XXIV SEMEAD
Seminários em Administração
novembro de 2021
ISSN 2177-3866

Machine Learning para detecção de transações financeiras fraudulentas

BRUNO HENRIQUE PAES
ESCOLA SUPERIOR DE PROPAGANDA E MARKETING (ESPM)

ANTONIO MARCOS SELMINI
ESCOLA SUPERIOR DE PROPAGANDA E MARKETING (ESPM)

Machine Learning para detecção de transações financeiras fraudulentas

1. INTRODUÇÃO

Como parte importante de uma contínua evolução, a tecnologia e a internet têm proporcionado profundas mudanças em todas as áreas, o que não poderia ser diferente com o mundo dos negócios, notadamente com o mercado financeiro. A internet, por sua vez, proporcionou o surgimento de novas formas de fazer negócio, mas trouxe também eventos indesejados que passaram a ocorrer com maior frequência causando transtornos aos indivíduos e as instituições financeiras que praticam negócios online.

De todos os campos da ciência da computação, a Inteligência Artificial (IA) sempre intrigou os pesquisadores no sentido de construir máquinas ou sistemas inteligentes que pudessem automatizar tarefas cotidianas. Com a evolução tecnológica, novas técnicas de Inteligência Artificial (IA) vêm sendo desenvolvidas e aplicadas na área financeira para identificar se uma transação é fraudulenta e tornar os sistemas mais seguros, buscando evitar fraudes e quaisquer tipos de problemas que impactam os negócios e os clientes. Um subcampo da IA em constante evolução e aperfeiçoamento é a Aprendizagem de Máquina.

Atualmente, tem-se aplicado Aprendizagem de Máquina no mercado financeiro para entender e analisar os dados de uma forma mais simples e fornecer diversos serviços por meio delas. 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.

2. OBJETIVOS E JUSTIFICATIVAS

Os avanços tecnológicos, trazidos por anos de pesquisa e desenvolvimento, proporcionaram 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 a 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 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.

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, ou seja, modelos que maximizem a taxa de assertividade e minimizem o número de falsos negativos (transações fraudulentas classificadas como não fraudulentas), que é um tipo de erro inaceitável neste problema de classificação.

3. FRAUDES FINANCEIRAS

Fraudes financeiras não são questões relacionadas somente a bancos e instituições financeiras. Produtos ou serviços negociados através de sistemas digitais – mesmo aqueles em que não há a troca 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).

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).

4. APRENDIZAGEM DE MÁQUINA

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).

4.1. *Tree Classifiers*

Tree Classifiers ou *Decision Trees* são um dos métodos de Aprendizado de Máquina Supervisionado mais simplórios (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 se dividem 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 feita através do caminho percorrido a partir do nó raiz até o nó folha. A Figura 1 ilustra uma árvore de decisão com gerada com o *Iris dataset* – composto por 150 registros de medidas das plantas da família *Iris*.

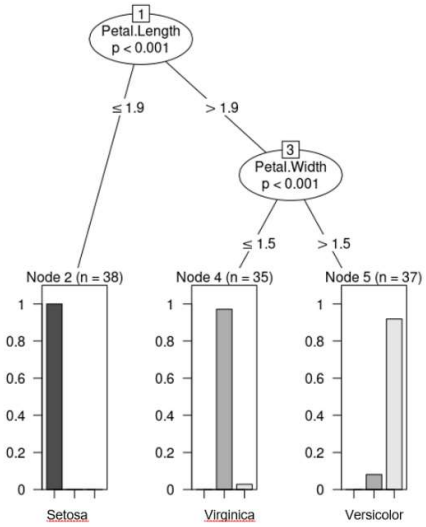

Figura 1 – *Decision Tree* do *Iris dataset.*
Fonte: Autores.

4.2. *Random Forests Classifiers*

Um outro algoritmo de classificação são os *Random Forests*, o qual desempenha o papel de diversas árvores de decisão, operando como um conjunto único. Surgiu 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* – método que utiliza múltiplos algoritmos de aprendizado num único modelo preditivo (Opitz, D; Maclin, R. 1999). O funcionamento de um *Random Forest* está apresentado na Figura 2.

