PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ESTUDO DE ALGORITMOS DE *MACHINE LEARNING* PARA PREDIÇÃO DE FRAUDES EM CARTÕES DE CRÉDITO

TÚLIO CÉSAR LEITE DE SOUSA

GOIÂNIA
2021

TÚLIO CÉSAR LEITE DE SOUSA

ESTUDO DE ALGORITMOS DE *MACHINE LEARNING* PARA PREDIÇÃO DE FRAUDES EM CARTÕES DE CRÉDITO

Trabalho de Conclusão de Curso apresentado à Escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.
Orientador:
Prof. Me. Aníbal Santos Jukemura
Banca Examinadora:
Profa. Ma. Lucília Gomes Ribeiro
Prof. Dr. Gustavo Siqueira Vinhal

GOIÂNIA
2021

TÚLIO CÉSAR LEITE DE SOUSA

ESTUDO DE ALGORITMOS DE MACHINE LEARNING PARA PREDIÇÃO DE FRAUDES EM CARTÕES DE CRÉDITO

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, e aprovado em sua forma final pela escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás em _____/_____/_____.

_____
Profa. Ma. Ludmilla Reis Pinheiro dos Santos
Coordenadora de Trabalho de Conclusão de Curso

Banca Examinadora:

_____
Orientador: Prof. Me. Aníbal Santos Jukemura

_____
Profa. Ma. Lucília Gomes Ribeiro

_____
Prof. Dr. Gustavo Siqueira Vinhal

GOIÂNIA
2021

A Deus por me promover esse momento excepcional.
A minha família por me encorajar e apoiar em toda essa jornada.

AGRADECIMENTOS

Primordialmente, agradeço a Deus pelas oportunidades que me proporcionou e por sempre guiar meus caminhos.

A minha mãe, Antonia Leite de Jesus Sousa, e a minha irmã, Mirele Leite de Sousa, por todo o apoio, encorajamento e incentivo, sem vocês duas, eu não chegaria até aqui.

Ao meu orientador, Aníbal Santos Jukemura, pelas admiráveis orientações e sabedoria compartilhada, além de toda a dedicação e esforço a este trabalho.

A todos os professores, que juntos contribuíram com a minha formação. Em especial, a professora Lucília Gomes Ribeiro e o professor Gustavo Siqueira Vinhal, que aceitaram compor a banca examinadora deste trabalho.

A todos que direta ou indiretamente colaboraram para a realização do meu trabalho.

"When somebody has a breakthrough innovation, it is rarely one little thing. Very rarely, is it one little thing. It's usually a whole bunch of things that collectively amount to a huge innovation."

Elon Musk

RESUMO

Com a maior demanda por pagamentos realizados por cartões de crédito, fraudes nessas transações são cada vez mais frequentes. Consequentemente, são causados impactos negativos a instituições financeiras, agravados por uma inviabilidade técnica na manutenção adequada da segurança tecnológica, gerando prejuízos às empresas integrantes desse mercado. Dado esse cenário, este trabalho avaliou os modelos preditivos *k-Nearest Neighbors*, *Support Vector Machines* e *Random Forest* aplicados a um conjunto de dados (*dataset*) de transações de cartões de crédito pré-processado, a fim de identificar a abordagem mais adequada para predições de fraudes em transações de cartões de crédito. A metodologia aplicada envolveu analisar os hiperparâmetros dos algoritmos, com o propósito de otimizar os resultados, e realizar uma classificação para cada subconjunto do *dataset*, os quais passaram por procedimentos de seleção de atributos e balanceamento de dados.

Palavras-chave: *Análise preditiva; Fraudes em transações; Cartão de crédito; Machine Learning.*

ABSTRACT

With the increased demand for payments made by credit cards, fraud in these transactions is more and more frequent. Consequently, negative impacts are caused to financial institutions, aggravated by a technical infeasibility in the adequate maintenance of technological security, generating losses to the companies that are part of this market. Given this scenario, this work evaluated the predictive models k-Nearest Neighbors, Support Vector Machines and Random Forest applied to a pre-processed credit card dataset to identify the most appropriate approach for predicting fraud in credit card transactions. The applied methodology involved analyzing the hyperparameters of the algorithms, with the purpose of optimizing the results, and performing classification for each subset of the dataset, which went through attributes selection and data balancing procedures.

Keywords: *Predictive analysis; Transaction fraud; Credit cards; Machine Learning.*

LISTA DE ILUSTRAÇÕES

Figura 1 - Indução de classificação em Aprendizado Supervisionado. 17
Figura 2- Exemplo da regra de classificação do k-NN. 18
Figura 3 - Hiperplano com margem pequena (a) e com margem máxima (b). 20
Figura 4 - Representação de uma árvore de decisão. 21
Figura 5 - Tabela dos resultados obtidos no trabalho de Nicola, Lauretto e Delgado (2020). 24
Figura 6 - Tabela demonstrando os resultados F1-*Score* Ótimos. 25
Figura 7 - Gráfico mostrando o desbalanceamento das classes. 30
Figura 8 - Gráfico das classes após a aplicação da técnica *Undersampling*. 31
Figura 9 – Gráfico das classes após a aplicação da técnica SMOTE. 32
Figura 10 - Exemplo do uso do *RandomForestClassifier*. 36
Figura 11 - Busca pelo hiperpâmetro *n_neighbors* com balanceamento RU. 38
Figura 12 - Busca pelo hiperpâmetro n_neighbors com balanceamento SMOTE. 39
Figura 13 - Busca pelo hiperpâmetro Kernel com balanceamento RU. 39
Figura 14 - Busca pelo hiperpâmetro Kernel com balanceamento SMOTE. 40
Figura 15 - Busca pelo hiperparâmetro n_estimators com balanceamento RU. 41
Figura 16 - Busca pelo hiperparâmetro n_estimators com balanceamento SMOTE. 41
Figura 17 – Gráfico com resultados alcançados nos experimentos com k-NN. 43
Figura 18 – Gráfico com resultados alcançados nos experimentos com SVM. 45
Figura 19 – Gráfico com resultados alcançados nos experimentos com RF. 47

LISTA DE TABELAS

Tabela 1 - Acurácia alcançada nos experimentos com k-NN. 42
Tabela 2 - F1-*Score* alcançado nos experimentos com k-NN. 42
Tabela 3 - Acurácia alcançada nos experimentos com SVM. 44
Tabela 4 - F1-*Score* alcançado nos experimentos com SVM.....44
Tabela 5 - Acurácia alcançada nos experimentos com RF. 46
Tabela 6 - F1-Score alcançado nos experimentos com RF. 46
Tabela 7 – Média entre Acurácia e F1-*Score* alcançada nos experimentos com k-NN. 48
Tabela 8 – Média entre Acurácia e F1-*Score* alcançada nos experimentos com SVM. 48
Tabela 9 - Média entre Acurácia e F1-*Score* alcançada nos experimentos com RF. 48

LISTA DE SIGLAS

DB	*Data Base*
GPUs	*Graphics Processing Units*
IA	*Inteligência Artificial*
k-NN	*k-Nearest Neighbors*
PCA	*Principal Component Analysis*
PK	*Prototypes Kluster*
RBF	*Radial Basis Function*
RF	*Random Forest*
RO	*Random Oversampling*
RU	*Random Undersampling*
SMOTE	*Synthetic Minority Oversampling Technique*
SVM	*Support Vector Machines*
ULB	*Université Libre de Bruxelas*

SUMÁRIO

1 INTRODUÇÃO 12
1.2 Objetivo geral e específicos 13
1.3 Justificativa 14
1.4 Resultados Esperados 14
1.5 Estrutura do Trabalho 14

2 REVISÃO BIBLIOGRÁFICA 16
2.1 *Machine Learning* 16
***2.1.1 K-Nearest Neighbors* 17**
***2.1.2 Support Vector Machines* 19**
2.1.2.1 Hiperplano 20
***2.1.3 Random Forest* 20**

3 TRABALHOS RELACIONADOS 23

4 AMBIENTE DE PRODUÇÃO E METODOLOGIA DE DESENVOLVIMENTO 27
4.1 Ambiente 27
***4.1.1 Python* 27**
***4.1.2 Bibliotecas* 28**
4.2 Metodologia 29
***4.2.1 Conjunto de dados* 29**
4.2.1.1 Análise de Componentes Principais (PCA) 29
***4.2.2 Balanceamento de dados* 30**
***4.2.3 Busca por hiperparâmetros* 32**
***4.2.4 Seleção de atributos* 33**
***4.2.5 Métricas de avaliação* 33**
***4.2.6 Treinamento, teste e predição* 35**

5 RESULTADOS 37

5.1 Seleção de hiperparâmetros 37
5.1.1 Número de vizinhos (n_neighbors) 37
5.1.2 Funções Kernel SVM (Kernel) 39
5.1.3 Árvores na floresta (n_estimators) 40
5.2 Avaliação dos Modelos Preditivos 41
5.2.1 k-NN 42
5.2.2 SVM 43
5.2.3 RF 45
5.2.4 Análise Geral 47

6 CONCLUSÃO 49

REFERÊNCIAS 52

1 INTRODUÇÃO

Ao longo das últimas décadas observa-se um aumento significativo na quantidade de transações financeiras eletrônicas. Um estudo que apresenta índices publicados em julho de 2017 pelo Banco Central revela que as transações com cartão de crédito somaram seiscentos e setenta e quatro bilhões de reais, o que reflete um aumento de 3% em relação a 2015 (VIEIRA, 2018). Associada ao grande volume financeiro envolvido e ao tráfego de informações sigilosas, há uma atenção especial de criminosos com o objetivo de obter vantagens financeiras de forma ilícita.

Existem vários motivos pelos quais se alcançou o crescimento da utilização dos cartões de crédito, principalmente devido à popularização do *e-commerce*. Além disso, seu uso é prático possibilitando diversas formas de pagamento sob várias condições determinadas. Há também a possibilidade de se obter crédito instantâneo, com gastos concentrados na fatura que podem ser pagas com até 30 dias corridos, além de ser aceito em estabelecimentos variados, já para aqueles que recebem pagamentos com cartões, o risco de haver inadimplência é notavelmente pequeno, é mais fácil manter o controle das vendas e há muitas pessoas dispostas a pagarem suas compras no crédito (OLIVEIRA, 2016).

Os cartões, sejam de crédito ou débito, são mecanismos de pagamento altamente utilizados no comércio em geral. Apesar disso, há espaço para que eles sejam ainda mais usados (OLIVEIRA, 2016). Por se tratar de um meio de pagamento, o cartão de crédito está profundamente ligado aos dados financeiros e pessoais dos usuários, que despertam o interesse de fraudadores. Por essa razão, o uso dessa tecnologia exige mecanismos que reflitam ampla segurança. Caso haja fraudes muito frequentes envolvendo muitos custos, os cartões perdem sua confiabilidade pelo público, gerando a diminuição de seu uso pela maioria das pessoas (OLIVEIRA, 2016). Por consequência, quanto maior a intensidade e o volume financeiro envolvido nas fraudes, maiores serão os prejuízos das empresas integrantes desse mercado, gerando assim, uma inviabilidade técnica na manutenção adequada e segura desse tipo de solução financeira.

De acordo com o Serasa, ocorreram um milhão, novecentas e sessenta mil tentativas de fraude em 2017, sendo o maior índice em três anos. O montante representa ainda alta de 8,2% em relação ao ano anterior. Em meio às tentativas dessas fraudes, o que mais se encontra é a modalidade com cartões de crédito, que

são efetuadas a partir da utilização de unidades falsas ou roubadas. Existem padrões de compras para cada cliente, e o desafio das operadoras de cartões é identificar esses padrões para a identificação de transações suspeitas. Em geral, tais práticas ilícitas geram prejuízos para o cliente, governo, comerciantes, organizações corporativas e financeiras e demais envolvidos (VIEIRA, 2018).

Em uma primeira análise, quando uma fraude é bem-sucedida, poderia se dizer que as empresas envolvidas arcam com os custos, mas, na verdade, essa perda operacional geralmente encarece preços para o consumidor final. Daí, combater fraudes nesse meio de pagamento traz benefícios para toda a sociedade. Essa tarefa, devido à sua complexidade e tamanho, tem que ser feita de uma forma concomitantemente eficaz e eficiente. Logo, há oportunidade para a aplicação de técnicas de Aprendizagem de Máquina, ou *Machine Learning* (OLIVEIRA, 2016), sendo este o principal propósito deste trabalho.

Visando manter níveis de resultados satisfatórios para a predição de fraudes, será utilizado neste projeto três modelos de aprendizagem de máquina. Segundo, Vieira (2018, p. 13):

> Com o aprendizado de máquina é possível analisar um grande volume de dados complexos e aprender com eles. Além disso, é possível criar algoritmos que possam detectar quando algo está fora do padrão com grande precisão e aumentando essa precisão cada vez mais com a prática.

