UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE LORENA

DOUGLAS SARDISCO

Estudo comparativo entre modelos preditivos para detecção de fraude em transações de cartão de crédito

Lorena
2021

DOUGLAS SARDISCO

Estudo comparativo entre modelos preditivos para detecção de fraude em transações de cartão de crédito

Projeto para desenvolvimento no trabalho de conclusão de curso apresentado à Escola de Engenharia de Lorena da Universidade de São Paulo para obtenção do diploma de graduação em Engenharia Física

Orientador: Prof. Dr. Carlos Yujiro Shigue

Lorena
2021

Ficha catalográfica elaborada pelo Sistema Automatizado
da Escola de Engenharia de Lorena,
com os dados fornecidos pelo(a) autor(a)

Sardisco, Douglas
Estudo comparativo entre modelos preditivos para detecção de fraude em transações de cartão de crédito / Douglas Sardisco; orientador Carlos Yujiro Shigue. - Lorena, 2021.
56 p.

Monografia apresentada como requisito parcial para a conclusão de Graduação do Curso de Engenharia Física - Escola de Engenharia de Lorena da Universidade de São Paulo. 2021

1. Modelos preditivos. 2. Fraude. 3. Fraude em transações. 4. Cartão de crédito. 5. Aprendizado de máquina. I. Título. II. Shigue, Carlos Yujiro, orient.

Dedico este trabalho a minha família, em especial meus pais, que sempre me transmitiram amor e apoio.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por auxiliar e oportunizar todos os momentos vivenciados. Sem ele, não seria possível.

Agradeço grandiosamente toda minha família, em especial meus pais, Giovanni e Margareth, por todo alicerce, esforços realizados e confiança depositada em toda minha vida. Sem eles, não seria possível realizar o sonho de cursar o bacharelado na melhor universidade da américa latina, ainda mais há 500 km da charmosa Araraquara-SP, orgulhosamente minha terra natal.

A todos os irmãos que encontrei na República Rapadura, que sempre estiverem juntos nessa jornada, tanto nos ótimos momento vivenciados, quanto nos tristes e eufóricos problemas superados. Em especial ao Matheus Patusco e Matheus Bianco, com suas particularidades, fizeram esses anos ainda melhores.

A todos os amigos de São Paulo, que o estágio proporcionou. Em especial, minha companheira para a vida, Karyn. Em todas as dificuldades me impulsionou, compreendeu o momento conturbado das viagens entre estágio (SP) e finalização do curso em Lorena e sempre, sempre esteve ao meu lado. Muito obrigado!

A todos os mestre e doutores que realizam a verdadeira ciência no campus da EEL. Mesmo com diversas dificuldades, jamais desviam da sua nobre missão perante a sociedade atual. Nesse contexto, agradeço Prof. Dr. Carlos Yujiro Shigue, por aceitar orientar-me nesse derradeiro projeto e por sempre contribuir para um ensino mais humano e atual, conforme a sociedade necessita.

RESUMO

Pelo aumento do pagamento através de cartão de crédito, o número de fraudes e seus diversos tipos crescem paralelamente. Consequentemente, a busca por métodos de diminuir o prejuízo causado é um dos temas mais valorizados pelas grandes empresas do setor no cenário atual. Nesse contexto, esse trabalho traz o emprego de Aprendizado de Máquina utilizando quatro modelos preditivos de classificação, sendo eles: Regressão Logística, k-Nearest Neighbors, Support Vector Machines e Árvores de Decisão, utilizando um dataframe com transações reais. A metodologia aplicada buscou analisar os principais hiperparâmetros e técnicas de balanceamento de dados a fim de melhorar a performance dos modelos usados. Além disso, métricas de avaliação de desempenho foram aplicadas com o intuito de comparar a melhor performance entre eles, resultando em acurácia 0,88306 e F1-Score 0,90147 para o dataset balanceado com a técnica SMOTE predição utilizando Regressão Logística.

Palavras-chave: *Modelos preditivos; fraude; fraude em transações; cartão de crédito; aprendizado de máquina.*

ABSTRACT

Due to the increase in payment by credit card, the number of frauds and their various types grow in parallel. Consequently, the search for methods to reduce the damage caused is one of the themes most valued by large companies in the sector in the current scenario. In this context, this work brings the use of Machine Learning using four predictive classification models, namely: Logistic Regression, k-Nearest Neighbors, Support Vector Machines and Decision Trees, using a dataframe with real transactions. The applied methodology sought to analyze the main hyperparameters and data balancing techniques in order to improve the performance of the models used. In addition, performance evaluation metrics were applied in order to compare the best performance between them, resulting in accuracy of 0.88306 and F1-Score 0.90147 for the balanced dataset with the SMOTE prediction technique using Logistic Regression.

Keywords: *Predictive models; fraud; transaction fraud; credit card; machine knowledge*

LISTA DE FIGURAS

Figura 1 – Diagrama de Venn das competências da Ciência de Dados. 13

Figura 2 – Fraude com cartão de crédito nos EUA em 2018. 14

Figura 3 – Tipos de ML e suas principais empregabilidades 19

Figura 4 – Principais linguagens e a % de menções no site Stack Overflow 20

Figura 5 – Principais bibliotecas python e a % de menções no site Stack Overflow..... 21

Figura 6 – Representação de um hiperplano para duas classes..... 24

Figura 7 – Representação de um hiperplano para duas classes, margem e os vetores de suporte. 25

Figura 8 – Representação esquemática de uma floresta aleatória composta por árvores de decisão. 26

Figura 9 – Representação esquemática de uma floresta aleatória composta por árvores de decisão e sua terminologia. 27

Figura 10 – Pontos de dispersão fictícios da relação de variável dependente e independente da regressão logística..... 29

Figura 11 – Classificação pelo modelo KNN..... 29

Figura 12 – Diversos modelos de classificação e como os dados comportam-se. 33

Figura 13 – Metodologia de trabalho em um projeto de ML. 34

Figura 14 – Gráfico da quantidade de transações fraudulentas e não fraudulentas, e sua representatividade (%) perante o dataset 35

Figura 15 – Gráfico da quantidade de transações fraudulentas e não fraudulentas, e sua representatividade (%) perante a subamostragem via Random Undersampling..... 37

Figura 16 – Gráfico da quantidade de transações fraudulentas e não fraudulentas, e sua representatividade (%) perante a subamostragem via SMOTE 38

Figura 17 – Matriz de correlação do dataframe original (desbalanceado) 39

Figura 18 – Matriz de correlação do dataframe balanceado (via Random Undersampling) . 39

Figura 19 – Invocação dos classificadores em Python. 42

Figura 20 – Gráfico da acurácia de F1-Score obtidos através da regressão logística..... 47

Figura 21 – Gráfico da acurácia de F1-Score obtidos através da KNN..... 48

Figura 22 – Gráfico da acurácia de F1-Score obtidos através da SVM..... 49

Figura 23 – Gráfico da acurácia de F1-Score obtidos através das Árvores de Decisão 50

Figura 24 – Gráfico da área sobre a curva ROC dos classificadores testados com os dataframes RU e SMOTE 51

LISTA DE TABELAS

Tabela 1 – Classificação das classes saídas dos modelos abordados. 31
Tabela 2 – Níveis de classificação gerado pela área da curva ROC 32
Tabela 3 – Modelos e hiperparâmetros estudos e suas definições 40
Tabela 4 – Possíveis valores para os hiperparâmetros 41
Tabela 5 – Hiperparâmetros para Regressão Logística. 44
Tabela 6 – Hiperparâmetros para KNN 45
Tabela 7 – Hiperparâmetros para SVM 45
Tabela 8 – Hiperparâmetros para Árvores de decisão 46

