *datasets* de treinamento, testes e validação. Entretanto, essas altas taxas de assertividade são resultados diretos do *class imbalance*, ou seja, a elevada quantidade de registros não fraudulentos, por sua desproporcionalidade, distorce a assertividade dos modelos. Por conta disso, o *dataset fraud* – composto apenas pelos registros fraudulentos dos *datasets* de teste e validação – foi utilizado para verificar a assertividade dos modelos, sem que assim, houvesse a distorção causada pelo *class imbalance*.

	fraud	training time
tree classifier	0.041	0:00:05.459905
random forest	0.595	0:07:41.670989
svm	0.365	0:03:38.943760

Figura 12 – Resultados dos modelos preditivos (desbalanceado) – *fraud dataset.*
Fonte: Autor.

A figura 12, como pode-se observar, demonstra as taxas de assertividade – dos mesmos modelos utilizados anteriormente – no *dataset* de fraudes. Nota-se que a assertividade dos modelos caiu de forma abrupta – indo de aproximadamente 100% para, no melhor dos casos, aproximadamente 60%. Esse vale entre os desempenhos apresentados pelos modelos nos *datasets* das figuras 11 e 12 são explicados pelo *class imbalance* e, embora, a comparação das tabelas indique o problema de *class imbalance*, a análise das matrizes de confusão ainda é válida para dimensionar o erro, bem como, seu tipo de erro (Falsos Negativos ou Falsos Positivos). As tabelas 1, 2 e 3 demonstram as matrizes de confusão dos modelos preditivos.

Tree Classifier											
	train			test			validation			fraud	
	Non-Fraud	Fraud		Non-Fraud	Fraud		Non-Fraud	Fraud		Non-Fraud	Fraud
Non-Fraud	5,083,511	0	Non-Fraud	635,437	0	Non-Fraud	635,459	0	Non-Fraud	0	0
Fraud	6,314	271	Fraud	794	31	Fraud	766	31	Fraud	1,575	68

Tabela 1 – Matrizes de Confusão – *tree classifier*
Fonte: Autor

Como pode-se observar na tabela 1, o modelo *tree classifier* obteve um péssimo desempenho em todos os *datasets*. Isso é evidenciado na matriz de confusão do *dataset* de treinamento – que obteve uma quantidade assustadora de Falsos Negativos (mesmo predizendo dados já vistos pelo modelo no treinamento).

Random Forests											
	train			test			validation			fraud	
	Non-Fraud	Fraud		Non-Fraud	Fraud		Non-Fraud	Fraud		Non-Fraud	Fraud
Non-Fraud	5,083,511	0	Non-Fraud	634,430	7	Non-Fraud	635,450	9	Non-Fraud	0	0
Fraud	0	6,585	Fraud	19	656	Fraud	154	649	Fraud	664	979

Tabela 2 – Matrizes de Confusão – *random forests.*
Fonte: Autor.

Como pode-se observar na tabela 2, o modelo *random forest*, diferentemente do *tree classifier*, não apresentou erros no *dataset* de treinamento. Nos demais *datasets*, por conta de uma baixa quantidade de Falsos Positivos, fica evidenciado um levíssimo grau de *overfitting* na classificação de não fraudes, no entanto, o maior problema encontra-se nos Falsos Negativos – onde o modelo apresentou, por conta do *class imbalance* – dificuldades para classificar corretamente. Vale ressaltar que a redução de dimensionalidade causou impacto direto no *random forest* – que como já mencionado possui uma péssima eficiência computacional – que com a redução de dimensionalidade, treinou em aproximadamente 11 minutos a menos (caiu de aproximadamente 18 minutos para aproximadamente 7 minutos) e obteve desempenho similar. O modelo SVM apresentou desempenho similar ao *random forest*, o que fica evidenciado na tabela 3.

SVM											
	train			test			validation			fraud	
	Non-Fraud	Fraud		Non-Fraud	Fraud		Non-Fraud	Fraud		Non-Fraud	Fraud
Non-Fraud	5,083,406	105	Non-Fraud	635,422	15	Non-Fraud	635,451	8	Non-Fraud	0	0
Fraud	537	2,375	Fraud	537	288	Fraud	499	304	Fraud	1,044	599

Tabela 3 – Matrizes de Confusão – SVM.
Fonte: Autor.