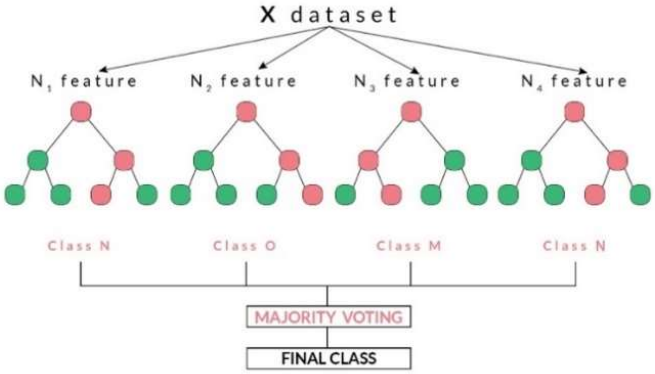

Figura 2 – *Random Forests.*
Fonte: Autores.

Como ser observado na Figura 2, o modelo 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, a 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).

4.3. *Support Vector Machines*

Support Vector[1] *Machines* ou 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).

Segundo Haykin, a Máquina de Vetor de Suporte é uma máquina linear com algumas propriedades interessantes. A principal ideia de uma SVM 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[2] 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 3 ilustra alguns dos possíveis hiperplanos que uma SVM construiria para separar duas classes de *datapoints*.

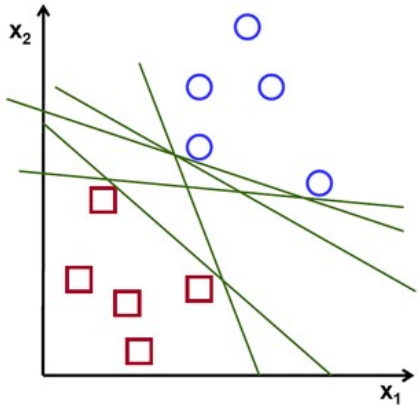

Figura 3 – Possíveis Hiperplanos.
Fonte: Gandhi, Rohith. **SVM – Introduction to ML Algorithms.**

Como pode ser notado na Figura 3 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 4 ilustra a configuração de um hiperplano ótimo.

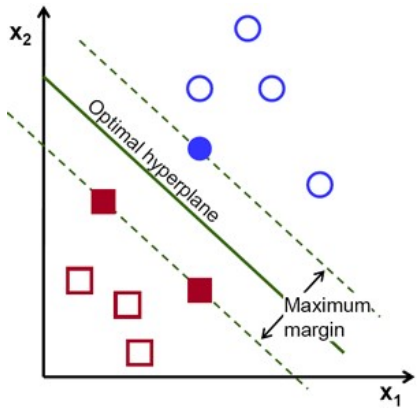

Figura 4 – Hiperplano Ótimo.
Fonte: Gandhi, Rohith. **SVM – Introduction to ML Algorithms.**

5. *SAMPLING*

Esta seção busca fundamentar duas técnicas de *sampling* que buscam resolver o *class imbalance*, ou seja, buscam balancear a quantidade de elementos de classes minoritárias com a classe majoritária para que os *datasets* possam ser utilizados no treinamento dos mais diversos modelos preditivos. *Imbalanced datasets* são caracterizados por uma severa inclinação na distribuição de suas classes (por exemplo *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).

5.1. *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*. As duas abordagens são:

- O *Random Undersampling* (RUS) consiste em excluir aleatoriamente exemplos da classe majoritária, ou seja, seleciona aleatoriamente exemplos da classe majoritária e o exclui do *dataset* de treinamento.
- O *Random Oversampling* (ROS) consiste em duplicar aleatoriamente exemplos da classe minoritária, ou seja, seleciona aleatoriamente exemplos 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).

5.2. *SMOTe*

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

Como pode ser notado na Figura 5, 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.