LISTA DE SIGLAS

ML - Machine Learning

KNN – K- Nearest Neighbors

SVM – Support Vector Machine

RV – Random Undersampling

SMOTE – Synthetic Minority Oversampling Technique

DATASET- Conjunto de dados

SUMÁRIO

1. INTRODUÇÃO 13
1.1 Contextualização 13
1.2 Objetivos 15
1.2.1 Objetivo Geral 15
1.2.2 Objetivos Específicos 15
2. Referência Teórico 16
2.1 Fraudes em cartão de crédito 16
2.2 Relação entre fraude e o cenário atual da computação 18
2.3 Python 19
2.4 Classificadores 22
2.5 Modelo Lineares 22
2.5.1 SVM – Máquina de vetores de suporte 23
2.5.2 Árvores de decisão 25
2.5.3 Regressão logística 28
2.6 K-Nearest Neighbors 29
2.7 Validação Cruzada 30
2.8 Métricas de avaliação de desempenho 30
2.8.1 Acurácia 31
2.8.2 Precisão 31
2.8.3 Revocação 32
2.8.4 F1 – Score 32
2.8.5 *Área sob a curva ROC* 32
3. METODOLOGIA 34
3.1 Exploração do conjunto de dados 34
3.1.4 Hiperparâmetros 39
3.2 Treinamento, teste e avaliação dos modelos 41
4. RESULTADO E DISCUSSÃO 43
4.1 Escolha dos hiperparâmetros 43
4.1.1 Hiperparâmetros - Regressão Logística 43
4.1.2 Hiperparâmetros – KNN 44
4.1.3 Hiperparâmetros – SVM 45
4.1.4 Hiperparâmetros – Árvores de Decisão 46
4.2 Modelos preditivos 47
4.2.1 Regressão logística 47

4.2.2 KNN 48
4.2.3 SVM 48
4.2.4 Árvores de Decisão 49
4.2.5 Curva ROC 50
4.2.6 Overview 51
5. CONCLUSÃO 52
REFERÊNCIAS BIBLIOGRÁFICAS 54
APÊNDICES 56

1. INTRODUÇÃO

1.1 Contextualização

O tratamento e interpretação da grande quantidade de dados produzido pelas ciências e pelos usuários dos serviços de Internet globais está sendo um dos maiores desafios para a sociedade atual [Bell et al., 2009]. Por tratar-se de um fenômeno recente em com alto potencial, diversas frentes de estudos vêm se formando. Nas ciências, há movimentação principalmente nas áreas biológicas, astronômicas, física e químicas. Na indústria, há grande esforço nas análises preditivas [Dhar, 2013]. No setor governamental, imensas bases de dados impulsionam um melhor funcionamento da máquina pública. Com isso, novas profissões especializadas na manipulação de modelos e interpretação dos dados surgiram, trazendo os métodos científicos para dentro da indústria.

A *Ciência de Dados*, surge exatamente nesse contexto, incorporando técnicas e teorias de diversos campos da ciência, principalmente a engenharia e ciências básicas, extremamente ligada a conceitos bem consolidados, formando assim, uma área interdisciplinar, como ilustrado na Figura 1.

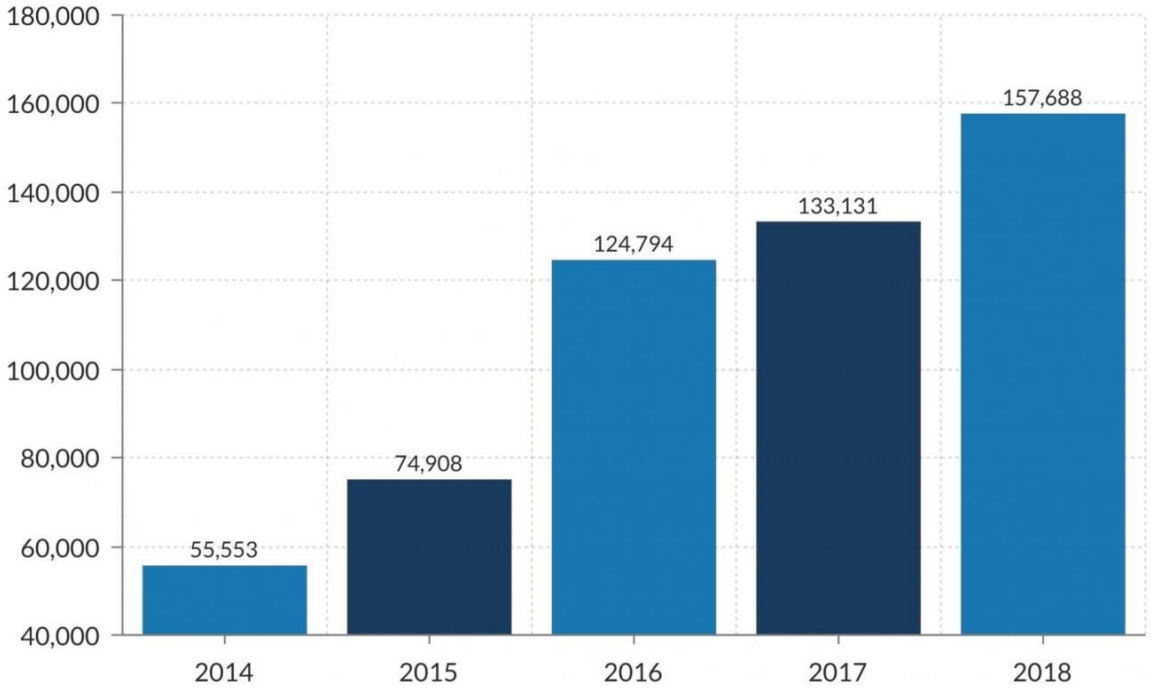
Figura 1 – Diagrama de Venn das competências da Ciência de Dados.

Fonte: Rautenberg, 2020.[1]
Disponível em < https://www.brapci.inf.br/index.php/res/download/112105/>.
Acesso em: 18 out. 2020.

Especificamente no mercado financeiro, um dos temas de grande investimento é a busca na prevenção de transações fraudulentas em cartões de crédito. Atualmente, o volume de pessoas que utilizam o cartão de crédito como forma de pagamento é significante. Com isso, o risco de operações fraudulentas das instituições também aumenta.

Segundo um levantamento realizado pela *Shift Processing,* em 2018 foram fraudados mais de $ 24.26 bilhões em todo o mundo. E esse é um valor que aumenta significativamente, como podemos ver na **Figura 2**, no caso específico para os Estados Unidos.

Figura 2 – Fraude com cartão de crédito nos EUA em 2018.



Fonte: Adaptado de Shift Processing, 2018.2
Disponível em <https://shiftprocessing.com/credit-card-fraud-statistics/#download>.
Acesso em: 20 out. 2020

Nesse contexto, juntamente com a Ciência de Dados, o uso de modelos preditivos para a detecção de uma transação fraudulenta tornou-se muito utilizado pelas instituições financeiras. Rotineiramente abordado como um problema de detecção de anomalias, a análise de fraudes em cartões se resume a um problema de classificação binária – fraude ou normal – permitindo a adoção de diversos modelos para solução. Esse trabalho em específico realizou uma análise da implementação de alguns modelos conceituados em *Machine Learning,* como Regressão Logística, K-Nearest Neighbors, Support Vector Machine e Árvore de Decisão.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral desse trabalho foi fazer uma análise do comportamento dos modelos de classificação Regressão Logística, K- Nearest Neighbors, Support Vector Machine e Árvore de Decisão, em uma base real de transações de cartão de crédito.

1.2.2 Objetivos Específicos

- Comparar o resultado finais dos modelos, baseando com indicadores chaves, como **quantidade de falso-positivos e falsos-negativos, acurácia, precisão, revocação e F1-score;**

- Evidenciar o melhor modelo preditivo para as condições empregadas;

- Evidenciar a importância da Ciência de Dados para a atualidade e seus usos.

2. REFERÊNCIA TEÓRICO

A seguir será apresentada uma revisão bibliográfica sobre fraudes em cartão de crédito e aprendizado de máquina e suas técnicas de modelagem. Através dessa revisão é possível se ter um entendimento sobre o assunto abordado neste trabalho, com a apresentação de conceitos fundamentais sobre o tema.

2.1 Fraudes em cartão de crédito

Por definição, fraude é: "O crime ou ofensa de deliberadamente enganar outros com o propósito de prejudicá-los usualmente para obter propriedade ou serviços dele ou dela injustamente" (DELMANTO, 2017). Essa definição enquadra-se satisfatoriamente ao mercado virtual, onde os cartões são a opção mais utilizada para pagamentos. Segundo o SPC Brasil, mais de 78% dos brasileiros utilizam o cartão de crédito como principal forma de pagamento online.