Nesse trabalho, serão utilizados os algoritmos k-NN (*K-Nearest Neighbors*), SVM (*Support Vector Machines)* e RF (*Random Forest*), e a partir dos modelos gerados pelos algoritmos, mostrar o que melhor se adequa ao conjunto de dados selecionado para testes, utilizando como métricas de comparação das predições a acurácia e o F1-*Score*, como será justificado nos capítulos 3 e 4.

Com base nos resultados gerados, o trabalho busca apresentar à comunidade acadêmica que é possível alcançar bons resultados para esse tipo de problema, utilizando algoritmos de classificação e aprendizado supervisionado em *machine learning.*

1.2 Objetivo geral e específicos

É objetivo geral deste trabalho avaliar modelos preditivos de aprendizagem de máquina, utilizando os algoritmos k-NN, SVM e *Random Forest*, tendo como

base, um conjunto de dados contendo transações realizadas por cartões de crédito, bem como, comparar os resultados e identificar a abordagem mais adequada para predizer fraudes em transações realizadas a partir de cartões de crédito.

Adicionalmente, pretende-se explorar como objetivos secundários os seguintes aspectos:

- a realização da análise comparativa dos modelos apresentados.
- a identificação, com base nos resultados obtidos, do modelo que apresente melhor performance conforme métricas definidas, como alvo focado na predição de fraudes bancárias.

1.3 Justificativa

Mediante o exposto e devido aos inconvenientes provocados por fraudes em transações, este trabalho se justifica pela avaliação dos modelos preditivos previamente citados, utilizando a base de dados pós-processados fornecida por *Machine Learning Group - Université Libre de Bruxelas* na plataforma Kaggle (2018). O intuito é verificar quais dos algoritmos estudados se comporta melhor, como modelo de predição que possa ser usado para identificar potenciais fraudes.

1.4 Resultados Esperados

Espera-se, com a avaliação dos modelos preditivos implementados neste trabalho, gerar um conjunto de informações que seja suficiente para sobrevir estudos mais abrangentes acerca do modelo que melhor se comportou durante as análises.

1.5 Estrutura do Trabalho

Este trabalho está organizado como segue: o capítulo 2 apresenta a revisão bibliográfica, descrevendo os conceitos dos algoritmos k-NN, SVM e *Random Forest*. O capítulo 3 apresenta um panorama sobre os trabalhos relacionados. O capítulo 4 descreve o ambiente e a metodologia de desenvolvimento do trabalho, dispondo as

tecnologias utilizadas, descrição do conjunto de dados utilizado e como os métodos de predição são utilizados. No capítulo 5, os resultados são apresentados, juntamente com a discussão sobre os mesmos. O capítulo 6 contém a conclusão e a proposta de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Para melhor compreender as predições de fraudes em cartões de crédito, primeiramente é necessária a conceituação básica dos modelos preditivos utilizados, que são os algoritmos k-NN, SVM e RF. Além disso, é importante descrever um panorama apresentando os trabalhos relacionados para se expandir o entendimento dos procedimentos e resultados já produzidos na área.

De antemão, os três modelos preditivos abordados neste trabalho são algoritmos de aprendizagem supervisionada que utilizam-se da classificação como base operacional (SOUZA, 2019). Um requisito para manipular modelos de *Machine Learning* para detectar fraudes em transações de cartões de crédito é compreender em detalhes seus conceitos e técnicas, bem como suas aplicações.

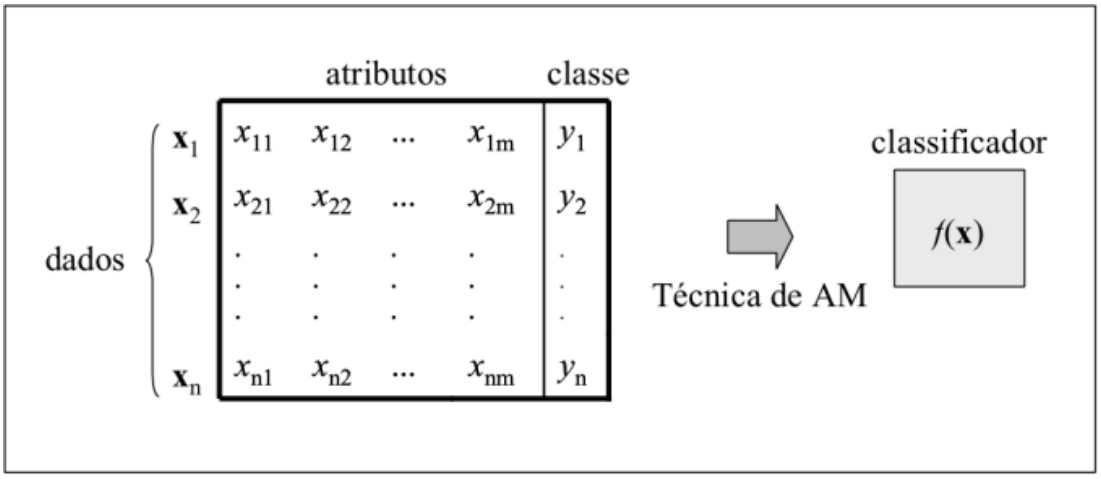
O primeiro conceito a ser descrito neste capítulo é o de *Machine Learning* e um breve esclarecimento sobre classificação e aprendizado supervisionado. O segundo conceito refere-se ao algoritmo de k-NN. Em terceiro será abordado o SVM e seu Hiperplano, e por último, descreve-se o preditor por *Random Forest* com árvores de decisão.

2.1 *Machine Learning*

Machine Learning ou Aprendizagem de Máquina pode ser definida como a área, através da qual se faz pesquisa, estudos e se define um conjunto de técnicas para, automaticamente, detectar padrões em dados e utilizar esses padrões descobertos para predizer acontecimentos futuros, ou até mesmo, para realizar outros tipos de tomada de decisão relacionados a eventos não determinísticos tipicamente, os problemas tratados pela área são: classificação, regressão e agrupamento (Clusterização) (OLIVEIRA, 2016).

Como citado anteriormente, todos os modelos de *Machine Learning* aplicados nesse trabalho são de Aprendizagem Supervisionada que pode ser descrita como um conjunto de treinamento contendo registros de exemplos com suas respectivas classes corretas e conhecidas, que é fornecido para o classificador. A partir daí, baseado nesses exemplos, o algoritmo generaliza as respostas recebidas e aplica essas generalizações a registos novos e de classes desconhecidas com a finalidade de determinar corretamente suas respectivas classes (OLIVEIRA, 2016).

Figura 1 - Indução de classificação em Aprendizado Supervisionado.



Fonte: OLIVEIRA JUNIOR, 2010.

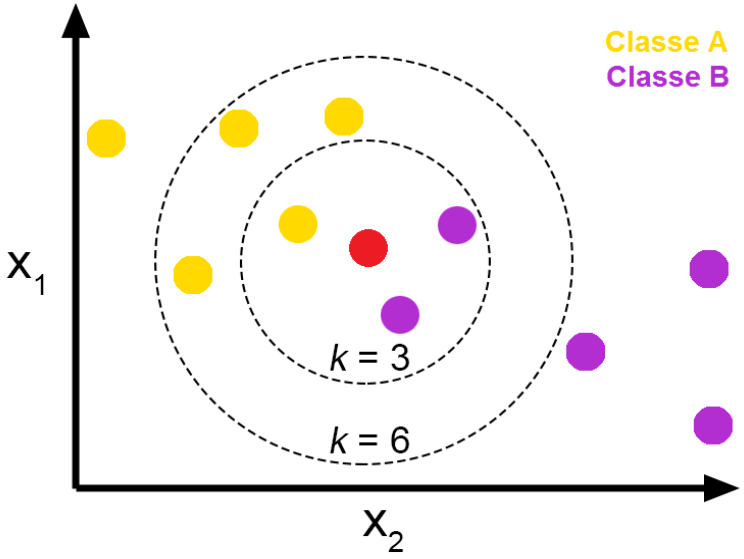
A Figura 1 mostra uma representação dos conceitos citados a respeito da geração do classificador através de um aprendizado supervisionado de forma simplificada. Os dados representam o conjunto de treinamento e x_i,..., x_n representam as instâncias. Cada instância x_i possui m atributos { x_{i1}, …, x_{im}}. Tais atributos podem ser categóricos ou contínuos e estão associados a uma classe y. A partir dos exemplos, é aplicado o algoritmo de aprendizagem de máquina para geração do classificador, que nada mais é do que uma função, no caso, chamada de f(x) (OLIVEIRA JUNIOR, 2010).

2.1.1 *K-Nearest Neighbors*

Considerado como uma das técnicas mais diretas de *Machine Learning* (CUNNINGHAM; DELANY, 2007), o algoritmo k-NN, ou modelo de vizinho mais próximo, baseia-se na vizinhança de uma amostra para classificá-la. De acordo com Russell e Norvig (2004, p. 711), “A ideia-chave de modelos de vizinho mais próximo é que as propriedades de qualquer ponto de entrada x específico têm probabilidade de serem semelhantes às propriedades de pontos na vizinhança de x”.

Existem diferentes meios para determinar a classe de um dado. Em geral, a abordagem mais comum é atribuir a classe majoritária entre os vizinhos mais próximos do ponto em questão (CUNNINGHAM; DELANY, 2007). Contudo, torna-se um desafio definir a regra de classificação, ou seja, o tamanho da vizinhança considerada para categorizar um ponto no k-NN. A figura 2 ilustra esse processo:

Figura 2- Exemplo da regra de classificação do k-NN.



Fonte: OLIVEIRA JUNIOR, 2010.

Na Figura 2, percebe-se que há um dado não classificado (em vermelho) e todos os outros dados já estão classificados (amarelo e roxo), cada um com sua classe (A e B, respectivamente). É calculada a distância do novo dado com todos os outros para saber quais estão mais próximos. No caso da Figura 2, os dados mais próximos do novo dado são aqueles que estão dentro do círculo. Em seguida, são verificados os 3 dados mais próximos. É possível verificar qual é a classe predominante ali dentro. Percebe-se que a classe B (roxo) se destaca. Logo, o novo dado que antes não estava classificado, agora será classificado como classe B.

Para determinar a distância entre os pontos, tem-se como métrica mais utilizada, a distância Euclidiana. Segundo Hu et. al. (2016), a distância euclidiana é dada por:

$$\mathrm{dist}(A,B) = \sqrt{\frac{\sum_{i=1}^{m}(x_i - y_j)^2}{m}} \qquad (1)$$

Onde A e B são representados por vetores de características A = ($x1$, $x2$, … , xm) e B = ($x1$, $x2$, … , xm), m é a dimensionalidade do espaço de características. Assim sendo, pode-se aplicar o k-NN à classificação de fraudes.

Outro algoritmo importante e interessante a ser apresentado é o *Support Vector Machines* (SVM), que será explicado a seguir.

2.1.2 *Support Vector Machines*

Support Vector Machines ou Máquinas de Vetores de Suporte foram desenvolvidas por Vapnik e colaboradores (GUNN, 1998). Tais máquinas apresentam a capacidade de resolver problemas de classificação e regressão, adquirindo com o aprendizado na etapa de treinamento, a capacidade de generalização. Como exemplo, considere um problema binário. O objetivo da SVM é separar as instâncias das duas classes binárias através de uma função que será obtida a partir dos exemplos conhecidos na fase de treinamento. O alvo do algoritmo consiste em produzir um classificador que funcione de forma adequada com exemplos não conhecidos, ou seja, exemplos que não foram aplicados durante o treinamento, adquirindo assim, a capacidade de predizer as saídas de futuras novas entradas (OLIVEIRA JUNIOR, 2010). A fim de se alcançar um entendimento mais abrangente de como funciona o algoritmo SVM, se torna indispensável descrever os conceitos de funções kernel.

As funções de kernel têm o objetivo de projetar vetores com características de entrada de um espaço de alta dimensão para classificação de problemas que se encontram em espaços não linearmente separáveis. Por esse motivo, à medida que se aumenta o espaço da dimensão do problema, consequentemente, também é aumentada a probabilidade que esse problema se torne linearmente separável em relação a um espaço de baixa dimensão. Contudo, para obter uma melhor distribuição para esse tipo de problema é preciso um conjunto de treinamento com um elevado números de instâncias (OLIVEIRA JUNIOR, 2010).