Como pode-se observar nas tabelas 1, 2 e 3, o grande volume de registros não fraudulentos distorce, não somente o treinamento – enviesando os modelos a classificarem com maior assertividade transações não fraudulentas – mas também em sua validação – mascarando as altas taxas de assertividade. Nas três tabelas pode-se notar que a quantidade de Falsos Positivos, quando comparada à quantidade de Falsos Negativos, é baixa, ou seja, os modelos

preditivos, não por conta de um possível *overfitting*, mas sim pelo *class imbalance*, são excelentes preditores de transações não fraudulentas. De modo geral, os três modelos, por conta do *class imbalance*, são excelentes preditores de transações não fraudulentas – o que fica evidenciado nesta seção – entretanto, para predizer transações fraudulentas, os modelos, pelos seus elevados números de Falsos Negativos, deixam à desejar.

3.4 Balanceando o *dataset* de treinamento

O objetivo desta seção é descrever o processo de otimização do *dataset* de treinamento e comparar o desempenho dos modelos preditivos que treinaram com o *dataset* otimizado. Como já descrito na seção anterior, os modelos preditivos, devido ao *class imbalance*, obtiveram um péssimo desempenho na classificação das transações fraudulentas.

Ao longo dos anos, diversas técnicas foram criadas e aperfeiçoadas para se corrigir o *class imbalance*. Como já descrito na seção 2.3 as técnicas mais simples são focadas em balancear o *dataset* de treinamento apenas pela duplicação de registros já existentes da classe minoritária (ROS) – que pode, por conta da repetição de informações, acarretar em *overfitting* – ou pela remoção de registros da classe minoritária (RUS). Outras técnicas, como por exemplo o SMOTe, baseiam-se na distância euclidiana entre os vizinhos para gerar novos registros e assim balancear o *dataset.* Neste trabalho o ROS e o SMOTe foram testados como técnicas de balanceamento do *dataset*.

3.4.1 ROS

O ROS, como já explicado na seção 2.3.1, duplica aleatoriamente os registros da classe minoritária até que elas, em volume, estejam iguais. Neste caso em específico, após a divisão dos *datasets*, o *dataset* de treinamento ficou com um total de 5.083.466 registros, destes aproximadamente 6.100 correspondiam a transações fraudulentas e para balancear o *dataset* o ROS necessitaria duplicar aleatoriamente 5.071.325 registros. O código do ROS pode ser encontrado no Apêndice H. A figura 13 ilustra as taxas de assertividade dos modelos preditivos.

	train	test	validation	training time
tree classifier	1.0	0.999	0.999	0:02:41.971565
random forest	1.0	0.999	0.999	0:27:33.624551
svm	1.0	0.999	0.999	0:24:56.213468

Figura 13 – Resultados dos modelos preditivos – *dataset* de treinamento balanceado (ROS).
Fonte: Autor.

De forma similar ao acontecido na seção anterior as taxas de assertividade, mostradas na figura 13, beiram os 100%. Da mesma forma como aconteceu com o *dataset* desbalanceado, utilizou-se de um quarto *dataset* – *fraud* – para validar o real desempenho dos modelos frente a transação fraudulentas – sem que houvesse, assim, a distorção causada pelo grande volume de registros não fraudulentos. A figura 14 ilustra o desempenho dos preditores no *dataset fraud* – composto pelo agrupamento das transações fraudulentas nos *datasets* de teste e validação.

	fraud	training time
tree classifier	0.719	0:02:41.971565
random forest	0.743	0:27:33.624551
svm	0.730	0:24:56.213468

Figura 14 – Resultados dos modelos preditivos (balanceado) – *fraud dataset* (ROS).
Fonte: Autor.