Com tal “mercado”, criminosos virtuais (e presenciais) especializam-se em fraudes, com uso de técnicas refinadas e softwares poderosos para obtenção de dados a fim de qualificar o furto. Em contrapartida, as instituições financeiras investem maciçamente em profissionais, métodos e modelos de aprendizado de máquina para evitar que essas operações ocorram, mitigando assim riscos e perdas aos seus clientes e a si mesmas.

Neste contexto, as 10 fraudes mais comuns na América Latina, segundo um estudo da VISA, realizado em 2017 são:

- **Fraude de apropriação de conta:** roubo de identidade em que o fraudador obtém acesso às contas bancárias ou ao cartão de pagamento da vítima – por meio da violação de dados ou do uso de malware ou phishing – utilizando as informações para fazer transações não autorizadas.

- **Fraude de afiliada:** atividade fraudulenta gerada por uma afiliada na tentativa de gerar receita ilegítima; por exemplo, afiliadas que induzem estabelecimentos comerciais a pagar comissões não devidas

- **Botnets**: rede privada de computadores infectados com um software malicioso. Esses computadores são controlados como um grupo, sem o conhecimento de seus proprietários, para, por exemplo, roubar dados, enviar spam e permitir que criminosos acessem dispositivos.

- **Teste de cartão:** quando fraudadores usam as lojas on-line para testar informações do cartão de pagamento que estão em seu poder. O objetivo é “testar” os cartões para descobrir se eles foram bloqueados/cancelados, e se os limites de crédito foram atingidos .

- **Fraude “limpa”:** utiliza informações roubadas do cartão de pagamento e, com grande quantidade de dados pessoais, os criminosos efetuam compras fazendo-se passar pelos verdadeiros portadores do cartão sem levantar suspeitas. Assim manipulam as transações para burlar as funcionalidades de detecção defraude .

- **Fraude “amigável”:** ocorre quando o consumidor faz uma compra on-line usando seu próprio cartão de pagamento e após receber o produto ou serviço, solicita o estorno ao emissor do cartão. Uma vez aprovado, o estorno cancela a transação de pagamento e o consumidor recebe de volta o montante gasto .

- **Roubo de identidade:** uso deliberado da identidade de outra pessoa, normalmente para obter vantagens financeiras, crédito e outros benefícios em seu nome .

- **Lavagem de dinheiro:** processo que oculta as origens de fundos obtidos ilegalmente, por transferências de recursos envolvendo bancos estrangeiros ou empresas legítimas. Isso faz com que fundos obtidos ilegalmente ou "dinheiro sujo" pareçam legais ou "limpos" .

- **Phishing/pharming/whaling**: são técnicas de engenharia social utilizadas para pessoas físicas ou jurídicas, ou para atraí-los para sites falsos na tentativa de obter informações como números de cartão de pagamento, senhas bancárias e outros dados .

- **Esquemas de triangulação**: criminosos usam cartões de pagamento roubados para comprar mercadorias arrematadas em leilões on-line ou adquiridas em sites de e-commerce. Em seguida, revendem essas mercadorias a clientes legítimos, que não estão envolvidos na fraude.

2.2 Relação entre fraude e o cenário atual da computação

Na era atual, a busca pelo enriquecimento rápido, principalmente entre os jovens, fez com que o ato fraudulento se tornar um grande aliado na temática de grandes ganhos com pouco esforço. Como relatado na seção anterior, diversas são as formas de fraude e para combatê-las usa-se o auxílio da computação moderna, com grande capacidade de processamentos dos dados. Porém, segundo Yaohoa e Mation (2018), na ótica computacional, o poderio de conhecimento fica limitado as instituições de forma particular, juntamente com sua tecnologia.

Visando aprimorar técnicas para o combate às fraudes e integrar com a tecnologia computacional atual, utiliza-se o *Aprendizado de Máquina*, ou *Machine Learning* (*ML*). Segundo Murphy (2012), ML pode ser definida como a área que pesquisa, propõe e estuda técnicas para que de forma automática, detectar padrões em situações pautadas em dados e utilizá-los para predizer acontecimentos futuros, ou auxiliar em tomada de decisão sobre eventos não determinísticos.

Através do Aprendizado de Máquina, pode-se, de forma resumida, manipular os dados a fim de conseguir uma *classificação*, *regressão* ou *agrupamento* entre eles. Para isso, segundo Marsland (2012), categoriza-se o ML em:

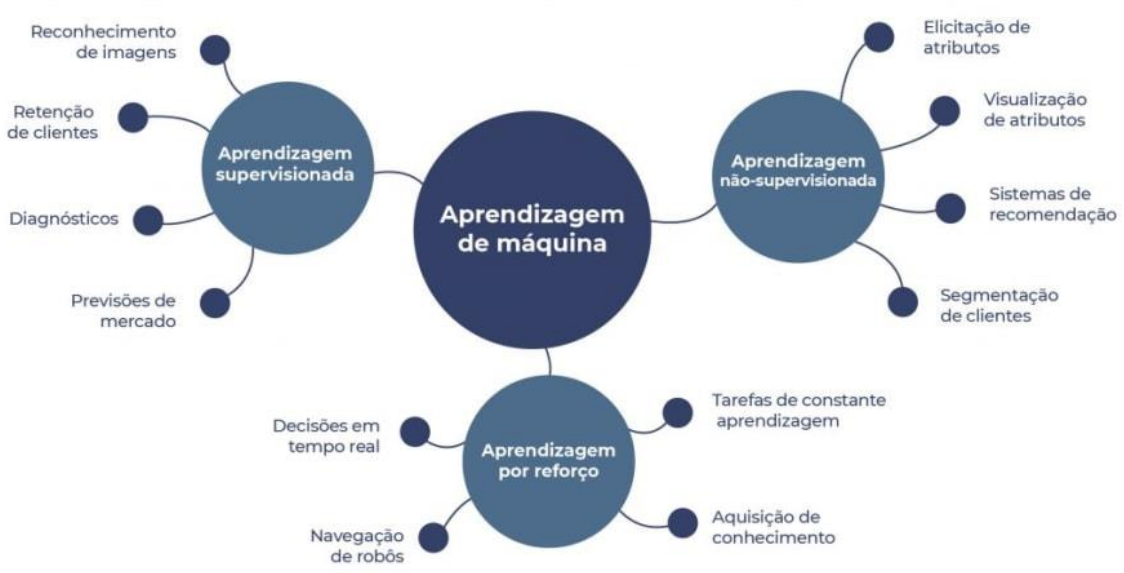
- **Aprendizado supervisionado:** No Aprendizado Supervisionado, um conjunto conhecido de dados é utilizado para treinamento do modelo. Com isso, baseado nesses exemplos, o algoritmo generaliza as respostas e aplica a generalização aos dados desconhecidos, com a finalidade de determinar corretamente sua classe. .

- **Aprendizado não supervisionado:** Não há fornecimento de conjunto de dados conhecidos ao modelo. Dessa forma, o algoritmo tenta identificar similaridades entre os dados para atribuir a mesma classe.

- **Aprendizado por reforço:** O algoritmo é beneficiado ou punido de acordo com a ação tomada, por forma de feedback. Assim, ele classifica se está correto ou não o procedimento e generalização adotada e com isso decide se altera ou não a estratégia.

Um exemplo das classificações e para qual finalidade são utilizados encontra-se na Figura 3.