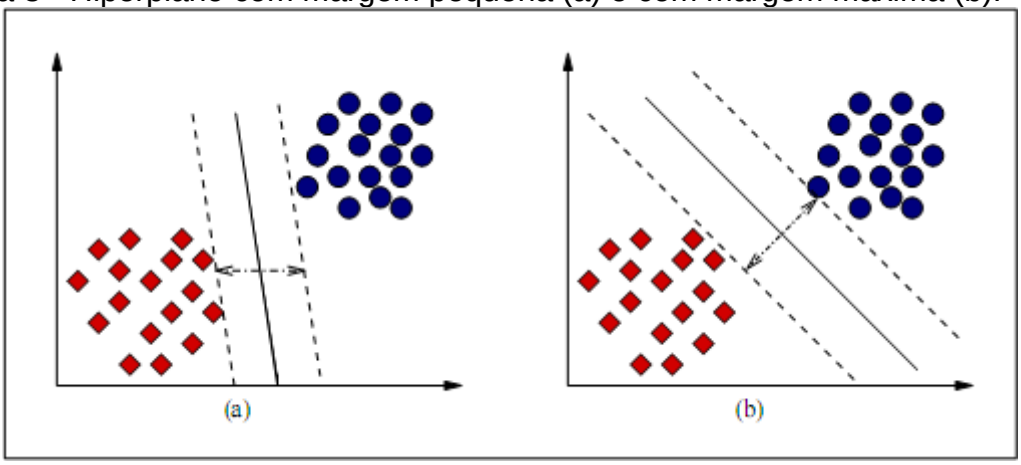
O SVM é um algoritmo que realiza um mapeamento dos dados de um espaço dimensional onde se tem um hiperplano de separação ótima, que funciona como uma divisa para decisão, comumente chamado de classificação (LORENA e CARVALHO, 2007). Apto a atuar com dados lineares, quanto dados não lineares, caracteriza-se como um dos modelos mais robustos dentro da aprendizagem de máquina.

A seguir, será explicado um exemplo de como um hiperplano pode ser utilizado conforme o SVM sugere.

2.1.2.1 Hiperplano

Uma SVM constrói um classificador de acordo com um conjunto de padrões por ele identificados nos exemplos de treinamento, onde a classificação é conhecida.

Figura 3 - Hiperplano com margem pequena (a) e com margem máxima (b).



Fonte: OLIVEIRA JUNIOR, 2010.

Considerando a Figura 3, percebe-se que existe um conjunto de classificadores lineares que separam duas classes, mas apenas um (em destaque) que potencializa a margem de separação (distância da instância mais próxima ao hiperplano de separação das duas classes em questão). O hiperplano com margem máxima é chamado de hiperplano ótimo, que será o objeto de busca do treinamento do classificador (GUNN, 1998).

Outro algoritmo a ser apresentado é o de *Random Forest*, que será descrito a seguir.

2.1.3 *Random Forest*

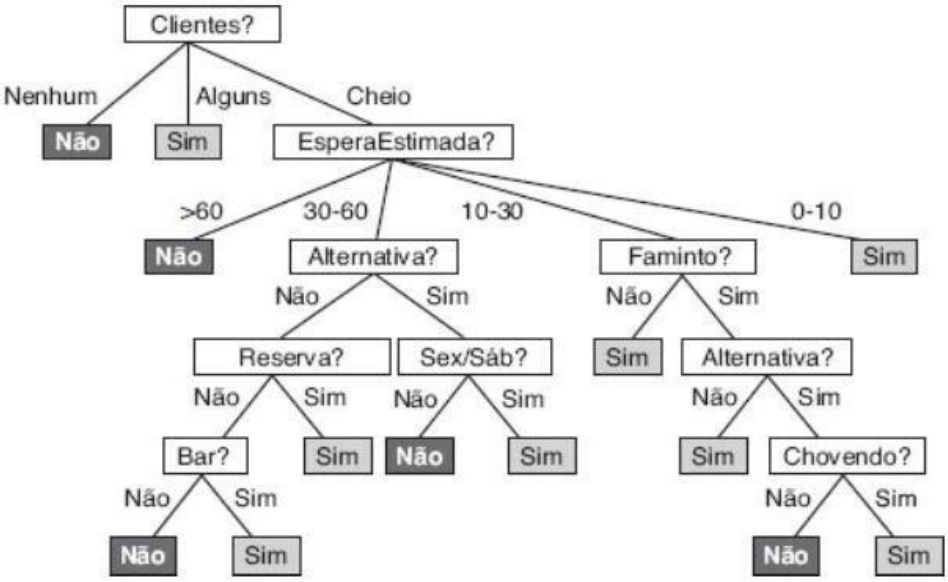
As *Random Forest* ou Florestas Aleatórias são compostas por Árvores de Decisão, que são geradas através de vetores aleatórios. As florestas aleatórias utilizam o método de votação para classificar determinada amostra. De acordo com Breiman (2001):

> Uma Floresta Aleatória é um classificador que consiste em uma coleção de árvores de decisão estruturadas $\{h(x, \Theta k), k = 1, \ldots\}$, onde $\{\Theta k\}$ são vetores aleatórios independentes distribuídos de forma idêntica e cada árvore emite um voto unitário para a classe mais popular na entrada x.

Para melhor entender o conceito de florestas aleatórias, é importante uma breve apresentação sobre as árvores de decisão. De acordo com a descrição de Russell e Norvig (2004, p. 633), “uma árvore de decisão toma como entrada um objeto ou situação descritos por um conjunto de atributos e retorna uma ‘decisão’ – o valor de saída previsto, de acordo com a entrada”.

Diante disso, para chegar a uma decisão, testes subsequentes são realizados de tal forma que, cada nó interno representa um teste baseado em atributos com uma ramificação para cada resultado possível. As folhas de uma árvore de decisão são as possíveis classes que, em outras palavras, são as decisões (QUINLAN, 1986).

Figura 4 - Representação de uma árvore de decisão.



Fonte: RUSSELL; NORVIG, 2004.

A Figura 4 demonstra um exemplo de árvore de decisão referente à escolha entre esperar ou não uma mesa em um restaurante. Nessa representação, atributos são fatores que influenciam na decisão de esperar por uma mesa como, por exemplo, preço, condição climática, espera estimada, entre outros. As folhas da árvore de decisão correspondem a duas possíveis decisões: sim, que corresponde a esperar, e não, que corresponde a não esperar.

Em linhas gerais, todas as árvores de decisão da floresta recebem uma mesma amostra e a classificam. Com isso, acontece uma votação e a classe majoritária entre as árvores de decisão corresponde à decisão da floresta (SOUZA, 2019).

Após elucidar os principais algoritmos utilizados para estudos nessa área, há de se ressaltar que existem diversas formas metodológicas de se prever fraudes. Dos vários estudos realizados pela comunidade científica, foram selecionados dois trabalhos que forneceram base técnica e teórica para elaboração deste trabalho. Enfatiza-se que as escolhas das metodologias empregadas, bem como das formas técnicas de se gerar as predições utilizadas aqui, foram baseadas nos estudos e resultados apresentados nesses trabalhos, os quais são descritos no Capítulo 3.

3 TRABALHOS RELACIONADOS

O aumento considerável no número de fraudes em cartões de crédito nos últimos anos alavancou diversos estudos envolvendo as predições com IA. Visando a detecção dessas fraudes, Vieira (2018), avaliou técnicas de aprendizado de máquina para predição de fraudes. O trabalho realizado utiliza os algoritmos de classificação SVM, k-NN e *Naive Bayes*, empregando como métricas, a acurácia, precisão, revocação e F1-*Score*. Tais métricas serão detalhadas no Capítulo 4.

Optando por não usar nenhum meio de normalização, Vieira não realizou o balanceamento dos dados, realizando seus experimentos com a base de dados completa e reduzindo-se a quantidade de registros e de colunas. Para reduzir o número de colunas, foi utilizado o algoritmo PCA, que realiza a otimização de variáveis, diminuindo 30 colunas em apenas 5. Com a finalidade de enriquecer as variáveis de forma com que os algoritmos tenham boa performance (VIEIRA, 2018). A técnica PCA estará melhor definida no subtópico 4.2.1.1.

Vieira chegou à conclusão de que o *Naive Bayes* foi o algoritmo que apresentou os melhores resultados em todos os experimentos, atingindo uma assertividade de 63,33% com a base completa. Já o k-NN foi o algoritmo que apresentou o pior desempenho, e o SVM teve um desempenho pouco melhor que o k-NN.

Insatisfeito com o mal comportamento do algoritmo SVM, Vieira deixa como sugestão de trabalhos futuros, a realização de novas pesquisas, aplicando o modelo em outros testes, sugerindo como exemplo, a técnica de kernel do algoritmo SVM, e a aplicação de um pré-processamento na base de dados usada por ele para balancear as classes. A partir disso, se justifica o emprego do mesmo conjunto de dados utilizados por ele e o modelo SVM, nesse trabalho.

Em outra linha de pesquisa, Nicola, Lauretto e Delgado (2020) realizaram uma avaliação empírica de classificadores e métodos de balanceamento para detecção de fraudes em transações com cartões de crédito. Os pesquisadores recorreram aos algoritmos k-NN, *Naive Bayes*, *Randon Forest*, Regressão Logística e Máquinas de Vetores de Suporte (SVM), e empregaram como métrica, apenas o F1-*Score*.

O trabalho de Nicola, Lauretto e Delgado (2020), também utilizou os dados do *ULB Machine Learning Group*, utilizado nesse trabalho e apresentou os testes com a base desbalanceada (DB), mas também descreveu várias técnicas de pré-

processamento, como a técnica de subamostragem (RU), subamostragem com cluster (PK), sobreamostragem (RO), sobreamostragem de minoria sintética (SMOTE ou SM) e SMOTE combinado com ligações de TOMEK (ST). A subamostragem por ligação de TOMEK consiste em identificar instâncias que possuem ligações e eliminar uma das duas observações, repetindo o processo até atingir o balanceamento desejado. Nicola, Lauretto e Delgado fizeram o uso das 5 formas de normalização, pois foram essas as técnicas que tiveram melhor desempenho nos trabalhos usados como referência estudados por eles. Duas dessas técnicas foram exploradas neste trabalho.

Foi utilizada por eles para o experimento do modelo, a base de dados completa com 30 colunas e por eliminação recursiva, optou-se por deixar somente 9 colunas que o algoritmo escolhe por conter características de maior relevância.

Com a aplicação dos algoritmos e todas as técnicas os pesquisadores obtiveram os seguintes resultados:

Figura 5 - Tabela dos resultados obtidos no trabalho de Nicola, Lauretto e Delgado (2020).

Técnicas de balanceamento	k-NN		Naïve Bayes	
	Completo	Reduzido	Completo	Reduzido
Desbalanceado (DB)	0,17 (0,012)	**0,83 (0,011)**	0,24 (0,010)	0,20 (0,005)
Random undersampling (RU)	0,01 (0,000)	0,15 (0,008)	0,21 (0,010)	0,16 (0,013)
Protótipos com kmeans (PK)	0,01 (0,001)	0,01 (0,000)	**0,37 (0,013)**	0,09 (0,008)
Random oversampling (RO)	0,30 (0,022)	0,83 (0,012)	0,21 (0,007)	0,15 (0,003)
SMOTE (SM)	0,08 (0,002)	0,72 (0,012)	0,25 (0,007)	0,16 (0,004)
SMOTE + Tomek (ST)	0,08 (0,003)	0,72 (0,012)	0,25 (0,007)	0,16 (0,004)

Técnicas de balanceamento	Random Forest		Regressão Logística	
	Completo	Reduzido	Completo	Reduzido
Desbalanceado (DB)	**0,87 (0,011)**	0,86 (0,008)	0,00 (0,000)	**0,71 (0,015)**
Random undersampling (RU)	0,12 (0,008)	0,08 (0,006)	0,01 (0,000)	0,12 (0,008)
Protótipos com kmeans (PK)	0,00 (0,000)	0,01 (0,000)	0,00 (0,000)	0,02 (0,000)
Random oversampling (RO)	0,87 (0,010)	0,85 (0,010)	0,13 (0,004)	0,12 (0,002)
SMOTE (SM)	0,87 (0,010)	0,72 (0,009)	0,23 (0,009)	0,17 (0,003)
SMOTE + Tomek (ST)	0,86 (0,009)	0,70 (0,010)	0,23 (0,009)	0,17 (0,003)

Técnicas de balanceamento	SVM	
	Completo	Reduzido
Desbalanceado (DB)	0,14 (0,030)	**0,71 (0,052)**
Random undersampling (RU)	0,05 (0,040)	0,14 (0,013)
Protótipos com kmeans (PK)	0,15 (0,043)	0,01 (0,000)
Random oversampling (RO)	0,12 (0,041)	0,14 (0,008)
SMOTE (SM)	0,15 (0,052)	0,18 (0,007)
SMOTE + Tomek (ST)	0,16 (0,037)	0,16 (0,004)

Fonte: Nicola, Lauretto e Delgado (2020).