A figura 14, como pode-se observar, demonsta as taxas de assertividade – dos mesmos modelos utilizados anteriormente – no *dataset* de fraudes. Nota-se que a assertividade dos modelos caiu – indo de aproximadamente 100% para, no melhor dos casos, aproximadamente 75%. Quando se compara as figuras 12 e 14, nota-se, por conta do balanceamento, uma melhoria significativa nos resultados obtidos pelos modelos preditivos.

3.4.2 SMOTe

O SMOTe baseia-se na distância euclidiana entre os vizinhos para gerar novos registros e assim balancear o *dataset*. Diferentemente do ROS que duplica os registros da classe minoritária, o SMOTe, por meio dos atributos dos vizinhos, gera novos registros únicos. O Apêndice I ilustra a aplicação do SMOTe. Vale ressaltar que a evolução natural do SMOTe é o ADASYN (He, Haibo; Bai, Yang; Edwardo, Garcia; Li Shutao. 2008) e só não foi utlizado neste trabalho pelo excelente desempenho apresentado pelo SMOTe. A figura 15 ilustra as taxas de assertividade dos modelos preditivos.

	train	test	validation	training time
tree classifier	1.000	1.000	0.999	0:02:33.888025
random forest	1.000	0.999	0.999	0:31:12.724637
svm	0.916	0.966	0.966	0:28:47.734978

Figura 15 – Resultados dos modelos preditivos – *dataset* de treinamento balanceado (SMOTe).
Fonte: Autor.

De forma similar ao acontecido na seção anterior as taxas de assertividade, mostradas na figura 15, estão próximas aos 100%. Da mesma forma como aconteceu com o *dataset* desbalanceado e com o ROS, utilizou-se de o *dataset fraud* para validar o real desempenho dos modelos frente a transação fraudulentas – sem que houvesse, assim, a distorção causada pelo grande volume de registros não fraudulentos. A figura 16 ilustra o desempenho dos preditores no *dataset fraud* – composto pelo agrupamento das transações fraudulentas nos *datasets* de teste e validação.

	fraud	training time
tree classifier	0.821	0:02:33.888025
random forest	0.847	0:31:12.724637
svm	0.834	0:28:47.734978

Figura 16 – Resultados dos modelos preditivos (balanceado) – *fraud dataset* (SMOTe).
Fonte: Autor.

A figura 16, como pode-se observar, demonstra as taxas de assertividade – dos mesmos modelos utilizados anteriormente – no *dataset* de fraudes. Nota-se que a assertividade dos modelos caiu – indo de aproximadamente 100% para, no melhor dos casos, aproximadamente 85%. Quando se compara as figuras 15 e 16, nota-se, por conta do balanceamento, uma melhoria significativa nos resultados obtidos pelos modelos preditivos. O próximo capítulo irá comparar, de forma mais abrangente, o desempenho dos modelos preditivos nos três cenários de *datasets* (desbalanceado, balanceado pelo ROS e balanceado pelo SMOTe).

4 Considerações Finais

Este capítulo, com o objetivo de apresentar as conclusões deste estudo, divide-se em duas seções. A primeira comparará, de forma mais abrangente, o desempenho dos modelos preditivos nos três cenários criados – *dataset* desbalanceado, *dataset* balanceado pelo ROS e o *dataset* balanceado pelo SMOTe – e concluirá, baseando-se nas comparações, se o objetivo do trabalho foi alcançado ou não. A segunda, por sua vez, busca indicar pontos de melhoria neste trabalho e apresentará ideias de estudos futuros.

4.1 *Benchmarking* Final

O *dataset*, como explicado nas seções iniciais do capítulo 3, representa transações financeiras – geradas artificialmente – realizadas por meio de dispositivos *mobile* e foi gerado com o objetivo de suprir a falta de *datasets* públicos relacionados com o tema. O *dataset*, pela sua natureza, é desbalanceado – indicando que do total de transações realizadas apenas uma baixa quantidade é fraudulenta. Essa característica, por mais natural que seja, causa problemas nos preditores – causando *overfitting* e um alto número de erros (Falsos Negativos ou Falsos Positivos). A figura 17 demonstra a evolução das taxas de assertividade do *tree classifier*.

	train	test	validation	fraud
imbalanced	0.999	0.999	0.999	0.041
ros	1.000	0.999	0.999	0.719
smote	1.000	1.000	0.999	0.821

Figura 17 – Evolução das taxas de assertividade – *tree classifier*.
Fonte: Autor.