Figura 3 – Tipos de ML e suas principais empregabilidades



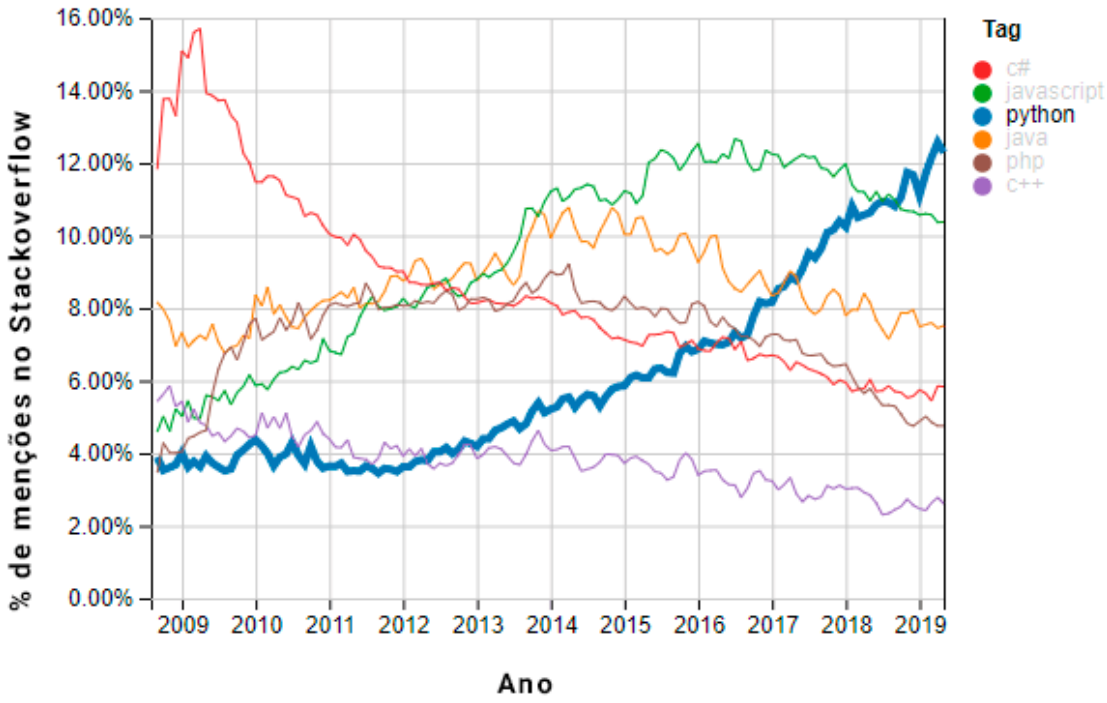
Fonte: Dev.to – Aprendizado de Máquina.1
[1] Disponível em <https://dev.to/beatrizmaiads/tipos-de-aprendizado-de-maquina-3-5d66>. Acesso em: 28 out. 2020

2.3 Python

Python é uma linguagem de programação de alto nível, criada por Guido Van Rossum, matemático holandês, nos inícios dos anos 90. Possui característica de ser dinâmica, interpretada, modular, multiplataforma e orientada a objetos.

Por possuir fácil sintaxe, grande performance e uma comunidade cooperativa, sua utilização e adeptos cresceu grandemente nos últimos anos, conforme pode ser observado na Figura 4, onde vemos a quantidade de menções no portal Stack Overflow (um dos principais sites relacionados a programação na atualidade) relativo as demais linguagens.

Figura 4 – Principais linguagens e a % de menções no site Stack Overflow



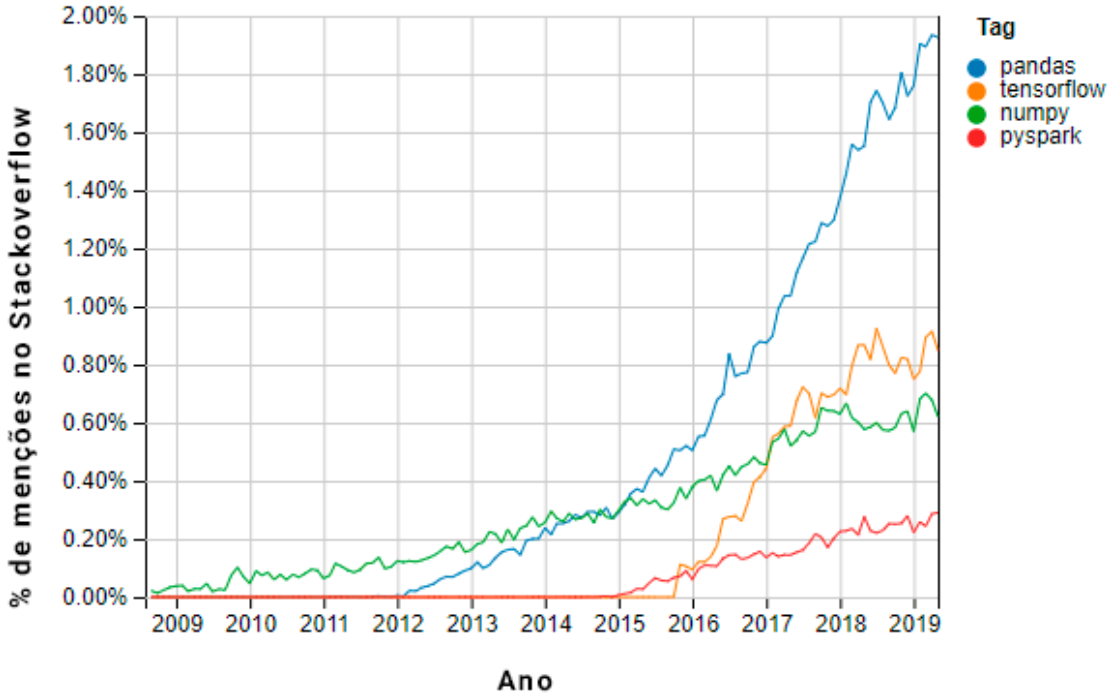
Fonte: Stack Overflow[1]

[1] Disponível em <https://harve.com.br/blog/programacao-python-blog/python-para-que-serve-top-5-utilidades/>

Acesso em: 28 jan 2021

Além disso, como dito anteriormente, é evidente que a comunidade adepta ao phyton é totalmente colaborativa. Pode-se citar que, o surgimento de novas bibliotecas que facilitam as manipulações dos dados (com diversas finalidades), ocorrem a todo momento. Na Figura 5, pode-se observar o crescimento das menções de quatro importantes bibliotecas no site Stack Overflow.

Figura 5 – Principais bibliotecas python e a % de menções no site Stack Overflow



Fonte: Stack Overflow[1]

[1] Disponível em <https://harve.com.br/blog/programacao-python-blog/python-para-que-serve-top-5-utilidades/>

Acesso em: 28 jan 2021

Entre as principais finalidades do uso de python, pode-se destacar:

- Data Science
- Visualização e análise de dados
- Desenvolvimento Web
- Desenvolvimento de aplicativos
- Automação de scripts
- Desenvolvimento de jogos
- Aplicações GUI
- Networking
- Testing
- Robotics
- Embedded aplications

Python será a linguagem utilizada nesse projeto, juntamente com o software Jupyter Nootebok.

2.4 Classificadores

Segundo Oliveira (2016), no âmbito de ML, um dos objetivos é organizar os dados em categorias, ou classes, previamente definidos. Dessa forma, há um *conjunto de classes*, um *conjunto de instâncias* e um *classificador*. Assim, a classificação é a ação, realizada pelo classificador, de atribuir uma classe a cada instância.

Com isso, para realizar uma classificação, é necessário construir um classificador, ou um método de classificação, que fará a função de agregar o dado a uma classe. Para a construção de um classificador, utiliza-se um conjunto de treinamento, ou seja, um conjunto no qual as amostras possuem as classes conhecidas. Após o treinamento, o classificador recebe amostras no qual as classes são desconhecidas para então prever a classe em que é o mais correto classificá-la.

No contexto de detecção de transações fraudulentas de cartão de crédito, o conjunto de classes é {transação fraudulenta; transação não fraudulenta}. O conjunto de amostras são as transações entre si. Já o conjunto de treinamento são as transações passadas no qual sabe-se a sua classe (fraudulenta ou não). O classificador então deve atribuir apenas uma classe a nova transação.

Os próximos assuntos abordados elucidam a base matemática e o funcionamento dos classificadores aqui utilizados.

2.5 Modelo Lineares

Os modelos classificadores relatados nesse projeto possuem, de alguma forma, a tarefa de regressão, na qual a função dá-se pela combinação linear da base de dados. Assim, pode-se definir a função como (SCIKIT-LEARN, 2020):

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \cdots + w_p x_p \tag{2.1}$$

onde, $\hat{y}$ é a função que estima valores, p os atributos, W o vetor com os pesos dos atributos e w_0 a interseção.