Conforme exposto na Figura 5, observa-se que o algoritmo k-NN obteve seus melhores desempenhos com a base de dados desbalanceada (**0,83**). *Naive Bayes* teve seu melhor resultado na configuração de subamostragem com cluster (**0,37**), o *Random Forest* obteve o melhor desempenho também com os dados desbalanceados e com o conjunto completo (**0,87**). Já os algoritmos de Regressão Logística e SVM obtiveram comportamento bastante similares. As duas tiveram seu melhor resultado com os dados desbalanceados e com o conjunto reduzido (**0,71**).

Logo, chegou-se à conclusão de que *Random Forest* foi o classificador que atingiu o maior F1-*Score*, e se apresentou muito consistente ao atingir o patamar de **0,87** em conjuntos desbalanceados e com sobreamostragem. Nenhum dos outros algoritmos atingiu desempenho estatisticamente equivalentes ao F1-*Score* ótimo. O único algoritmo que teve o F1-*Score* médio semelhante foi o k-NN.

Figura 6 - Tabela demonstrando os resultados F1-*Score* Ótimos.

Balanceamento	k-NN (Reduzido)	Naïve Bayes (Completo)	Random Forest (Completo)	Regressao Logística (Reduzido)	SVM (Reduzido)
	Média (d. p.)	Média (d. p.)	Média (d. p.)	Média (d. p.)	Média (d. p.)
DB	0,83 (0,011)	0,24 (0,010)	**0,87 (0,011)**	0,71 (0,015)	0,71 (0,052)
RU	0,15 (0,008)	0,21 (0,010)	0,12 (0,008)	0,12 (0,008)	0,14 (0,013)
PK	0,01 (0,000)	0,37 (0,013)	0,00 (0,000)	0,02 (0,000)	0,01 (0,000)
RO	0,83 (0,012)	0,21 (0,007)	0,87 (0,010)	0,12 (0,002)	0,14 (0,008)
SM	0,72 (0,012)	0,25 (0,007)	0,87 (0,010)	0,17 (0,003)	0,18 (0,007)
ST	0,72 (0,012)	0,25 (0,007)	0,86 (0,009)	0,17 (0,003)	0,16 (0,004)

Fonte: Nicola, Lauretto e Delgado (2020).

Como pode ser observado na Figura 6, a Regressão Logística e SVM apresentaram desempenhos ótimos similares, entretanto inferior à de *Random Forest* e do k-NN. Em último lugar, fica o algoritmo *Naive Bayes* que teve seu desempenho muito inferior aos demais, atingindo um F1-*Score* de **0,37**, com balanceamento utilizando subamostragem com cluster.

Nicola, Lauretto e Delgado (2020), observaram em seu trabalho que ao treinar um conjunto de dados balanceados e testar conjuntos desbalanceados, nota-se uma tendência de degradação no desempenho em quase todos os casos. Levando isso em consideração, eles deixam como realização em trabalhos futuros, um estudo mais abrangente a respeito de técnicas de balanceamento. Baseado no trabalho de Nicola, Lauretto e Delgado (2020) e visando ter resultados satisfatórios nesse trabalho, serão utilizados os algoritmos que obtiveram os melhores desempenhos.

Assim sendo, as técnicas de normalização e pré-processamentos serão discutidas no próximo capítulo, onde serão tratados o ambiente de produção e a metodologia para o desenvolvimento dos experimentos.

4 AMBIENTE DE PRODUÇÃO E METODOLOGIA DE DESENVOLVIMENTO

Neste capítulo, primeiramente é abordado o ambiente de desenvolvimento dos experimentos. Com isso, descreve-se a aplicação web *Google Colab*, a linguagem de programação *Python* e as bibliotecas utilizadas nos algoritmos selecionados para predições de fraudes em transações. Encadeando-se à descrição do ambiente de desenvolvimento, a metodologia apresenta o conjunto de dados utilizado e os métodos de normalização aplicados. Por fim, as etapas de aprendizado de máquina são descritas, desde a seleção de hiperparâmetros até os testes.

4.1 Ambiente

Para a implementação e avaliação das predições de fraudes em transações de cartões de crédito, neste trabalho utiliza-se a aplicação web *Google Colab* como ambiente de desenvolvimento. O *Google Colab* permite escrever código *Python* no navegador WEB, com nenhuma configuração ou instalação necessária, acesso gratuito a GPUs de alto processamento e compartilhamento fácil. Os notebooks gerados pelo *Google Colab* permitem combinar código executável e texto em um só documento, além de imagens, HTML, e muitas outras aplicações (COLAB, 2020). E a linguagem *Python* foi definida como padrão.

4.1.1 *Python*

Para a realização dos experimentos, os modelos foram escritos e testados utilizando a linguagem *Python* na versão 3.9.2. *Python* é poderoso, rápido, é uma linguagem multipropósito de fácil entendimento, com código aberto, e hospeda milhares de módulos de terceiros. Tanto as bibliotecas padrões do *Python* quanto os módulos contribuídos pela comunidade permitem infinitas possibilidades para seu uso (PYTHON, 2020).

4.1.2 Bibliotecas

A implementação dos experimentos deste trabalho realizou-se com o uso de três bibliotecas da linguagem de programação *Python*. São elas: *Pandas*, *Scikit-learn* e *Seaborn*. A seguir, é apresentado características dessas bibliotecas:

- ***Pandas***: é uma ferramenta de análise e manipulação de dados de código aberto rápida, poderosa, flexível e fácil de usar, construída sobre a linguagem de programação *Python* (PANDAS, 2020). A *Pandas* é um componente essencial deste trabalho, porque é através dela que os *datasets* são importados para a aplicação *Google Colab Notebook*.

- ***Scikit-learn***: A *Scikit-learn* é composta por diversas ferramentas para *Machine Learning* em *Python*, as quais são divididas em seis categorias principais: métodos para classificação, regressão, clusterização, redução de dimensionalidade, seleção de modelos e pré-processamento de dados (SCIKIT-LEARN, 2019). Esta pesquisa utiliza três ferramentas *Scikit-learn*, as quais fazem parte da categoria de métodos para classificação. A primeira é a *KNeighborsClassifier*, correspondente ao modelo k-NN. A segunda é a classe SVC, que como o nome sugere, utiliza o classificador SVM. A terceira classe, *RandomForestClassifier*, atua como o classificador de Florestas Aleatórias. Além disso, classes complementares, como a seleção de hiperparâmetros (*RandomSearchCV*) e métricas de avaliação (*metrics*), são utilizadas.

- ***Seaborn***: é uma biblioteca de visualização de dados *Python* baseada em *matplotlib* . Ela fornece uma interface de alto nível para desenhar gráficos estatísticos atraentes e informativos. A biblioteca auxilia no entendimento dos dados. Com funções de plotagem operam em *dataframes* e matrizes contendo conjunto de dados inteiros, realizando o mapeamento semântico e agregação estatística necessários para produzir gráficos (SEABORN, 2021).

Tendo apresentado o ambiente de desenvolvimento, a linguagem *Python* e suas bibliotecas, a próxima seção descreve a metodologia deste trabalho.

4.2 Metodologia

Este tópico apresenta a metodologia utilizada neste trabalho. Abordam-se aqui, características a respeito do conjunto de dados e os métodos de balanceamento a ele aplicados. Em sequência, descreve-se as etapas para predição de fraudes em transações de cartões de crédito, demonstrando a técnica para otimizar as predições, as métricas de avaliação utilizadas e também os estágios da aprendizagem de máquina (treinamento, teste e predição).

4.2.1 Conjunto de dados

O *dataset* escolhido para este trabalho foi disponibilizado por *Machine Learning Group - Université Libre* de Bruxelas na plataforma *Kaggle* (2018). Os conjuntos de dados contêm transações feitas por cartões de crédito em setembro de 2013 por titulares de cartões europeus. Este conjunto de dados apresenta as transações que ocorreram em dois dias, onde temos 492 fraudes em 284.807. Possui 31 atributos (colunas) numéricos, *Time*, *Amount*, *Class*, que são, respectivamente, o tempo que foram realizadas, o valoras das transações e a classe, e mais 28 atributos (V1, V2, V3, … V28) genéricos, derivados de variáveis originais por meio da Análise de Componentes Principais (*PCA*). Por se tratar de dados pessoais, essas variáveis originais foram omitidas. A escolha dessa base foi definida por conter registros de transações de cartão de crédito reais e, portanto, será usada para treinamento dos modelos e análise da performance de cada algoritmo. A seguir, será descrito como funciona o método PCA.

4.2.1.1 Análise de Componentes Principais (PCA)

O PCA é o método de redução de dimensionalidade, sendo constantemente utilizado para reduzir a dimensionalidade de grandes conjuntos de dados, decompondo um grande conjunto de variáveis em um conjunto menor que ainda há a maioria das informações do conjunto original. Conjunto de dados menores são mais fáceis de realizar a análise dos dados, a exploração e visualização, também se torna

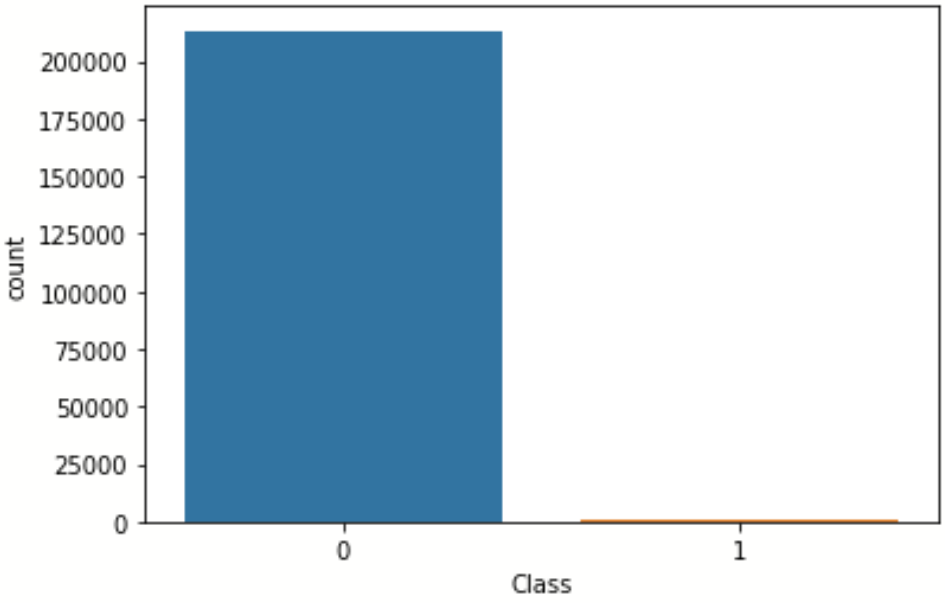
mais rápida para algoritmos de *Machine Learning*, sem a necessidade de que grandes variáveis sejam processadas (JAADI, 2017).

Com a utilização do método PCA neste trabalho, pretende-se obter valores próximos das variáveis originais, sem perder seu valor analítico, já que no conjunto de dados em sua maioria são variáveis genéricas, provenientes da aplicação da técnica.

4.2.2 Balanceamento de dados

Por consequência da desproporção entre os dados, visto que a classe onde há fraudes representa somente 0,172% dos registros do conjunto, é necessário o balanceamento dos dados para obter o maior desempenho nas predições. Cinco técnicas de balanceamento são exploradas em Nicola, Lauretto e Delgado (2020). A partir da pesquisa selecionada, serão utilizadas somente duas técnicas descritas, uma vez que, como observado no Capítulo 3, não se obteve resultados satisfatórios em mais de duas combinações de balanceamento.

Figura 7 - Gráfico mostrando o desbalanceamento das classes.

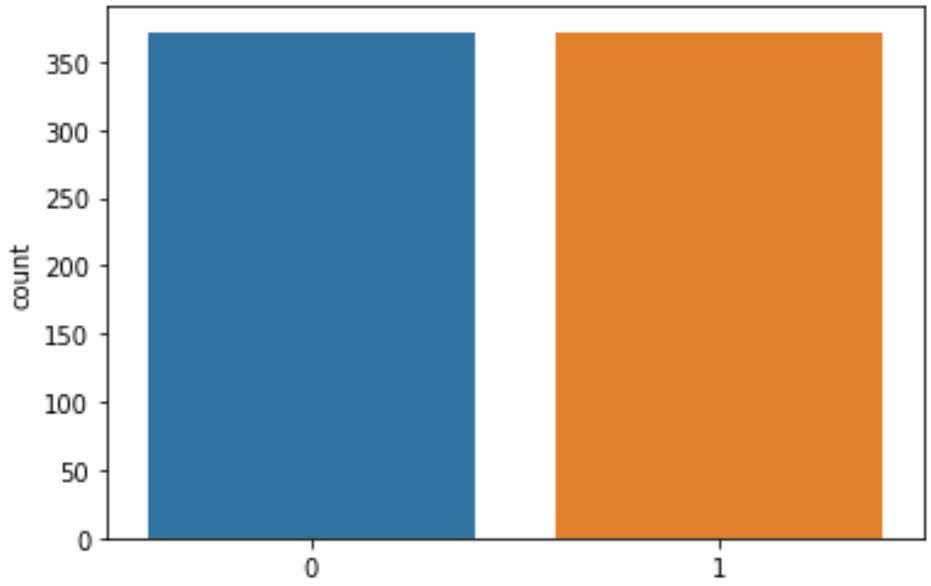


Fonte: Elaborado pelo autor.