Como mostra a figura 17, as taxas de assertividade dos *datasets* de treinamento, teste e validação são basicamente as mesmas – fenômeno explicado pelo *class imbalance* (visto que somente o *dataset* de treinamento foi balanceado) – o que invalida qualquer tentativa de analisar a evolução do desempenho dos modelos por meio destes *datasets*. Com isso em mente, o *dataset fraud* foi utilizado para validar e quantificar as melhorias de desempenho que métodos como o ROS e o SMOTe trouxeram aos modelos preditivos ao balancear o *dataset* de treinamento. Observando a coluna *"fraud"* na figura 17, nota-se que o *tree classifier* sem hiperparametrização[7] ou algum tipo de busca pelo melhor conjunto de hiperparâmetros (*grid search*, por exemplo) saiu de uma assertividade de 0,041 no *dataset* desbalanceado, para 0,821 no *dataset* balanceado pelo SMOTe. A tabela 4 ilustra a evolução das matrizes de confusão no *dataset* de validação com o modelo *tree classifier*.

[7] Um parâmetro é um argumento de configuração interna do modelo preditivo e cujo valor pode ser estimado a partir dos dados de treinamento – como os pesos sinápticos de uma RNA – já os hiperparâmetros são argumentos que ajudam a definir o valor dos parâmetros (Claesen, M; Moor, B.D. 2015).

Tree Classifier - validation			
		Non-Fraud	**Fraud**
Imbalanced	**Non-Fraud**	635,459	0
	Fraud	766	31
		Non-Fraud	**Fraud**
ROS	**Non-Fraud**	635,102	357
	Fraud	239	654
		Non-Fraud	**Fraud**
SMOTe	**Non-Fraud**	635,152	307
	Fraud	102	701

Tabela 4 – Evolução das matrizes de confusão – *tree classifier.*
Fonte: Autor.

Nota-se, ao se observar a tabela 4, que o desempenho do modelo *tree classifier* no *dataset* de validação nos três diferentes cenários. No primeiro momento o *dataset* estava desbalanceado o que, por sua vez, acarretou alto número de falsos negativos – cerca de 31 acertos contra 766 erros – ocasionando uma assertividade de aproximadamente 0,038 (tratando-se apenas de transações realmente fraudulentas). Vale ressaltar que a assertividade geral no *dataset* desbalanceado foi de aproximadamente 0,999. Com o ROS nota-se uma melhoria de desempenho excepcional – indo de 0,038 para 0,732 (considerando somente as transações realmente fraudulentas). O SMOTe, por sua vez, obteve um desempenho ainda melhor – tendo em vista que o funcionamento do SMOTe gera, de fato, novos registros. A assertividade local do SMOTe (considerando apenas as transações realmente fraudulentas) foi de aproximadamente 0,948 – diminuindo consideravelmente a quantidade de falsos negativos (erro inaceitável). A tabela 5 demonstra a evolução das matrizes de confusão do *random forest*.

Random Forest - validation			
		Non-Fraud	**Fraud**
Imbalanced	**Non-Fraud**	635.437	0
	Fraud	794	31
		Non-Fraud	**Fraud**
ROS	**Non-Fraud**	635.202	280
	Fraud	183	597
		Non-Fraud	**Fraud**
SMOTe	**Non-Fraud**	635.222	260
	Fraud	92	688

Tabela 5 – Evolução das matrizes de confusão – *random forest.*
Fonte: Autor.

De forma semelhante ao acontecido com o *tree classifier* o *random forest*, ao classificar as transações fraudulentas, obteve uma melhora significativa em seu desempenho. A tabela 5 demonstra tal evolução. Nota-se que o número de falsos negativos passou de 794 – com a assertividade de aproximadamente 0,037 – para 92 falsos negativos – com a assertividade de aproximadamente 0,882 utilizando o SMOTe. A tabela 6 ilustra a evolução das matrizes de confusão da SVM.