2.5.1 SVM – Máquina de vetores de suporte

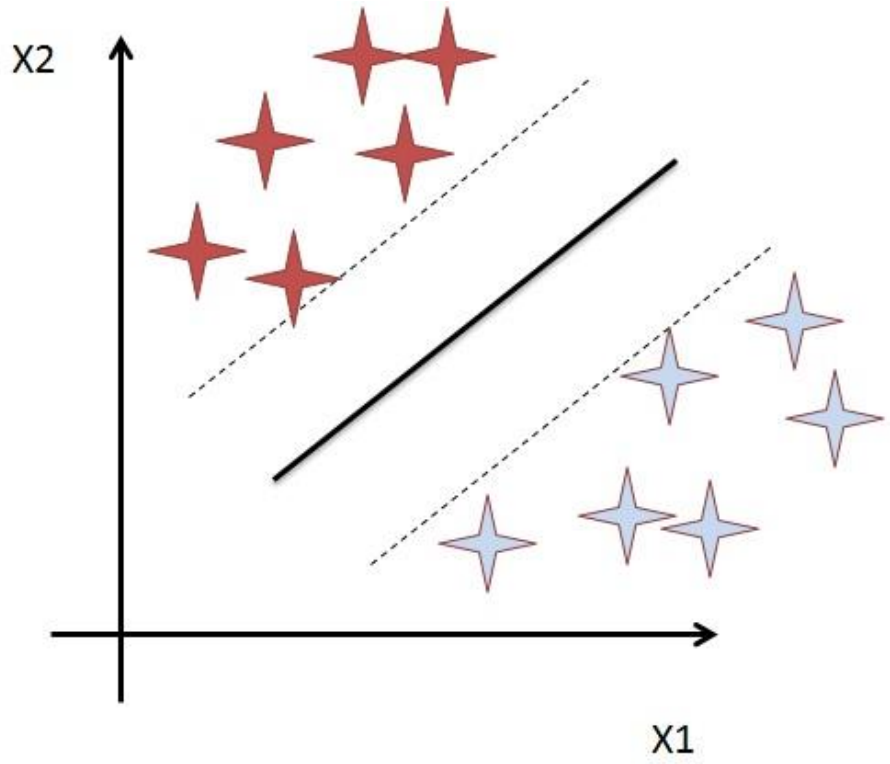
O modelo SVM (Support Vector Machine), ou Máquina de Vetores de Suporte, é uma metodologia de classificação proposta por Cortes e Vapnik em 1995, altamente utilizada, principalmente em situações de regressão e classificação (IZBICKI; SANTOS, 2020).

Simplificadamente, SVM é um algoritmo que cria a melhor separação entre os dados, através de um hiperplano (Figura 6). Para um sistema linear de h(x), tem-se (IZBICKI; SANTOS, 2020) a Equação 2.2:

$$h(\mathrm{x}) = \mathrm{w} \cdot \mathrm{x} + b \tag{2.2}$$

onde $\mathrm{w} \cdot \mathrm{x}$ é o produto escalar entre os vetores, e $\frac{b}{\|\mathbf{w}\|}$ é a distância do hiperplano à origem. Dessa forma, o modelo será ajustado para localizar um pequeno w para a Equação 2.2.

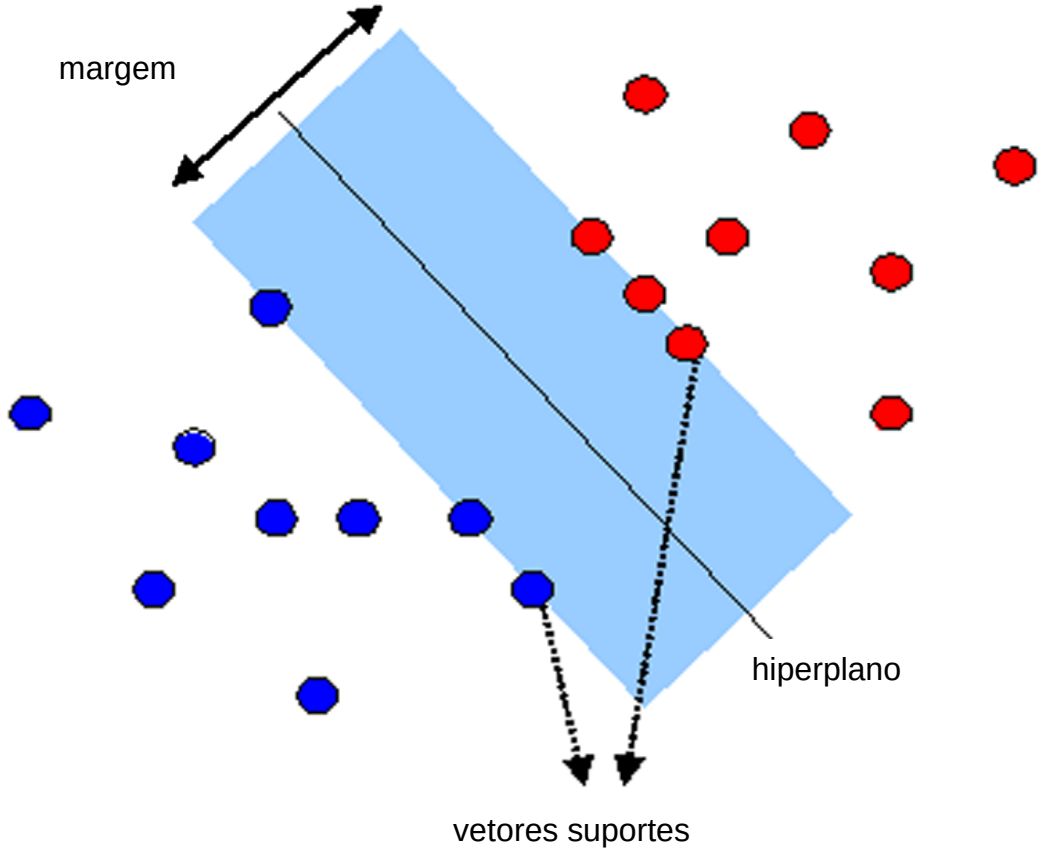
Figura 6 – Representação de um hiperplano para duas classes



Fonte: Minerando dados[1]
[1] Disponível em <https://minerandodados.com.br/famoso-svm/>.
Acesso em: 28 jan 2021

Dessa forma, o algoritmo otimiza o melhor hiperplano. Para isso, calcula-se os vetores de suporte, que são os pontos mais próximos dos hiperplanos. A distância entre esses vetores, são chamados de *Margens,* conforme a Figura 7.

Figura 7 – Representação de um hiperplano para duas classes, margem e os vetores de suporte.



Fonte: Adaptado de Minerando dados[1]
[1] Disponível em <https://minerandodados.com.br/famoso-svm/>.
Acesso em: 28 jan 2021

Para buscar soluções mais complexas, usa-se o artifício das funções de kernel (IZBICKI; SANTOS, 2020). Essas funções, possuem a capacidade de projetar vetores para espaços não linearmente separáveis.