O gráfico exibido na Figura 7 apresenta uma melhor noção de quanto o conjunto de dados é altamente desbalanceado. A classe 0, que corresponde às transações legítimas, consiste em 284.315 registros, enquanto a classe 1 que correspondente às transações fraudulentas, representa somente 492 registros.

Inicialmente, a primeira técnica utilizada é a de Subamostragem, ou *Random Undersampling* (RU), que consiste em minimizar tuplas da classe majoritária até que se atinja o nível da classe minoritária, obtendo o balanceamento desejado (BARELLA, 2015). A Figura 8 apresenta um melhor entendimento da técnica. Nela percebe-se que as variáveis não fraudulentas são altamente reduzidas para se ter um equilíbrio entre as duas classes.

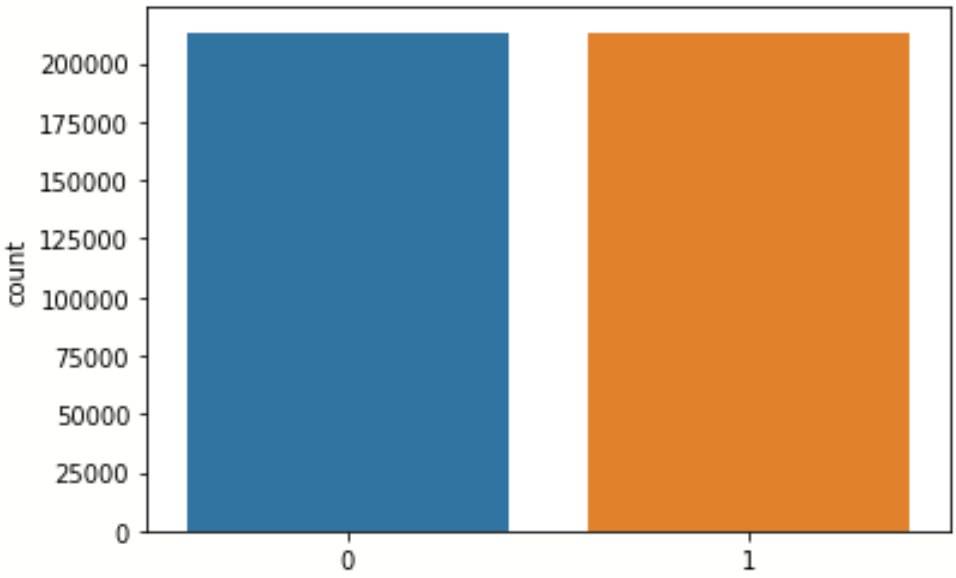
Figura 8 - Gráfico das classes após a aplicação da técnica *Undersampling*.



Fonte: Elaborado pelo autor.

A segunda é a Técnica de Sobreamostragem Minoritária Sintética, ou *Synthetic Minority Oversampling Technique* (SMOTE). Contrária à técnica RU, a SMOTE constitui novas tuplas. Seu funcionamento é parecido ao do algoritmo k-NN. A cada iteração, uma nova amostra da classe minoritária é selecionada aleatoriamente e os k-vizinhos mais próximos são encontrados. Um dos vizinhos é selecionado e, logo, forma-se uma nova amostra que combina as duas tuplas (CHAWLA et al., 2002).

Figura 9 – Gráfico das classes após a aplicação da técnica SMOTE.



Fonte: Elaborado pelo autor.

A Figura 9 demonstra o resultado do balanceamento usando a técnica SMOTE. Observa-se que, enquanto a abordagem de Subamostragem reduz a quantidade de transações sem fraudes, a SMOTE amplia a parcela de transações com fraudes. Após obter-se os novos *datasets* com as normalizações, pode ser aplicado a etapa de aprendizado de máquina.

Para adquirir o máximo desempenho de um algoritmo de aprendizado de máquina, é de exímia importância buscar alcançar os melhores hiperparâmetros (SOUZA, 2019), o tópico 4.2.3 a seguir descreve essa busca.

4.2.3 Busca por hiperparâmetros

Para a seleção dos hiperparâmetros de cada algoritmo, é adotado o uso da função *RandomizedSearchCV* da biblioteca *Scikit-learn* e o *RandonSearch* também conhecido por pesquisa aleatória. O *RandomSearch* realiza a busca por um método de ajuste e um método de pontuação, que aqui será a acurácia. Os parâmetros do estimador usados para aplicar esses métodos são otimizados por pesquisa de validação cruzada sobre configurações dos parâmetros. Nem todos os valores de parâmetros são testados, mas, um número fixo de configurações de parâmetro é amostrado a partir das distribuições especificadas. Se todos os parâmetros são apresentados como uma lista, a amostragem sem substituição é realizada. Se pelo

menos um parâmetro é fornecido como uma distribuição, a amostragem com substituição é usada (SCIKIT-LEARN, 2021).

Neste trabalho, os hiperparâmetros avaliados são: o número de vizinhos considerados pelo k-NN (*n_neighbors*), qual a melhor função kernel para SVM (*kernel*), e o total de árvores de decisão no *Random Forest* (*n_estimators*). Em virtude de limitações acerca do tempo e processamento computacional, neste trabalho, para estimar os demais hiperparâmetros de cada algoritmo, utiliza-se dos parâmetros padrões da biblioteca *Scikit-learn*, que serão melhor elucidados no tópico 5.1.

A seguir, a apresentação de como será realizada a seleção de atributos.

4.2.4 Seleção de atributos

Inicialmente foram realizados experimentos com todos os 30 atributos para descrever uma transação, mas, levando em consideração todo o tempo para a seleção dos hiperparâmetros, como descrito no tópico 5.1**,** decidiu-se que para a seleção da quantidade de atributos (colunas), tomou-se como base o trabalho de Nicola, Lauretto e Delgado (2020), que utilizando da regressão logística com validação cruzada, chegaram em 30% de atributos (9 colunas). Entretanto, esse trabalho busca resultados compatíveis e não similares, optou-se por fazer duas pesquisas. Portanto, os experimentos serão realizados com uma parte do *dataset* por vez: a princípio serão trabalhadas as 18 primeiras colunas (60%), e posteriormente as outras 12 colunas (40%), visando manter a manipulação de todo o conjunto de dados, para assim ter um resultado mais satisfatório.

4.2.5 Métricas de avaliação

A fim de calcular a performance de um modelo preditivo, torna-se essencial determinar uma ou mais métricas de avaliação. Na avaliação dos modelos preditivos deste trabalho, estão estabelecidos como métricas a acurácia e a média ponderada entre precisão e revocação, conhecida como F1-*Score*. Para ter uma melhor compreensão das métricas de avaliação e suas características, primeiro é interessante mostrar o conceito de matrizes de confusão.

Para descrição de uma matriz de confusão, existem duas classes: **SIM e NÃO**. Tais classes podem representar se há ou não fraudes em transações. Então, a saída prevista para uma amostra pode atribuir-se um entre quatro possíveis valores.

- **Verdadeiro Positivo (VP):** amostras positivas classificadas corretamente. Casos em que retornamos a classe com fraude e a transação realmente era fraudulenta.

- **Verdadeiro Negativo (VN):** amostras negativas classificadas corretamente. Casos em que retornamos a classe sem fraude e que realmente eram transações legítimas.

- **Falso Positivo (FP):** amostras classificadas erroneamente como sim, quando o correto seria não. Casos em que retornamos a classe fraudulenta e na verdade eram transações não fraudulentas.

- **Falso Negativo (FN):** amostras classificadas erroneamente como não, quando o correto seria sim. Caso em que retornamos que não eram fraudulentas e na verdade eram fraudulentas.

Isto posto, pode-se descrever melhor as métricas de avaliação. A acurácia determina o quão próximo o valor real está da saída do classificador (SHALEVSHWARTZ; BEN-DAVID, 2014). Em outras palavras, adiciona-se o número de verdadeiros positivos ao número de verdadeiros negativos (VP + VN) e, posteriormente, divide-se pelo total de amostras classificadas (VP + VN + FP + FN). A fórmula da acurácia é dada por (WITTEM, FRANK E HALL, 2011):

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \quad (2)$$

De outra forma, a métrica F1-*Score* faz uma média ponderada entre precisão e revocação. A precisão pode ser considerada como a capacidade do classificador de não classificar como positiva uma amostra negativa (SCIKIT-LEARN, 2019). Sua fórmula é dada por:

$$Precisão = \frac{VP}{VP + FP} \quad (3)$$

Resumindo, a quantia de verdadeiros positivos (VP) é dividida pela quantia de verdadeiros positivos adicionados aos falsos positivos (VP + FP). Enquanto, a revocação é a capacidade de o classificador descobrir todas as amostras positivas (SCIKIT-LEARN, 2021). A fórmula é dada por:

$$Revocação = \frac{VP}{VP + FN} \quad (4)$$

Em que o resultado de verdadeiros positivos (VP) é dividido pelos verdadeiros positivos somados aos falsos negativos (VP + FN). Levando em consideração as explicações anteriores, podemos apresentar a fórmula do F1-*Score* dada por:

$$F1 - Score = \frac{2\ x\ (precisão\ x\ revocação)}{precisão + revocação} \quad (5)$$

Conforme ocorrem os cálculos, os valores oriundos da acurácia e F1-*Score* variam entre 0 pior valor, e 1 para o melhor valor.

Com as métricas de avaliação definidas, é possível encontrar os melhores hiperparâmetros e, posteriormente, empregá-los para o treinamento dos modelos preditivos (SOUZA, 2019). É bom deixar claro que os modelos preditivos são trabalhados de forma individual e independente, isto é, não há um seguimento de uso entre eles. Isso acontece com a finalidade de avaliar e comparar os resultados de cada um deles.

A seguir são apresentados os conceitos de treinamento, teste e predição, demonstrando exemplos de como esse passo é importante para nosso experimento.

4.2.6 Treinamento, teste e predição

Os algoritmos aplicados para predições de fraudes em transações de cartão de crédito (k-NN, SVM, *Random Forest*), possui semelhança em sua entrada e saída de dados quando utilizados a partir da biblioteca *Scikit-learn*.

A princípio, a fase de treinamento acontece através de duas matrizes, X e Y. A matriz X, de tamanho (número de variáveis x números de atributos), inclui as amostras de treinamento representadas como vetores de números reais. Por sua vez, a matriz Y, de tamanho (número de variáveis x 1), contém os valores esperados para a saída

de cada amostra de treino. As matrizes para teste possuem as mesmas características das matrizes de treinamento (SCIKIT-LEARN, 2020).

Para que uma predição seja realizada, é necessário instanciar um classificador. Na Figura 10 é demonstrado um exemplo utilizando o preditor *RandomForestClassifier*. Pode ser observado que a variável **pred** recebe uma instância do classificador.

Figura 10 - Exemplo do uso do *RandomForestClassifier*.

```
from sklearn.ensemble import RandomForestClassifier
pred = RandomForestClassifier()
pred.fit(X_treino, Y_treino)
pred.predict(X_teste)
```

Fonte: Elaborado pelo autor.

O método **fit(X, Y)** é o responsável por realizar o treinamento. Após executar o treino, o modelo está pronto para realizar predições na fase de teste. A fase de teste apresenta ao modelo preditivo novas amostras, para avaliar a efetividade das predições (SHALEV-SHWARTZ; BENDAVID, 2014).

A Figura 10 também apresenta uma chamada ao método **predict**. O método **predict(X)** é usado para realizar predições nos classificadores. Para ocorrer essas predições, apenas a matriz **X_teste**, que contém as amostras, precisa ser transmitida ao método de predição. Diante a execução, o método retorna uma matriz com os valores preditos. No sentido de verificar a performance, compara-se os valores de saída com os valores esperados aplicando as métricas de avaliação (SOUZA, 2019).