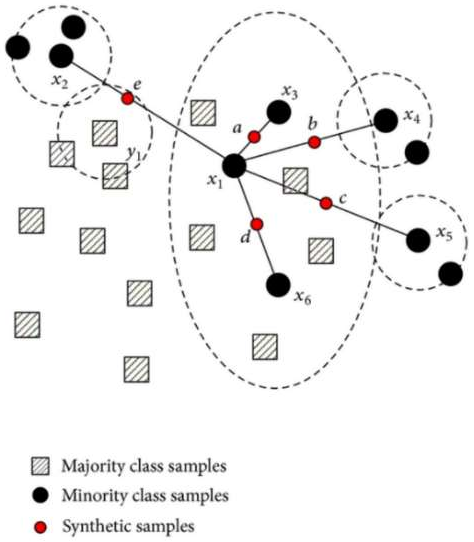

Figura 5 – Funcionamento do SMOTe.
Fonte: Battacharyya, Indresh. **SMOTe and ADASYN.**

6. BASE 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*. Segundo Lopez-Rojas *et al*. (2016) existe uma falta de bases de dados públicas principalmente na área de transações monetárias realizadas em aplicativos móveis por ser uma área relativamente nova. Os autores acreditam que boa parte desse problema se deve a natura privada das transações financeiras, levando dessa forma a não publicação dos dados (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* apresentam cerca de 31 dias de transações financeiras.

6.1. Análise Exploratória

O *dataset* é composto por 11 atributos – numéricos ou não – e possui 6.354.407 registros. Cada atributo é explicado na Tabela 1.

Tabela 1 – Explicação dos atributos do *dataset.*

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 marcada como fraude pelo sistema da NTNU?	0

Fonte: Autores.

Ao se analisar o comportamento da variável *amount* (valor total da transação – como pode ser observado na Tabela 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 6.

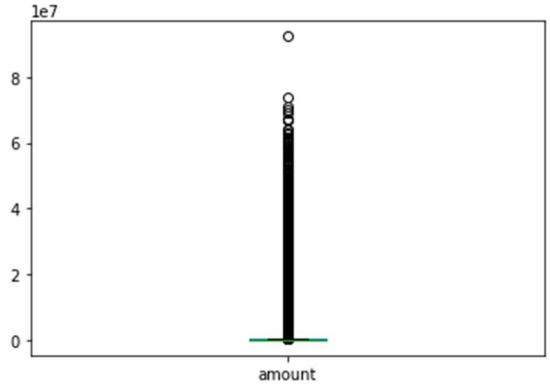

Figura 6 – *Boxplot* da variável *amount.*
Fonte: Autores.

Ao se observar a Figura 6, nota-se que a variável *amount*, em sua grande parte, é composta por valores *outliers*, tem desvio padrão de $ 603.858,23 e valor médio 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 trazer problemas durante o treinamento dos modelos preditivos.

6.2. 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 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* e outras anomalias estatísticas.

Com o intuito de otimizar o desempenho dos preditores – menor custo computacional, mas se mantendo 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 mais impactavam na variável dependente (*isFraud*).

Por conta de sua simplicidade e desempenho, optou-se por utilizar a regressão linear (que verificou a relevância estatística de cada variável) 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, 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*. 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 utilização do princípio de Pareto ou a proporção de 80/20 (GUYON, Isabelle. 1997) – 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 a testes e os 10% restantes à validação dos modelos – no entanto, independentemente da bibliografia, dos autores 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 da7dos 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 Tabela 2 ilustra a divisão do *dataset*.

Tabela 2 – Divisão do *dataset.*

sample	fraud_proportion	non_fraud_proportion	fraud_exemples	Non_fraud_exemples	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

Fonte: Autores.

Ao se observar a Tabela 2 nota-se que após a divisão do *dataset* original – treinamento, teste e validação – a proporcionalidade original de exemplos que correspondiam a fraudes e não fraudes se manteve, ou seja, cada *dataset* possui os mesmos 0,12% de registros correspondentes a transações fraudulentas.

7. *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 a 1 e 0 – esperava-se, inicialmente, que o *dataset* contivesse proporções próximas a 50% para cada classe, o que, como visto nas seções anteriores, não ocorreu e que poderia acarretar em *Overfitting* dos modelos preditivos. O *Overfitting*, por sua vez, poderia induzir os modelos preditivos a 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 menor tolerância – visto que na prática enviar uma transação a um analista humano é um cenário menos ruim do que não considerar fraude uma transação fraudulenta.