2.5.2 Árvores de decisão

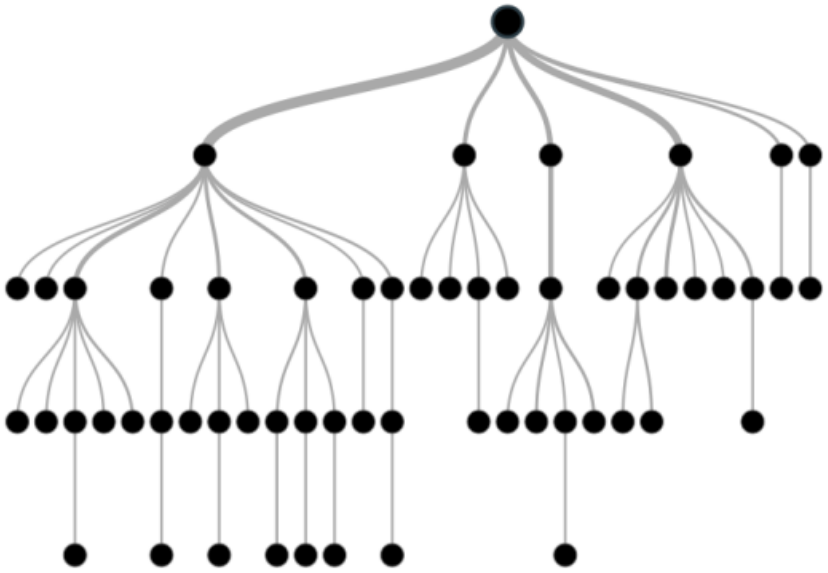
O modelo Random Forest, ou Florestas Aleatórias, é composto por um conglomerado de Árvore de Decisão, que são oriundas de vetores aleatórios. Em suma, as Florestas Aleatórias usam a classificação para determinar uma amostra, conforme a definição a definição de Breinan (2001), "*uma floresta aleatória é um classificador que consiste em uma coleção de árvores estruturadas* $\{h(x, \Theta k), k =$

1, …}, *onde* $\{\Theta k\}$ *são vetores aleatórios independentes distribuídos de forma idêntica e cada árvores emite um voto unitário para a classe mais popular de uma entrada x."*

Dessa forma, é de grande valia elucidar o conceito de Árvores de Decisão (Decision Trees).

Árvore de Decisão é um tipo de algoritmo utilizado em classificação e regressão, funcionando para variáveis categóricas e contínuas de entrada e saída. Dessa forma, divide-se a população ou amostra em subpopulações com base nos diferenciadores mais significativos das variáveis de entrada, conforme a Figura 8 ilustra a esquemática do modelo.

Figura 8 – Representação esquemática de uma floresta aleatória composta por árvores de decisão.

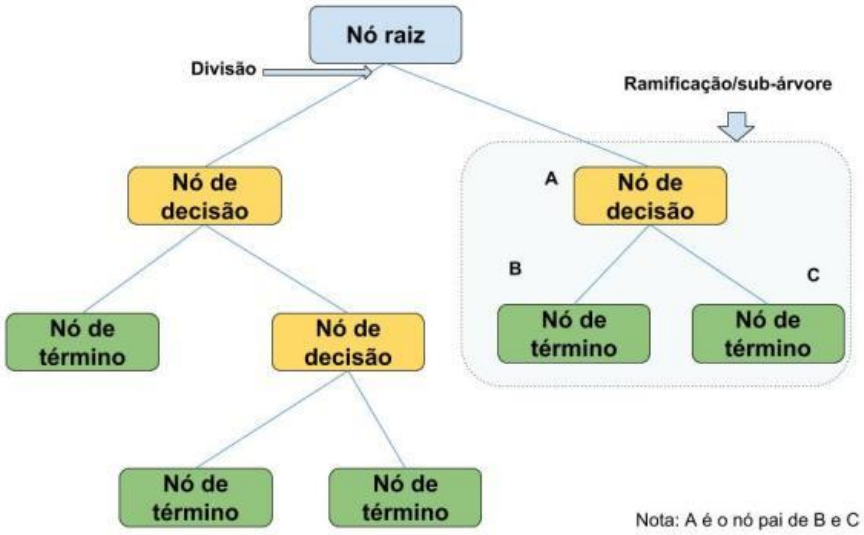


Fonte: Vooo – Insights
[1] Disponível em <https://www.vooo.pro/insights/um-tutorial-completo-sobre-a-modelagem-baseada-em-tree-arvore-do-zero-em-r-python/>.
Acesso em: 28 jan 2021

Segundo Izbicki e Santos (2020), as terminologias relacionadas às árvores de decisão são elencadas abaixo e podem ser observadas na esquematização demonstrada na Figura 9:

- **Nó Raiz**: Representa a população inteira ou amostra, sendo ainda dividido em dois ou mais conjuntos homogêneos.
- **Divisão:** É o processo de dividir um nó em dois ou mais sub-nós.
- **Nó de Decisão:** Quando um sub-nó é dividido em sub-nós adicionais.
- **Folha ou Nó de Término**: Os nós não divididos são chamados Folha ou Nó de Término.
- **Poda**: O processo de remover sub-nós de um nó de decisão é chamado poda. Podemos dizer que é o processo oposto ao de divisão
- **Ramificação/Sub-Árvore**: Uma subseção da árvore inteira é chamada de ramificação ou sub-árvore
- **Nó pai e nó filho**: Um nó que é dividido em sub-nós é chamado de nó pai. Os sub-nós são os nós filhos do nó pai.

Figura 9 – Representação esquemática de uma floresta aleatória composta por árvores de decisão e sua terminologia.



Fonte: Vooo – Insights
[1] Disponível em <https://www.vooo.pro/insights/um-tutorial-completo-sobre-a-modelagem-baseada-em-tree-arvore-do-zero-em-r-python/>.
Acesso em: 28 jan 2021

Em suma, cada árvore de decisão de uma floresta recepta a mesma amostragem e então classifica-a. Após, ocorre uma votação e então a classe majoritária entre as diversas árvores será à decisão da floresta.

2.5.3 Regressão logística

A regressão logística no âmbito de ML, é similar a regressão linear, porém para situações de classificação, onde teremos uma resposta binária ao final do processo (1 ou 0, sim ou não) (IZBICKI; SANTOS, 2020). Dessa forma, utiliza-se a função logística (sigmóide) com o intuito de achatar a função original (Equação 2.1):

$$\text{logit}^{-1}(\alpha) = \frac{1}{1+e^{-\alpha}} = \frac{e^{\alpha}}{1+e^{\alpha}} \tag{2.3}$$

Dessa forma, em um cenário de regressão logística binária, e igualando a Equação 2.1, chega-se a (MESQUITA, 2014):

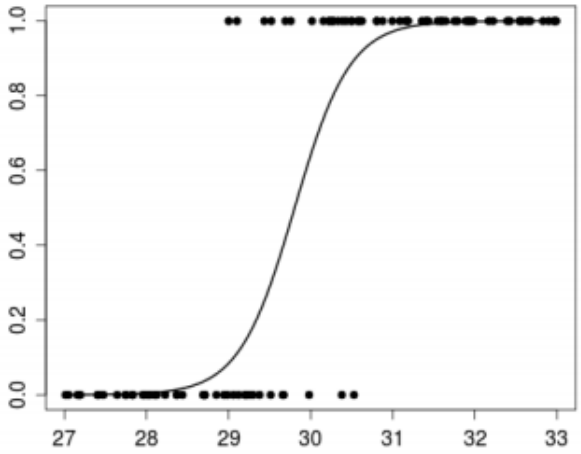
$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 \tag{2.4}$$

O intuito do modelo logístico é estimar p. Isolando-o, após algumas passagens matemáticas, chega-se então a (MESQUITA, 2014) .

$$\hat{p} = \frac{e^{\beta_0+\beta_1 x_1}}{1+e^{\beta_0+\beta_1 x_1}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1)}} \tag{2.5}$$

A equação 2.5, conhecida como “Equação de regressão estimada”, é a função que ilustra o objetivo desse modelo, pois $\hat{p}$ é a probabilidade estimada de qualquer coeficiente e variável a ela aplicada, como observado na Figura 10, onde mostra-se os pontos de dispersão da relação de variável dependente e independente da regressão logística (GONZALEZ, 2018).

Figura 10 – Pontos de dispersão fictícios da relação de variável dependente e independente da regressão logística.