Logo, as etapas de aprendizado de máquina expostas neste tópico são executadas para os classificadores k-NN, SVM e *Random Forest*. O experimento faz a utilização de cada um dos modelos preditivos para a realização das predições e testes com cada um dos *datasets* de seleção de atributos e balanceamento de dados. O próximo capítulo exibe os resultados adquiridos.

5 RESULTADOS

Para a realização dos experimentos utilizou-se um computador com processador AMD Ryzen 7 1700, 16GB de memória RAM e Sistema Operacional MS-Windows® 10. Seguindo as etapas apresentadas no Capítulo 4, inicialmente houve a avaliação dos hiperparâmetros com a função *RandomizedSearchCV*, da biblioteca *Scikit-learn*. Em sequência, os modelos preditivos de k-NN (*KNeighborsClassifier*), SVM (*SVC*) e *Random Forest* (*RandomForestClassifier*) foram treinados com os melhores hiperparâmetros encontrados e posteriormente, foi executado a fase de testes. Todos os passos abordados serão apresentados a seguir.

5.1 Seleção de hiperparâmetros

Os hiperparâmetros referentes à quantidade do número de vizinhos (*n_neighbors*) para k-NN, o melhor kernel (*Kernel*) para SVM e quantidade de árvores (*n_estimators*) na *Random Forest* foram avaliados através da pesquisa aleatória, como foi esclarecido no tópico 4.2.3. A métrica de avaliação escolhida é a acurácia.

Nos experimentos realizados neste trabalho, o tempo médio de execução para cada seleção de um hiperparâmetros é bastante alto, dependendo do balanceamento empregado, houve casos como do SVM utilizando da normalização SMOTE, levar 43 horas para ser executado. Em razão ao tempo e por se tratar de uma busca exaustiva, optou-se por utilizar dos valores padrões da biblioteca *Scikit-learn*, para os demais parâmetros de cada algoritmo. Os próximos três tópicos apresentam os resultados de cada busca.

5.1.1 Número de vizinhos (*n_neighbors*)

Para buscar o melhor valor para o hiperparâmetro *n_neighbors*, o espaço de busca foi realizado no intervalo de 1 a 30, como visto no trabalho de Nicola, Lauretto e Delgado (2020), valores acima de 30 vizinhos, não se obtém novos resultados. Vale ressaltar que cada execução da função *RandomizedSearchCV,* realiza 5 execuções para cada valor k de teste.

A seleção dos hiperparâmetros foi dividida em duas partes: a primeira fazendo uso do balanceamento RU e a segunda do SMOTE. A Figura 11 ilustra como

acontecem os testes e a busca. Como pode ser observado, a função *RandomizedSearchCV* seleciona 10 valores de forma aleatória para realizar os testes, a partir disso, os 5 testes são executados e no final, o melhor hiperparâmetro é apresentando através da média aritmética da acurácia de todos os testes (a última coluna demonstra a média – *mean_test_score*). Vale ressaltar que a acurácia é de valor numérico e varia entre 0 e 1, sendo 0 para o pior valor e 1 para o melhor.

A acurácia da melhor média foi de **0.6585** que é referente ao k igual a 10, logo, o valor selecionado para o hiperparâmetro *n_neighbors* é 10 (Figura 11).

Figura 11 - Busca pelo hiperpâmetro n_neighbors com balanceamento RU.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
{'n_neighbors': 3}	0.6577	0.6107	0.6846	0.6779	0.6216	0.6505
{'n_neighbors': 22}	0.6107	0.6846	0.6711	0.6644	0.5676	0.6397
{'n_neighbors': 30}	0.6644	0.6711	0.6309	0.6846	0.5676	0.6437
{'n_neighbors': 18}	0.6443	0.6846	0.6443	0.6577	0.5946	0.6451
{'n_neighbors': 8}	0.6376	0.6913	0.6711	0.6779	0.6081	0.6572
{'n_neighbors': 1}	0.7047	0.5973	0.6309	0.7181	0.6014	0.6505
{'n_neighbors': 17}	0.6242	0.6711	0.6510	0.6443	0.6014	0.6384
{'n_neighbors': 10}	0.6711	0.6779	0.6711	0.6577	0.6149	0.6585
{'n_neighbors': 19}	0.6443	0.6846	0.6242	0.6644	0.5811	0.6397
{'n_neighbors': 11}	0.6309	0.6644	0.6376	0.6644	0.6284	0.6451

Fonte: Elaborado pelo autor.

A Figura 12 apresenta a execução da busca pelo hiperparâmetro fazendo o uso do balanceamento SMOTE. De acordo com a linha em destaque na Figura 12, obteve-se como melhor resultado, a média (*mean_test_score*) nos testes realizados com o k igual a 3, com acurácia de **0.9797**. Portanto, foi atribuído ao hiperparâmetro *n_neighbors* o valor 3.

Figura 12 - Busca pelo hiperpâmetro n_neighbors com balanceamento SMOTE.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
{'n_neighbors': 23}	0.9243	0.9245	0.9245	0.9257	0.9244	0.9247
{'n_neighbors': 28}	0.9133	0.9141	0.9135	0.9157	0.9138	0.9141
{'n_neighbors': 3}	0.9794	0.9799	0.9797	0.9802	0.9794	0.9797
{'n_neighbors': 27}	0.9164	0.9168	0.9170	0.9186	0.9168	0.9171
{'n_neighbors': 8}	0.9613	0.9625	0.9613	0.9631	0.9606	0.9618
{'n_neighbors': 11}	0.9534	0.9545	0.9532	0.9557	0.9532	0.9540
{'n_neighbors': 19}	0.9326	0.9330	0.9328	0.9344	0.9326	0.9331
{'n_neighbors': 17}	0.9373	0.9379	0.9377	0.9391	0.9372	0.9378
{'n_neighbors': 7}	0.9657	0.9678	0.9665	0.9681	0.9661	0.9668
{'n_neighbors': 12}	0.9489	0.9498	0.9486	0.9506	0.9486	0.9493

Fonte: Elaborado pelo autor.

5.1.2 Funções Kernel SVM (*Kernel*)

Para encontrar o melhor hiperparâmetro para funções kernel do SVM, foram aplicados testes em 4 kernels distintos: gaussiano (RBF), polinomial, sigmóide e linear, pois, com base no trabalho de Oliveira Junior (2010), são esses os mais utilizados no âmbito acadêmico. A Figura 13, destaca que o kernel linear apresentou a melhor média da acurácia (*mean_test_score*) com o valor de **0.9005**.

Figura 13 - Busca pelo hiperpâmetro Kernel com balanceamento RU.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
{'kernel': 'rbf'}	0.5638	0.5369	0.5101	0.5436	0.5270	0.5363
{'kernel': 'poly'}	0.5570	0.5839	0.5503	0.5973	0.4932	0.5564
{'kernel': 'sigmoid'}	0.5034	0.5906	0.4899	0.5101	0.4459	0.5080
{'kernel': 'linear'}	0.8993	0.8859	0.9195	0.9195	0.8784	0.9005

Fonte: Elaborado pelo autor.

Já a Figura 14 apresenta os resultados obtidos com o balanceamento SMOTE. A média da acurácia mais alta de **0.9984**, se deu também com o kernel linear, porém, como pode-se notar na última coluna da tabela (*mean_test_score*), a média de todos os resultados foram extremamente próximos, contrário aos resultados alcançando fazendo uso da normalização RU, que obteve o kernel linear com acurácia muito

superior ao demais. Vale ressaltar que o kernel linear é bastante utilizado para resolução de problemas de aprendizagem, com ele, é possível resolver problemas lineares, através do mapeamento para um espaço de maior dimensão (OLIVEIRA JUNIOR, 2010).

Figura 14 - Busca pelo hiperpâmetro Kernel com balanceamento SMOTE.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
{'kernel': 'rbf'}	0.9983	0.9983	0.9983	0.9982	0.9982	0.9983
{'kernel': 'poly'}	0.9983	0.9983	0.9983	0.9982	0.9982	0.9983
{'kernel': 'sigmoid'}	0.9983	0.9983	0.9983	0.9982	0.9982	0.9983
{'kernel': 'linear'}	0.9985	0.9984	0.9984	0.9984	0.9985	0.9984

Fonte: Elaborado pelo autor.

5.1.3 Árvores na floresta (*n_estimators*)

A busca pelo melhor valor para o hiperparâmetro *n_estimators*, que corresponde ao número de árvores de decisão na *Random Forest*, fez uso dos valores 90, 100, 200, 300, 500, 750, 1000, 1500 e 2000 estimadores para realizar os testes. Devido às limitações acerca do tempo, para esse trabalho optou-se por usar alguns valores de estimadores descritos no trabalho de Nicola, Lauretto e Delgado (2020) e os outros foram escolhidos de forma aleatória.

A Figura 15 apresenta o resultado da busca utilizando o balanceamento RU. Neste caso, observa-se que o valor para *n_estimators* que obteve a média de maior acurácia **0.9355** (*mean_test_score*), corresponde à **100** árvores.

Figura 15 - Busca pelo hiperparâmetro n_estimators com balanceamento RU.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
{'n_estimators': 90}	0.9396	0.9128	0.9262	0.9530	0.9392	0.9341
{'n_estimators': 100}	0.9396	0.9262	0.9262	0.9463	0.9392	0.9355
{'n_estimators': 200}	0.9396	0.9195	0.9195	0.9463	0.9459	0.9342
{'n_estimators': 300}	0.9396	0.9128	0.9195	0.9463	0.9527	0.9342
{'n_estimators': 500}	0.9463	0.9128	0.9195	0.9463	0.9392	0.9328
{'n_estimators': 750}	0.9463	0.9128	0.9262	0.9463	0.9392	0.9341
{'n_estimators': 1000}	0.9396	0.9060	0.9195	0.9463	0.9459	0.9315
{'n_estimators': 1500}	0.9463	0.8993	0.9262	0.9463	0.9392	0.9315
{'n_estimators': 2000}	0.9396	0.9060	0.9195	0.9463	0.9459	0.9315

Fonte: Elaborado pelo autor.

E por último, como é observado na Figura 16, o melhor resultado obtendo a média da acurácia de **0.9436 (***mean_test_score***)**, utilizando o balanceamento SMOTE, ficou com o número de 300 árvores de decisão para definir a floresta.

Figura 16 - Busca pelo hiperparâmetro n_estimators com balanceamento SMOTE.

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score
{'n_estimators': 90}	0.9329	0.9463	0.9396	0.9195	0.9459	0.9368
{'n_estimators': 100}	0.9329	0.9329	0.9396	0.9262	0.9459	0.9355
{'n_estimators': 200}	0.9396	0.9463	0.9396	0.9262	0.9392	0.9382
{'n_estimators': 300}	0.9396	0.9463	0.9463	0.9329	0.9527	0.9436
{'n_estimators': 500}	0.9396	0.9463	0.9396	0.9262	0.9527	0.9409
{'n_estimators': 750}	0.9329	0.9396	0.9463	0.9329	0.9459	0.9395
{'n_estimators': 1000}	0.9396	0.9396	0.9396	0.9329	0.9527	0.9409
{'n_estimators': 1500}	0.9329	0.9396	0.9396	0.9396	0.9527	0.9409
{'n_estimators': 2000}	0.9396	0.9396	0.9463	0.9262	0.9527	0.9409

Fonte: Elaborado pelo autor.

5.2 Avaliação dos Modelos Preditivos

Após encontrar os melhores valores para os hiperparâmetros de cada algoritmo, avançou-se para a etapa de treinamento e teste dos modelos preditivos. Para avaliar a performance dos testes, utilizou-se como métricas de avaliação a

acurácia e F1-*Score*. Os três tópicos seguintes apresentam os resultados obtidos nas predições.

5.2.1 k-NN

Os valores de acurácia obtidos nos testes com k-NN estão expostos na Tabela 1. Nota-se que a melhor acurácia, **0.9989 (99,89%),** foi obtido através da técnica de seleção de atributos de 40% do *dataset* (12 colunas), e utilizando do balanceamento SMOTE.

Tabela 1 - Acurácia alcançada nos experimentos com k-NN.

ACURÁCIA – k-NN		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.6460	0.8668
SMOTE	0.9984	**0.9989**

Fonte: Elaborado pelo autor.

Para os testes apresentarem melhores resultados efetivos, este trabalho também se utiliza da métrica F1-*Score* para avaliar o modelo preditivo. A Tabela 2 demonstra os resultados obtidos com essa métrica, que combina precisão e revocação. Observa-se que, como a acurácia, o melhor F1-*Score* se deu também com 40% do conjunto e fazendo uso da técnica de balanceamento SMOTE, atingindo um resultado de **0.9987 (99,87%)**.

Tabela 2 - F1-*Score* alcançado nos experimentos com k-NN.