		SVM - validation	
		Non-Fraud	**Fraud**
Imbalanced	**Non-Fraud**	635.459	0
	Fraud	766	31
		Non-Fraud	**Fraud**
ROS	**Non-Fraud**	635.102	357
	Fraud	239	654
		Non-Fraud	**Fraud**
SMOTe	**Non-Fraud**	635.152	307
	Fraud	91	712

Tabela 6 – Evolução das matrizes de confusão – SVM.
Fonte: Autor.

De forma semelhante ao acontecido com o *tree classifier* e com o *random forest* a SVM, ao classificar as transações fraudulentas, obteve uma melhora significativa em seu desempenho. A tabela 6 demonstra tal evolução. Nota-se que o número de falsos negativos passou de 766 – com a assertividade de aproximadamente 0,038 – para 92 falsos negativos – com a assertividade de aproximadamente 0,886 utilizando o SMOTe.

4.2 Conclusão

Diante dos resultados apresentados e explicados neste capítulo e baseando-se na evolução de métricas como taxa de assertividade geral, taxa de assertividade local (considerando apenas as transações realmente fraudulentas) e número de falsos negativos – tipo de erro inaceitável – pode-se concluir que técnicas de pré-processamento para normalização, redução de dimensionalidade, limpeza e balanceamento de classes são fundamentais para o sucesso de qualquer algoritmo de predição – independentemente de sua complexidade e custos computacionais. O *tree classifier*, por exemplo, um dos preditores mais simples utilizados neste trabalho, sem o balanceamento de classes, embora tenha apresentado uma ótima taxa de assertividade, apresentou uma matriz de confusão repleta de falsos negativos, o que, por sua vez, invalidava a utilização do modelo em ambiente de produção. O *random forest* e o SVM

além da melhoria de desempenho e diminuição de falsos negativos apresentaram, por conta da normalização e da redução de dimensionalidade, tiveram uma melhora em seu tempo de treinamento. A tabela 7 demonstra o comparativo das taxas de assertividade tendo como base as taxas de assertividade dos modelos no *dataset* de treinamento desbalanceado (sem o uso do ROS ou do SMOTe).

		train	test	validation	fraud
tree classifier	ROS	0.10%	0.00%	0.00%	94.30%
	SMOTe	0.10%	0.10%	0.00%	95.01%
random forest	ROS	0.00%	-0.10%	-0.10%	19.92%
	SMOTe	0.00%	-0.10%	-0.10%	29.75%
svm	ROS	0.10%	0.00%	0.00%	50.00%
	SMOTe	-9.06%	-3.42%	-3.42%	56.24%

Tabela 7 – Comparativo das taxas de assertividade.
Fonte: Autor.

Como pode-se observar na tabela 7 – principalmente na coluna *fraud* – as taxas de assertividade dos três modelos utilizados neste trabalho ao longo das otimizações feitas nos *datasets* de treinamento. Na primeira linha da tabela pode-se observar o desempenho do *tree classifier* nos *datasets* otimizados, respectivamente, pelo ROS e SMOTe. Nota-se que a correção do *class imbalance* pelo ROS e SMOTe, respectivamente, trouxeram, ao *tree classifier*, no *dataset fraud*, uma melhora de mais de 94,30% e 95,01% (indo de 0,041 para 0,719 com o ROS e 0,821 com o SMOTe). Para o *random forest*, as melhorias, por conta do desempenho inicial no *dataset* de treinamento desbalanceado, não foram tão acentuadas como no *tree classifier* – aumentando as taxas de assertividade em cerca de 19,92% com o ROS e 29,75% com o SMOTe. Por fim, o SVM – em desempenho foi o segundo termo entre o *tree classifier* e o *random forests* – sua assertividade no *dataset fraud* aumentou cerca de 50% com o ROS e 56,24% com o SMOTe.