7.1. *Benchmarking* Preliminar

Os algoritmos – *Tree Classifiers*, *Random Forests* e SVM – usados neste trabalho foram escolhidos por sua eficiência e popularidade em diversos problemas de classificação. Os *datasets*, anteriormente separados, foram utilizados para o treinamento, teste e validação dos modelos, entretanto, um quarto *dataset* – *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, assim, a distorção causada pelo massivo volume de registros não fraudulentos. A Tabela 3 demonstra o desempenho dos três modelos utilizados.

Tabela 3 – Resultados dos modelos preditivos – *dataset* de treinamento desbalanceado.

	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

Fonte: Autores.

Como pode ser observado, os três modelos – *Tree Classifier*, *Random Forest* e SVM – 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*. A Tabela 4, como se pode observar, demonstra as taxas de assertividade – dos mesmos modelos utilizados anteriormente – no *dataset* de fraudes.

Tabela 4 – Resultados dos modelos preditivos (desbalanceado) – *fraud dataset.*

	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

Fonte: Autores.

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 Tabelas 3 e 4 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 5, 6 e 7 demonstram as Matrizes de Confusão dos modelos preditivos.

Tabela 5 – Matrizes de Confusão – *tree classifier*

Tree Classifier								
	Train		Test		Validation		Fraud	
	Non-Fraud	Fraud	Non-Fraud	Fraud	Non-Fraud	Fraud	Non-Fraud	Fraud
Non-Fraud	5.083.511	0	635.437	0	635.459	0	0	0
Fraud	6.314	271	794	31	766	31	1.575	68

Fonte: Autores.

Como se pode observar na Tabela 5, 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 enorme quantidade de Falsos Negativos (mesmo predizendo dados já vistos pelo modelo no treinamento).

Tabela 6 – Matrizes de Confusão – *random forests.*

Random Forests								
	Train		Test		Validation		Fraud	
	Non-Fraud	Fraud	Non-Fraud	Fraud	Non-Fraud	Fraud	Non-Fraud	Fraud
Non-Fraud	5.083.511	0	634.430	7	635.450	9	0	0
Fraud	0	6.585	19	656	154	649	664	979

Fonte: Autores.

Como se pode observar na Tabela 6, 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 fazer predições corretas. 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 uma acurácia similar. O modelo SVM apresentou desempenho similar ao *Random Forest*, o que fica evidenciado na Tabela 7.

Tabela 7 – Matrizes de Confusão – SVM.

SVM								
	Train		Test		Validation		Fraud	
	Non-Fraud	Fraud	Non-Fraud	Fraud	Non-Fraud	Fraud	Non-Fraud	Fraud
Non-Fraud	5.083.406	105	635.422	15	635.451	8	0	0
Fraud	537	2.375	537	288	499	304	1.044	599

Fonte: Autores.

Como se pode observar nas Tabelas 5, 6 e 7, 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 sua validação – mascarando as altas taxas de assertividade. Nas três Tabelas, nota-se 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 a desejar.

7.2. 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*. 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*.

7.2.1. Utilização do ROS

O ROS, como já explicado, duplica aleatoriamente os registros da classe minoritária até que elas, em volume, estejam iguais à classe majoritária. 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. A Tabela 8 ilustra as taxas de assertividade dos modelos preditivos.

Tabela 8 – Resultados dos modelos preditivos – *dataset* de treinamento balanceado (ROS).

	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

Fonte: Autores.

De forma similar ao acontecido na seção anterior as taxas de assertividade, mostradas na Tabela 8, 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 Tabela 9 ilustra o desempenho dos preditores no *dataset fraud* – composto pelo agrupamento das transações fraudulentas nos *datasets* de teste e validação.

Tabela 9 – Resultados dos modelos preditivos (balanceado) – *fraud dataset* (ROS).

	fraud	training time
tree classifier	0,719	0:02:41,971565
random forest	0,743	0:27:33,624551
SVM	0,730	0:24:59,213468

Fonte: Autores.

A Tabela 9, como se pode 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 75%. Quando se compara as Tabelas 4 e 9, nota-se, por conta do balanceamento, uma melhoria significativa nos resultados obtidos pelos modelos preditivos.