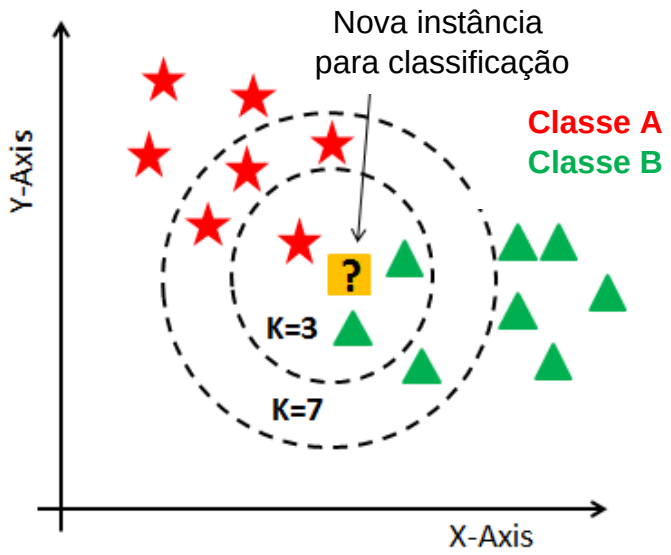


Fonte: (GONZALEZ, 2018)

2.6 K-Nearest Neighbors

O classificador K- Nearest Neighbors, ou KN-N (KNN), ou modelo de vizinho mais próximo, é um modelo baseado na associação das características da vizinhança da amostra. Segundo Izbicki e Santos (2020), o modelo KNN busca a classe mais frequente entre as observações mais próximas ao vetor de covariáveis x de interesse, como demonstrado na Figura 11:

Figura 11 – Classificação pelo modelo KNN



Fonte: Adaptado de Inferir[1]

[1] Disponível em <https://inferir.com.br/artigos/algoritimo-knn-para-classificacao/ >. Acesso em: 12 fev. 2021

Na Figura 11, há um dado não classificado (?) diante de duas classes bem definidas (classe A em vermelho e classe B em verde). Então, define a quantidade de vizinhos (k-vizinhos) que serão utilizados para classificar o novo dado. Se k=3, calcula-se a distância euclidiana através da Equação 2.6 (IZBICKI; SANTOS, 2020) do novo dado com os 3 dados vizinhos mais próximos (primeiro círculo). Se k=7, calcula-se a distância euclidiana novo dado com os 7 dados vizinhos mais próximos e assim sucessivamente. Dessa forma, é possível classificar qual a classe predominante para a situação e então classificar o novo dado.

$$\text{dist}\,(A,B) = \sqrt{\frac{\sum_{i=1}^{m}\left(x_i - y_j\right)^2}{m}}\,, \tag{2.6}$$

onde A e B são os vetores de características e *m* a dimensionalidade do espaço.

2.7 Validação Cruzada

Na realidade de ML, a validação cruzada constitui-se em um método de re-amostragem utilizado para avaliar o modelo evitando que ocorra o teste no mesmo conjunto de dados no qual ele foi treinado, a fim de evitar diversos erros na modelagem. Há vários métodos de validação cruzada, como o de saída única, holdout e k-fold (IZBICKI; SANTOS, 2020).

2.8 Métricas de avaliação de desempenho

Interessado em mensurar o desempenho dos modelos, as métricas de avaliação servem como balizador imprescindível nesse contexto. Para melhor entendimento, deve-se elencar alguns conceitos, como o de matriz de confusão.

Sabe-se que os classificadores aqui mencionados, geram saídas binárias: SIM, para transações fraudulentas e NÃO para transações não fraudulentas. Entretanto, pode-se classificar as saídas como demonstrado na Tabela 1:

Tabela 1 – Classificação das classes saídas dos modelos abordados.

VP - Verdadeiro Positivo	**amostras classificadas corretamente como fraude. Nesse caso, o modelo retorna a classe como fraude e realmente é uma transação fraudulenta**
VN - Verdadeiro Negativo	amostras classificadas corretamente como não fraude. Nesse caso, o modelo retorna a classe como não fraude e realmente não é uma transação fraudulenta
FP - Falso Positivo	amostras classificadas de forma errada fraude. Nesse caso, o modelo retorna a classe como fraude e não é uma transação fraudulenta
FN - Falso Negativo	amostras classificadas de forma errada como não fraude. Nesse caso, o modelo retorna a classe como não fraude, porém é uma transação fraudulenta

Fonte: (SCIKIT-LEARN, 2021)

Com os conceitos elucidados, pode-se então descrever as métricas de avaliação de desempenho.

2.8.1 Acurácia

Segundo Izbicki e Santos (2020), a Acurácia (Accuracy) pode ser entendida como a métrica que indica a performance geral do modelo. De todo o conjunto de dados classificadas, quantos foram classificadas corretamente, conforme a Equação 2.7. Em resumo, é a quantidade classificações verdadeira positivo e verdadeira negativa, sobre o conjunto total.

$$Acurácia = \frac{VP+VN}{VP+VN+FP+FN} \tag{2.7}$$

2.8.2 Precisão

Precisão (Precision), segundo Izbicki e Santos (2020), é a capacidade do modelo em não classificar em verdadeiro positivo, uma amostra negativa, conforme a Equação 2.8.

$$Precisão = \frac{VP}{VP+FP} \tag{2.8}$$

2.8.3 Revocação

A revocação, também conhecida como *recall score*, é a capacidade do modelo classificar todas as amostras positivas, dada pela Equação 2.9 (IZBICKI; SANTOS, 2020).

$$Revocação = \frac{VP}{VP+FN} \tag{2.9}$$

2.8.4 F1 – Score

A métrica F1 – Score, pode ser considerada como um complemento as demais já apresentadas. Em suma, é a média harmônica entre as métricas Precisão e Revocação. Com ela, há a capacidade de observar somente uma métrica e sua combinação ao invés de duas. Normalmente, quando se tem um F1 – Score baixo, é um sinal de que a precisão ou revocação está também com baixos níveis, conforme a Equação 2.10.

$$F1 - Score = \frac{2x\ (\text{precisão}\ x\ \text{revocação}\ )}{\text{precisão + revocação}} \tag{2.10}$$

2.8.5 *Área sob a curva ROC*

Segundo Oliveira (2016), o termo ROC – Receiver Operating Characteristic, é utilizado para apurar a taxa de acertos perante a taxa de falsos alarmes. Dessa maneira, em nosso contexto, a curva ROC é um gráfico onde se tem índice de verdadeiros positivos vs índice de falsos positivos, sendo então considerada mais uma métrica de qualidade do classificador. Com isso, quando maior a área da curva (AROC), melhor o desempenho, conforme observado na Tabela 2.

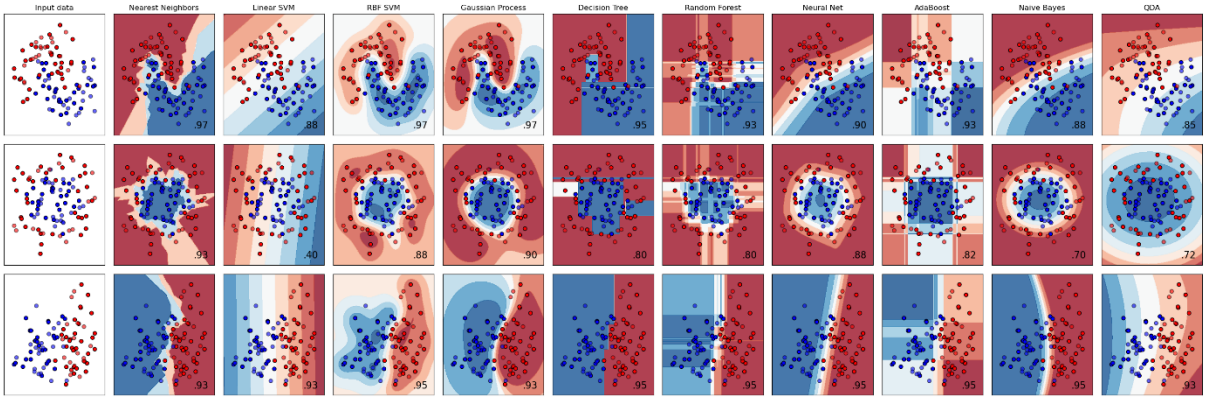
Tabela 2 – Níveis de classificação gerado pela área da curva ROC

Valor AROC	Poder de Classificação
AROC=0,5	Não há
0,7<=AROC<0,8	Aceitável
0,8<=AROC<0,9	Muito bom
AROC>=0,9	Excelente

Fonte: Oliveira (2016)

Vale ressaltar que há diversas outras métricas de avaliação de performance de um modelo classificador, assim também como outras técnicas de classificação. Bem como a combinação entre elas, gerando novos classificadores, como pode ser observado na Figura 12. Entretanto, nas próximas seções desse projeto, apenas os classificadores e métricas de performance aqui mencionados serão utilizados.