Em suma, como já mencionado anteriormente, técnicas de pré-processamento são fundamentais para qualquer projeto de *data science* em que os dados adquiridos, muitas vezes, são bagunçados. De modo geral tais técnicas, por melhorarem significativamente o desempenho dos modelos preditores num *dataset* complexo, tornaram este trabalho um sucesso – tanto por ter agregado conhecimentos técnicos de *data science* e aprendizado de máquina – algoritmos de classificação, técnicas de pré-processamento e *data science* – quanto por ter trazido à tona as dificuldades encontradas por equipes de *data science* que trabalham com predições de transações fraudulentas.

4.3 Sugestões futuras

Como propostas de trabalhos futuros, sugere-se:

- Testar o ADASYN (a evolução natural do SMOTe) e comparar seu desempenho com outras técnicas de balanceamento – como por exemplo o próprio SMOTe;
- Comparar o desempenho dos preditores criados neste trabalho com o desempenho do preditor utilizado pela *Nowergian University of Science and Technology* que gerou a coluna do *isFlaggedFraud* do *dataset* original;
- Utilizar GANs para:
    - Balancear o *dataset* pelor *Generator*, que deverá gerar novos registros que correspondem a transações financeiras;
    - Validar os dados gerados pelo *Generator*;
    - Treinar – usando os dados gerados pelo *Generator* e os dados reais – o *Discriminator* para classificar as transações.

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.

CLAESEN, Marc; MOOR, Bart De. **Hyperparameter Search in Machine Learning.** 2015.
Disponível em: <https://arxiv.org/pdf/1502.02127.pdf>. Acesso em: 14 de maio de 2020.

Didática Tech. **Underfitting e Overfitting.** 2019. Acesso em 16 de maio de 2020. Disponível em < https://didatica.tech/underfitting-e-overfitting/>.

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))
```

Apêndice B – N.A. Values

In [1]:
```
dict_na = {
       'columns': list(df.columns),
       'na': []
}

for i in range(len(df.columns)):
              dict_na.get('na').append(sum(df[df.columns[i]].isna()))

pandas.DataFrame(dict_na).set_index('columns')
```

Out [1]:

columns	na
step	0
type	0
amount	0
oldbalanceOrg	0
newbalanceOrig	0
oldbalanceDest	0
newbalanceDest	0
isFraud	0

Apêndice C – Variável *"type"*

```
In [2]:      df.loc[df['type'] == 'CASH_IN', 'type'] = 0
             df.loc[df['type'] == 'CASH_OUT', 'type'] = 1
             df.loc[df['type'] == 'DEBIT', 'type'] = 2
             df.loc[df['type'] == 'PAYMENT', 'type'] = 3
             df.loc[df['type'] == 'TRANSFER', 'type'] = 4
```

Apêndice D – Redução de Dimensionalidade

```
In [3]:      lm = linear_model.LinearRegression().fit(x, y)
             attr_reduction = SelectFromModel(lm, prefit=True)
             df_reduced = pandas.DataFrame(attr_reduction.transform(x))
```

Apêndice E – MinMax Standartization

```
In [4]:      scaler = preprocessing.MinMaxScaler().fit(df_reduced)
             df_reduced_norm = pandas.DataFrame(scaler.transform(df_ reduced))
```

Apêndice F – Criação do *fraud dataset*

```
In [5]:      df_fraud, fraud_validation = helpers.data_separation(df_fraud)
             x_fraud, y_fraud = helpers.x_and_y_separation(
                    fraud_validation.reset_index(drop=True)
             )
             y_fraud = pandas.DataFrame(y_fraud)
```

Apêndice G – Preditores

Apêndice G.1

```
In [6]:      d = datetime.datetime.now()
             model = DecisionTreeClassifier().fit(x_train, y_train)
             helpers.time_screening(d)

Out [6]:     0:02:33.888025
```

Apêndice G.2

```
In [7]:      d = datetime.datetime.now()
             model = DecisionTreeClassifier().fit(x_train, y_train)
             helpers.time_screening(d)

Out [7]:     0:31:12.724637
```

Apêndice G.3

```
In [8]:      d = datetime.datetime.now()
             model = LinearSVC(C=0.01, penalty="l1", dual=False,
             max_iter=10000).fit(x_train, y_train.isFraud.values)
             helpers.time_screening(d)

Out [8]:     0:28:47.734978
```

Apêndice H – *Random Oversampling* (ROS)