7.2.2. Utilização do SMOTe

Como já explicado em sua seção, o SMOTe se baseia 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. Vale ressaltar que a evolução natural do SMOTe é o ADASYN (He, Haibo; Bai, Yang; Edwardo, Garcia; Li Shutao. 2008) e só não foi utilizado neste trabalho pelo excelente desempenho apresentado pelo SMOTe. A Tabela 10 ilustra as taxas de assertividade dos modelos preditivos.

Tabela 10 – Resultados dos modelos preditivos – *dataset* de treinamento balanceado (SMOTe).

	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

Fonte: Autores.

De forma similar ao acontecido na seção anterior as taxas de assertividade, mostradas na Tabela 10, 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 Tabela 11 ilustra o desempenho dos preditores no *dataset fraud* – composto pelo agrupamento das transações fraudulentas nos *datasets* de teste e validação.

Tabela 11 – Resultados dos modelos preditivos (balanceado) – *fraud dataset* (SMOTe).

	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

Fonte: Autores.

A Tabela 11, como se pode observar, demonstra as taxas de assertividade – dos mesmos modelos utilizados anteriormente – no *dataset fraud*. Nota-se que a assertividade dos modelos caiu – indo de aproximadamente 100% para, no melhor dos casos, aproximadamente 85%. Quando se compara as Tabelas 4, 9 e 11, nota-se, por conta do balanceamento, uma melhoria significativa nos resultados obtidos pelos modelos preditivos.

CONSIDERAÇÕES FINAIS

O *dataset*, como já explicado, 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 Tabela 12 demonstra a evolução das taxas de assertividade do *Tree Classifier*.

Tabela 12 – Evolução das taxas de assertividade – *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

Fonte: Autores.

Como mostra a Tabela 12, 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 Tabela 12, nota-se que o *Tree Classifier* sem hiperparametrização[3] 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 13 ilustra a evolução das Matrizes de Confusão no *dataset* de validação com o modelo *Tree Classifier*.

Tabela 13 – Evolução das Matrizes de Confusão – *Tree Classifier.*

Tree Classifier – validation			
		Non-Fraud	**Fraud**
imblanced	**Non-Fraud**	635.459	0
	Fraud	766	31
ROS	**Non-Fraud**	635.102	357
	Fraud	239	654
SMOTe	**Non-Fraud**	635.152	307
	Fraud	102	701

Fonte: Autores.

Nota-se, ao se observar a Tabela 13, 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 14 demonstra a evolução das Matrizes de Confusão do *random forest*.

Tabela 14 – Evolução das Matrizes de Confusão – *Random Forests.*

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

Fonte: Autores.

De forma semelhante ao acontecido com o *Tree Classifier* o *Random Forest*, ao classificar as transações fraudulentas, obteve-se uma melhora significativa em seu desempenho. A Tabela 14 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 15 ilustra a evolução das Matrizes de Confusão da SVM.

Tabela 15– Evolução das Matrizes de Confusão – SVM.

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

Fonte: Autores.

De forma semelhante ao acontecido com o *Tree Classifier* e com o *Random Forest* a SVM, ao classificar as transações fraudulentas, obteve-se uma melhora significativa em seu desempenho. A Tabela 15 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.

CONCLUSÃO

Diante dos resultados apresentados e explicados e baseado 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 16 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).

Tabela 16 – Comparativo das taxas de assertividade.

		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%

Fonte: Autores.

Como se pode observar na Tabela 16 – 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 agregarem conhecimentos técnicos de *data science* e Aprendizado de Máquina, quanto por ter trazido à tona as dificuldades encontradas por equipes de *data science* que trabalham com predições de transações fraudulentas.

NOTAS DE FIM

1 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).

2 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).

3 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).

REFERÊNCIAS BIBLIOGRÁFICAS

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.

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.

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.

GUYON, Isabella. **A Scaling law for the validation-set training-set size ratio**. 1997. Disponível em: < https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.1337>. Acesso em: 15 de maio 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.

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.

YIU, Anthony. **Understanding Random Forest**. 2019. Disponível em: <https://towardsdatascience.com/understanding-random-forest>. Acesso em: 15 de abril de 2020.