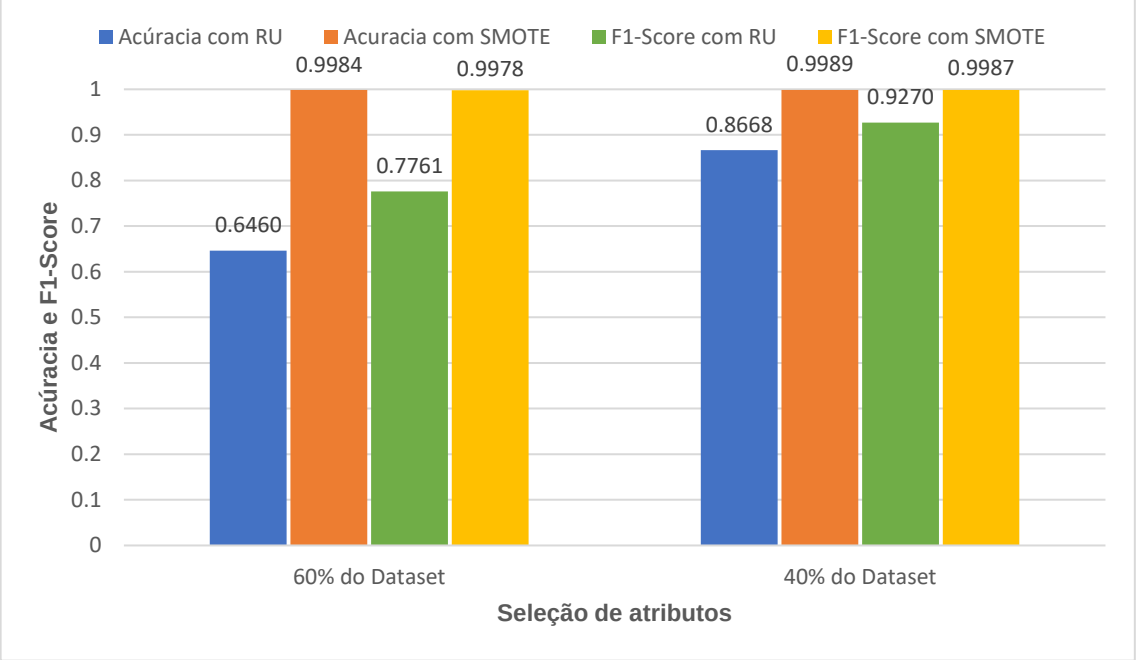
F1-*SCORE* – k-NN		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.7761	0.9270
SMOTE	0.9978	**0.9987**

Fonte: Elaborado pelo autor.

O que é notável no experimento com o algoritmo k-NN, corresponde aos melhores resultados obtidos através do balanceamento SMOTE em ambos os modos de seleção de atributos, além de, os valores de acurácia quanto do F1-*Score* serem

bastante próximos. O Gráfico apresentado na Figura 17 apresenta um maior panorama destes valores obtidos com k-NN.

Figura 17 – Gráfico com resultados alcançados nos experimentos com k-NN.



Fonte: Elaborado pelo autor.

A barra em azul representa a acurácia com a técnica de balanceamento RU, a barra em laranja representa a acurácia com o balanceamento SMOTE, enquanto a barra verde representa o F1-*Score* com balanceamento RU e por último a barra amarela representa o F1-*Score* com o balanceamento SMOTE. Como se pode perceber as barras se repetem em dois grupos, sendo, o primeiro representando os experimentos com 60% do *dataset* e o segundo grupo representando os experimentos com 40%.

5.2.2 SVM

As acurácias obtidas nas predições com SVM estão exibidas na Tabela 3. Observa-se que o melhor resultado, **0.9986 (99,86%)**, foi adquirido na predição a qual houve o balanceamento SMOTE e a seleção de 60% do conjunto (18 colunas).

Tabela 3 - Acurácia alcançada nos experimentos com SVM.

ACURÁCIA – SVM		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.9957	0.9080
SMOTE	**0.9986**	0.9974

Fonte: Elaborado pelo autor.

A Tabela 4 exibe os resultados da métrica F1-*Score*. Novamente, o experimento que obteve o melhor F1-*Score*, **0.9983 (99,83%)**, foi a mesma combinação que obteve a melhor acurácia, fazendo o uso do SMOTE como balanceamento e 60% dos atributos.

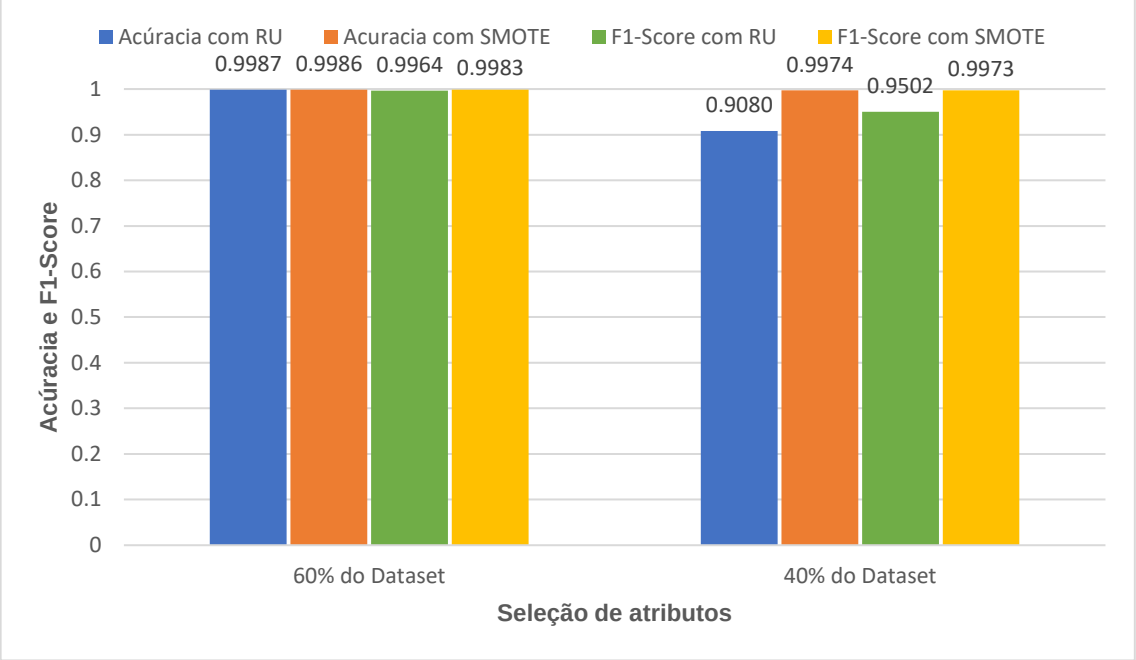
Tabela 4 - F1-*Score* alcançado nos experimentos com SVM.

F1-*SCORE* – SVM		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.9964	0.9502
SMOTE	**0.9983**	0.9973

Fonte: Elaborado pelo autor.

Diferente do k-NN, o algoritmo SVM, adquiriu um comportamento com melhores resultados com a seleção de atributos de 60% do conjunto. Entretanto, também obteve os melhores valores de acurácia e F1-*Score* com a técnica de balanceamento SMOTE. Para melhor visualização observe a Figura 18:

Figura 18 – Gráfico com resultados alcançados nos experimentos com SVM.



Fonte: Elaborado pelo autor.

O Gráfico da Figura 18 apresenta os valores de acurácia e F1-*Score* obtidos com SVM. Seguindo a mesma metodologia do gráfico anterior (Figura 17), as barras azuis e laranjas representam a acurácia com o balanceamento RU e SMOTE, respectivamente, enquanto as barras verdes e amarelas representam o F1-*Score* com o balanceamento RU E SMOTE, e então são divididos em dois grupos de barras, sendo um com 60% do *dataset* e outro com 40%.

5.2.3 RF

Por último, os resultados de acurácia provenientes da avaliação com *Randon Forest* são apresentados na Tabela 5. Assim como nas predições com o algoritmo SVM, o melhor resultado, **0.9995 (99,95%)**, é proveniente do subconjunto de dados de 60% dos atributos (18 colunas) e com o balanceamento SMOTE. Todavia, observa-se que a predição com o método de balanceamento SMOTE, resultou em valores maiores que o balanceamento RU.

Tabela 5 - Acurácia alcançada nos experimentos com RF.

ACURÁCIA – *Random Forest*		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.9764	0.9191
SMOTE	**0.9995**	0.9990

Fonte: Elaborado pelo autor.

Como na acurácia, os resultados das predições com *Random Forest* avaliados com a métrica F1-*Score*, mostram que a técnica de balanceamento SMOTE combinado com a seleção de atributos de 60%, ocasionou a melhor pontuação, **0.9994 (99,94%)**, de maneira que, pode ser observado na Tabela 6.

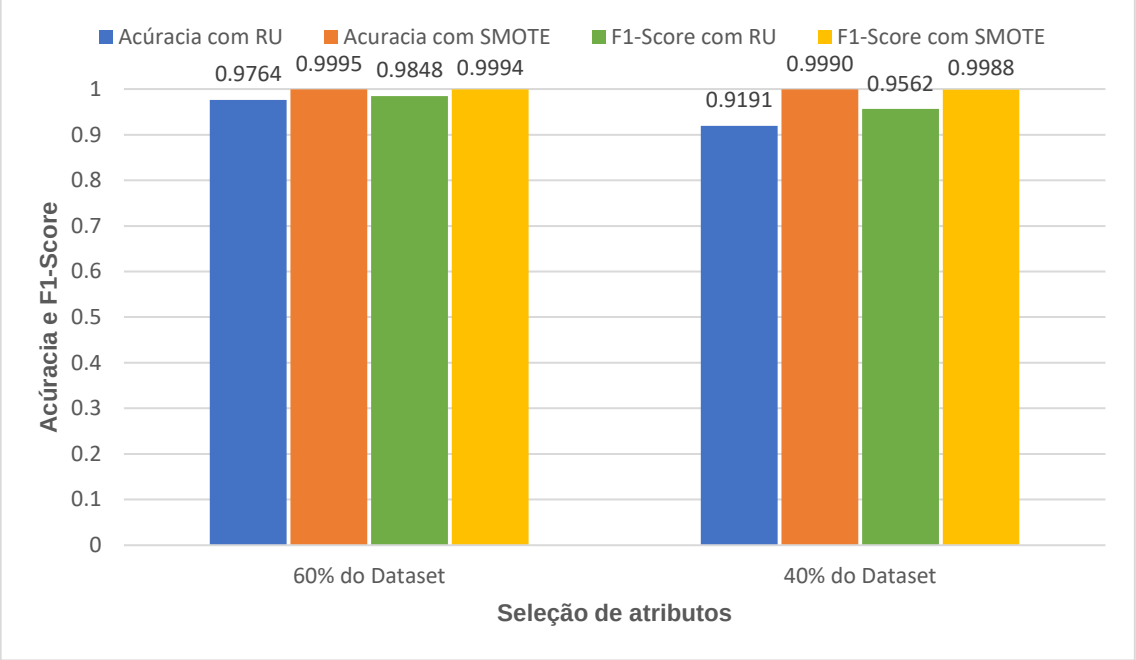
Tabela 6 - F1-*Score* alcançado nos experimentos com RF.

F1-*SCORE* – *Random Forest*		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.9848	0.9562
SMOTE	**0.9994**	0.9988

Fonte: Elaborado pelo autor.

Assim como no modelo preditivo SVM, os melhores resultados de acurácia e F1-*Score* foram alcançados utilizando do balanceamento SMOTE e de 60% do conjunto. É melhor ilustrado na Figura 19.

Figura 19 – Gráfico com resultados alcançados nos experimentos com RF.



Fonte: Elaborado pelo autor.

Da mesma maneira que os gráficos apresentando os resultados do k-NN e SVM, o Gráfico da figura 19 apresenta os resultados obtidos no RF e suas cores se reprisam, sendo que as barras azuis e laranjas representam a acurácia com o balanceamento RU e SMOTE, respectivamente, enquanto as barras verdes e amarelas representam o F1-*Score* com o balanceamento RU E SMOTE, também, respectivamente, divididos em dois grupos de barras, com 60% e 40% do *dataset*.

5.2.4 Análise Geral

Para uma análise geral e comparativa entre os modelos preditivos k-NN, SVM e *Random Forest*, em cada predição, calculou-se a média aritmética entre a acurácia e a métrica F1-*Score*. Os resultados são apresentados na Tabela 7 (k-NN), Tabela 8 (SVM) e Tabela 9 (*Random Forest*).

Tabela 7 – Média entre Acurácia e F1-*Score* alcançada nos experimentos com k-NN.