```
In [9]:      ros = RandomOverSampler(random_state=0)
             x_train, y_train = ros.fit_resample(x_train, y_train)
```

Apêndice I – SMOTe

```
In [11]:     x_train, y_train = SMOTE().fit_sample(x_train, y_train)
```

Anexo A – *Support Vectors*

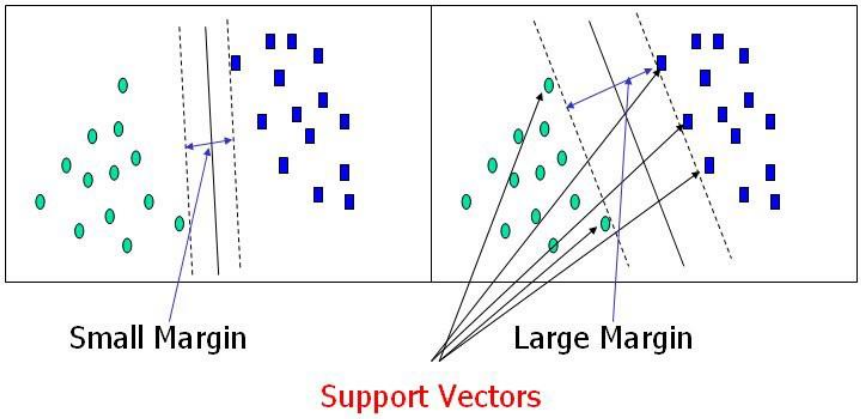

Figura 18 – *Support Vectors.*
Fonte: Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

Anexo B – Hiperplanos

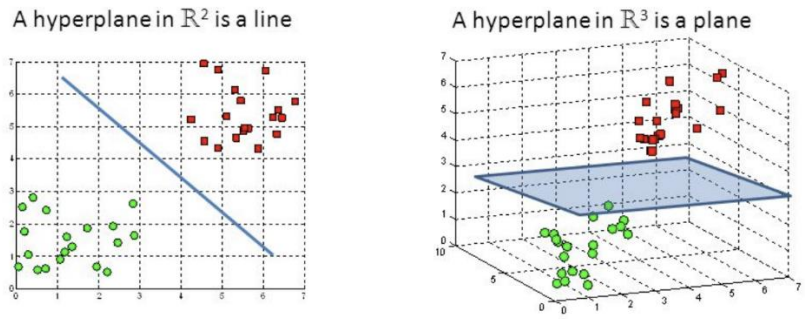

Figura 19 – Hiperplanos.
Fonte: Gandhi, Rohith. **Support Vector Machine – Introduction to ML Algorithms.**

Anexo C – *Overfitting*

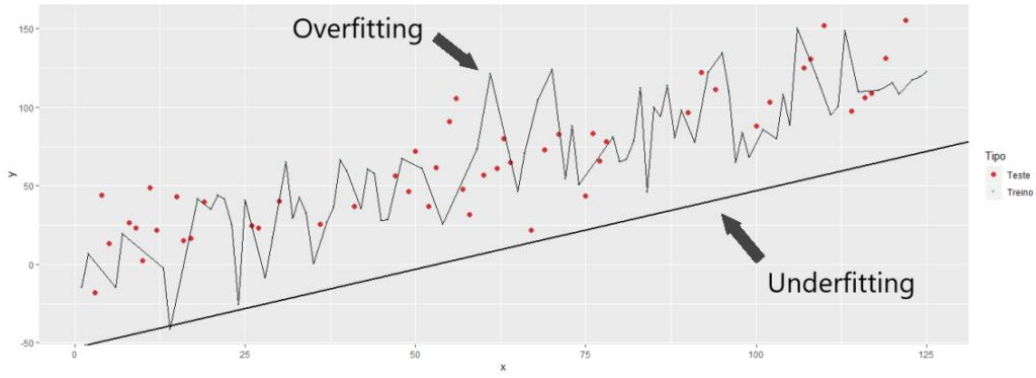

Figura 20 – *Overfitting.*
Fonte: Didática Tech. ***Underfitting* e *Overfitting*.**