MÉDIA – k-NN		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.7110	0.8969
SMOTE	0.9981	**0.9988**

Fonte: Elaborado pelo autor.

Tabela 8 – Média entre Acurácia e F1-*Score* alcançada nos experimentos com SVM.

MÉDIA – SVM		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.9960	0.9291
SMOTE	**0.9984**	0.9973

Fonte: Elaborado pelo autor.

Tabela 9 - Média entre Acurácia e F1-*Score* alcançada nos experimentos com RF.

MÉDIA – *Random Forest*		
Método de Balanceamento	Método de Seleção de atributos	
	60% do conjunto	40% do conjunto
RU	0.9862	0.9376
SMOTE	**0.9994**	0.9989

Fonte: Elaborado pelo autor.

Observa-se que, nos três algoritmos, ao considerar a relação entre as métricas acurácia e F1-*Score*, as melhores predições são resultantes do método de balanceamento SMOTE. O k-NN obteve seu maior valor médio de **0.9988 (99.88%)**, com a seleção de atributos de 40% do conjunto, enquanto os algoritmos SVM, **0,9984 (99,84%),** e *Random Forest,* ***0,9994 (99,94%)*** obtiveram seus melhores valores utilizando 60% do conjunto.

6 CONCLUSÃO

Diante do crescente número de fraudes em transações de cartões de crédito e transtornos provocados por elas, este trabalho ansiou avaliar os modelos preditivos de k-NN, SVM e *Random Forest* para identificar a abordagem que melhor se adequa em predições de fraudes em transações. Para isso, utilizou-se o *dataset* pré-processado fornecido por *Machine Learning Group - Université Libre* de Bruxelas na plataforma *Kaggle* (2018), o qual forneceu dados referentes a dois dias de transações de cartão de crédito que ocorreram em bancos europeus. Além do mais, a base de dados foi balanceada a fim de otimizar os resultados.

Para a execução dos procedimentos, primeiramente foi aplicada uma pesquisa acerca dos modelos preditivos empregados neste trabalho, a qual, estruturou tanto o referencial teórico, quanto todo o prático. Como ambiente de desenvolvimento, escolheu-se a aplicação web *Google Colab*, em companhia da linguagem *Python* e suas bibliotecas.

A metodologia para realizar as predições, consistiu em primeiro lugar, em aplicar a técnica de pesquisa aleatória, para definir os melhores hiperparâmetros dos modelos preditivos. Após identificá-los, diversas atribuições abrangendo métodos de seleção de atributos e de balanceamento de dados foram executadas. Para avaliar os resultados, adotou-se acurácia e F1-*Score* como métricas.

No modelo k-NN, a melhor acurácia foi de 0.9989 (99,89%). Esse valor é resultante da seleção de atributos com 40% do conjunto de dados, e com balanceamento SMOTE. De modo igual, a maior pontuação para a métrica F1-*Score*, de 0.9987 (99,87%), foi resultante também do mesmo subconjunto de seleção de atributos e balanceamento (40% do conjunto e SMOTE).

Para os melhores resultados da acurácia e F1-*Score* obtidos no modelo SVM, foram 0.9986 (99,86%) e 0.9983 (99,83%), respectivamente. Tanto os valores de acurácia quanto do F1-*Score* foram atingidos através do *dataset* com seleção de atributos de 60% do conjunto e fazendo o uso do balanceamento SMOTE.

O terceiro e último modelo avaliado, *Random Forest*, resultou em 0.9995 (99,95%), de acurácia na melhor predição. Para isso, o conjunto de dados utilizado foi de 60% dos atributos combinado com a técnica de balanceamento SMOTE, semelhante ao que aconteceu ao modelo SVM. A melhor pontuação da métrica F1-

Score para a *Random Forest*, 0.9994 (99,94%), ocorreu no mesmo cenário que a acurácia.

Diante dos resultados apresentados, concluiu-se que, nos três algoritmos de predição, as acurácias e F1-*Score* mais altos foram resultantes da técnica de balanceamento de dados SMOTE. Então, para uma análise que relacionasse as duas métricas, calculou-se a média aritmética entre acurácia e F1-*Score* em cada predição.

Baseado nos resultados da média aritmética, descobriu-se que, tanto no modelo SVM, quanto no *Random Forest*, o melhor aproveitamento derivou-se da seleção de atributos de 60% do conjunto de dados com balanceamento SMOTE. Enquanto o modelo k-NN, obteve seu melhor aproveitamento também fazendo uso do balanceamento SMOTE, porém, com a seleção de atributos contando com 40% do conjunto. Os valores mais satisfatórios foram: 0.9988 (99,88%) para k-NN, 0.9984 (99,84%) para SVM e 0.9994 (99,94%) para *Random Forest*.

Também baseado nos valores adquiridos da média aritmética, foi observado que os melhores resultados de cada modelo preditivo foram consideravelmente próximos um ao outro. Ainda assim, ressaltando o modelo *Random Forest* que obteve um exíguo melhor resultado se comparado aos demais algoritmos de 0.9994 (99,94%), seguido pelo modelo k-NN, 0.9988 (99,88%), e em último lugar apresentando o pior desempenho ficando com o SVM, 0.9984 (99,84%).

Portanto, nas predições de fraudes em transações de cartões de crédito expostas neste trabalho, utilizando o conjunto de dados disponibilizado por *Machine Learning Group - Université Libre* de Bruxelas e a metodologia apresentada, o modelo que melhor se adaptou foi *Random Forest*. A melhor predição, considerando a relação entre as métricas de avaliação, alcançou 0.9995 (99,95%) de acurácia e 0.9994 (99,94%) de F1-*Score*. Apesar deste trabalho apresentar desempenho maior, esse resultado condiz com as conclusões tiradas em Nicola, Lauretto e Delgado (2020), onde as melhores predições para fraudes em transações de cartões de crédito foram obtidas com o modelo preditivo de *Random Forest*. O código-fonte dos experimentos realizados neste trabalho são disponibilizados no repositório https://github.com/Tulio-Cesar/analise-preditiva-fraude-em-transacoes.

Sugere-se como trabalhos futuros, testar outros algoritmos, como por exemplo algoritmos de Regressão e/ou Clusterização, e claro, buscar seus melhores hiperparâmetros, além de, aprofundar os conhecimentos teóricos sobre as técnicas

de balanceamento e de seleção de atributos. E se possível, realizar parcerias com instituições financeiras que realizam transações com cartão de crédito com o intuito de obter diferentes bases de dados (*datasets*).

REFERÊNCIAS

BARELLA, Victor Hugo. **Técnicas para o problema de dados desbalanceados em classificação hierárquica**. 2015. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2015. doi:10.11606/D.55.2016.tde-06012016-145045. Acesso em: 15 nov. 2019.

BENZ, Gabriel Lorenzoni. **Sistema de Apoio à Detecção de Fraudes em E-Commerce**. 2017. 100 f. Dissertação (Mestrado em Engenharia de Controle e Automação) - Instituto Federal do Espírito Santo, 2017.

BREIMAN, Leo. Random Forests. Machine Learning, [S. l.], v. 45, n. 1, p.5-32, 2001. Springer Nature. http://dx.doi.org/10.1023/a:1010933404324.

CHANG, C.-C. & LIN, C.-J. (2011). **Libsvm: A library for support vector machines.** ACM Trans. Intell. Syst. Technol., 2(3):27:1--27:27. ISSN 2157-6904.

CHAWLA, Nitesh V. et al. SMOTE: Synthetic Minority Over-sampling Technique. Journal Of Artificial Intelligence Research, [S. l.], v. 16, p.321-357, 2002.

COLAB. **Colab: Olá, este é o Colaboraty** – Google 2020. Disponível em: <https://colab.research.google.com/notebooks/intro.ipynb>. Acesso em: 04 nov. 2020.

CUNNINGHAM, Padraig; DELANY, Sarah Jane. **k-Nearest neighbour classifiers**. Multiple Classifier Systems, v. 34, n. 8, p. 1-17, 2007.

FELIPE JÚNIOR, José. **Mineração de dados para detecção de fraudes em transações eletrônicas**. 2012. 110 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Belo Horizonte, 2012.

Gunn, S. R. **Support Vector Machine for Classification and Regression.** Faculty of Engineering, Science and Mathematics School of Electronics and Computer Science, 1998.

JAADI, Zakaria. **A step by step explanation of Principal Component Analysis**. [S. l.], 28 fev. 2017. Disponível em: https://towardsdatascience.com/a-step-by-step-explanationof-principal-component-analysis-b836fb9c97e2. Acesso em: 10 abr. 2019.

KAGGLE. **Credit Card Fraud Detection** - Machine Learning Group - ULB 2018 <https://www.kaggle.com/mlg-ulb/creditcardfraud>. Acesso em: 08 set 2020.

LORENA, Ana C.; CARVALHO, André C. P. L. F.) **Uma Introdução às Support Vector Machines**. 25 f. 2007. DOI: https://doi.org/10.22456/2175-2745.5690.

MCKINNEY, Wes. **Python for Data Analysis**. Sebastopol: O'reilly Media, 2013.

NICOLA, Victor; LAURETTO, Marcelo; VALDIVIA DELGADO, Karina. Avaliação empírica de classificadores e métodos de balanceamento para detecção de fraudes em transações com cartões de créditos. *In*: ENCONTRO NACIONAL DE INTELIGÊNCIA ARTIFICIAL E COMPUTACIONAL (ENIAC), 17. , 2020, Evento Online. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, 2020 . p. 70-81. DOI: https://doi.org/10.5753/eniac.2020.12118.

OLIVEIRA JUNIOR, Gilson Medeiros. **Maquinhas de Vetores de Suporte: Estudos e Análise de Parâmetros Para Otimização de Resultado**. 2010. 46 f. Trabalho de Conclusão de Curso (Bacharel em Ciência da Computação) – Centro de Informática, Universidade Federal de Pernambuco, 2010.

OLIVEIRA, Paulo Henrique Maestrello Assad. **Detecção de fraudes em cartões: um classificador baseado em regras de associação e regressão logística**. 2016. 103 f. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

PANDAS. **Pandas**: powerful Python data analysis toolkit – pandas 0.25.3. 2019. Disponível em: < https://pandas.pydata.org/pandas-docs/stable/>. Acesso em: 06 nov. 2019.

QUINLAN, J. R. Induction of Decision Trees. **Machine Learning**, Boston, v. 1, n. 1, p.81-106, 1986.

RUSSELL, Stuart; NORVING, Peter. **Inteligência Artificial**. 2. ed. Rio de Janeiro: Campus-elsevier, 2004. 1021 p.

SCIKIT-LEARN. **Scikit-learn user guide**: Release 0.21.3. 2019. Disponível em: < https://scikit-learn.org/stable//_downloads/scikit-learn-docs.pdf>. Acesso em: 06 nov. 2020.

SEABORN. **Seaborn**: statistical data visualization. 2020. Disponível em: < https://seaborn.pydata.org/index.html>. Acesso em: 10 mar. 2021.

SHALEV-SHWARTZ, Shai; BEN-DAVID, Shai. **Understanding Machine Learning**: From Theory to Algorithms. New York: Cambridge University Press, 2014. 449 p. (ISBN: 978-1-107-05713-5).

SOUZA, Heldo Augusto Rodrigues. **Estudo de Algoritmos para Predição de Atrasos Aéreos**. 2019. 69 f. Trabalho de Conclusão de Curso (Bacharel em Ciências da Computação) - Escola de Ciências Exatas e da Computação, Pontifícia Universidade Católica de Goiás, Goiânia, 2019.

VIEIRA, Hugo Fernandes. **Detecção de Fraude no Uso de Cartões de Crédito Utilizando Técnicas Supervisionadas de *Machine Learning***. 2018. 51 f. Trabalho

de Conclusão de Curso (Bacharel em Engenharia da Computação) – Centro Universitário de Anápolis, Anápolis, 2018.

WITTEN, Ian H.; FRANK, Eibe; HALL, Mark A. **Data Mining**: Practical Machine Learning Tools and Techniques. 3. ed. Burlington: Elsevier, 2011. 629 p.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Túlio César Leite de Sousa do Curso de Ciência da Computação, matrícula 2021.1.0028.0057-5, telefone: (62) 98522-4733 e-mail tuliocesar.ls@hotmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado ESTUDO DE ALGORITMOS DE MACHINE LEARNING PARA PREDIÇÃO DE FRAUDES EM CARTÕES DE CRÉDITO, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 12 de JUNHO de 2021.

Assinatura do(s) autor(es): Túlio César Leite de Sousa

Nome completo do autor: TÚLIO CÉSAR LEITE DE SOUSA

Assinatura do professor-orientador:

Nome completo do professor-orientador: ANIBAL SANTOS JUKEMURA