Figura 12 – Diversos modelos de classificação e como os dados comportam-se.

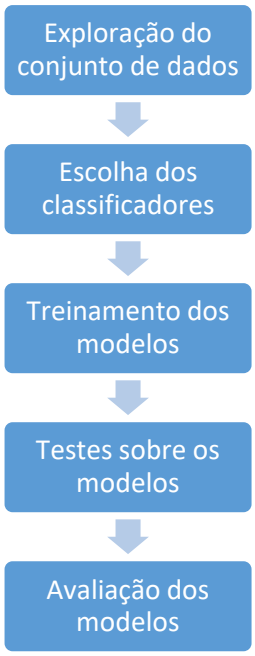


Fonte: (SCIKIT-LEARN, 2021)

3. METODOLOGIA

A metodologia utilizada nesse projeto é comumente utilizada em diversos projetos dentro da comunidade de Ciência de Dados, conforme pode ser observado na Figura 13.

Figura 13 – Metodologia de trabalho em um projeto de ML.



Fonte: Autor

Dessa forma, a primeira etapa será a exploração do conjunto de dados no qual os modelos serão testados e treinados. Após o entendimento inicial dos dados, e com os classificadores escolhidos, a próxima etapa dá-se pelo treinamento dos modelos aqui analisados, seguindo então para a etapa de testes e avaliação deles.

A primeira etapa constitui-se em entender os dados que irão ser utilizadas, conforme explicado no próximo tópico.

3.1 Exploração do conjunto de dados

Nessa etapa, há a exploração do conjunto de dados a fim de entender o que está sendo manipulados. Com isso, alguns erros comuns são evitados.

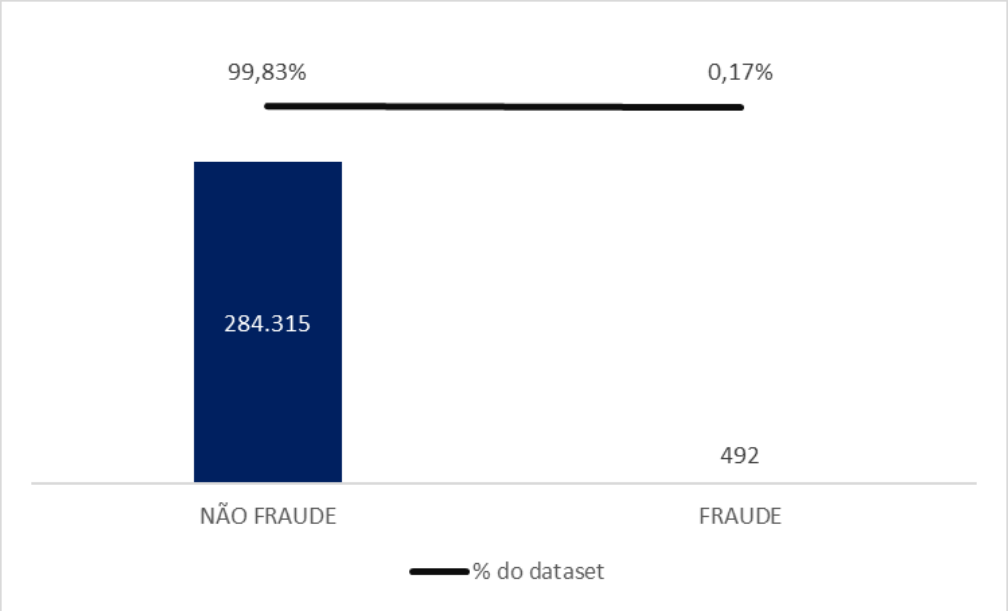
O *dataset* utilizado, é um conjunto de dados reais de transações de cartão de crédito realizadas em dois dias em *setembro de 2013*. Tal *dataset* está disponibilizado de forma gratuita na internet em diversas plataformas, como OpenML[1] e Kaggle na forma de arquivo csv, com o intuito de fomentar estudo da comunidade ML sob dados reais.

Além disso, sabe-se que foi aplicado a técnica PCA no *dataset.* Segundo, Izbicki e Santos (2020), PCA (Principal Componet Analysis) é técnica na qual busca-se por variáveis que sejam combinações lineares das covariáveis originais. Com isso, diminui-se drasticamente o conjunto de dados finais, não alterando significativamente as características do conjunto inicial de dados e otimizando a análise de dados.

Outra informação relevante é que se tem 28 classes (V1, V2, V3...V28) nas quais os dados foram omitidos por privacidade aos clientes (oriundos do PCA). As classes disponíveis e que serão aqui utilizadas são Time, Amount e Class (1 para fraude e 0 para não fraude). Destaca-se também que houve normalização dos dados.

Em relação a quantidade de transações fraudulentas perante as não fraudentas há grande diferença, conforme demonstrado na Figura 14.

Figura 14 – Gráfico da quantidade de transações fraudulentas e não fraudulentas, e sua representatividade (%) perante o dataset



Fonte: Autor

[1] disponível em < https://www.openml.org/d/1597> Acesso em 12 mar. 2021.

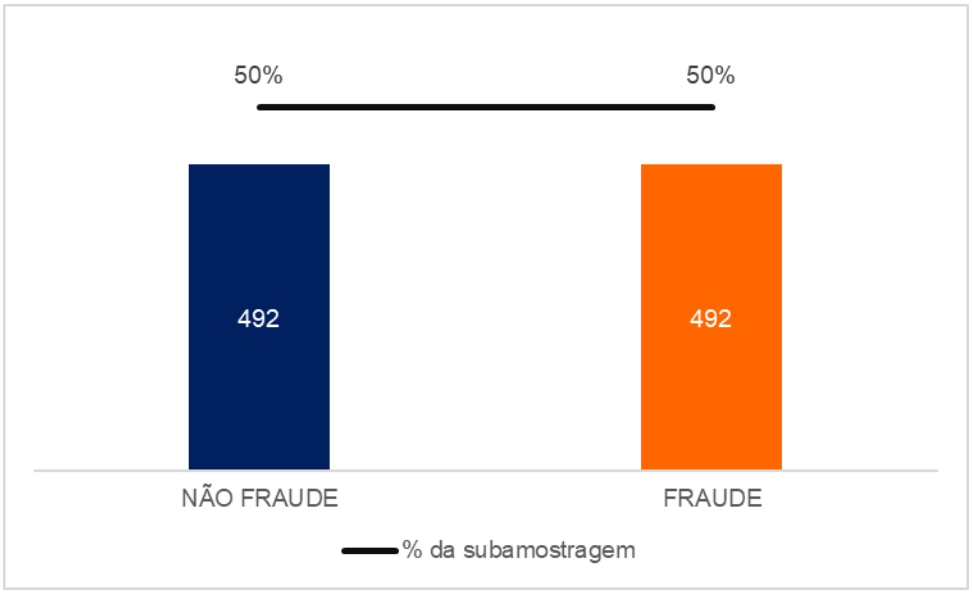
Como pode ser observado, há muito mais dados não fraudulentos do que fraudulentos, caracterizando um *dataset desbalanceado*. Conforme Oliveira (2016), um dataset é perfeitamente balanceado quando cada x classe possui 100/x% das instâncias do conjunto. Caso a representatividade de uma ou mais classe diferir sensivelmente das demais, tem-se um conjunto de dados desbalanceados. Tal situação comum para o caso de fraude em cartões de crédito, por exemplo.

A consequência de utilizar o conjunto de dados desbalanceados é que os classificadores, nesse caso, irão reconhecer que a maioria dos dados são do tipo de transações não fraudulentas erroneamente e não irão detectar padrões de indícios de fraude.

Há diversas técnicas para contornar essas desproporções. Em suma, busca-se sempre obter uma quantidade de 50% de dados positivos (nesse caso transações fraudulentas) e 50% de dados negativos (nesse caso transações não fraudulentas) para aplicar aos classificadores. Nesse projeto, utilizaremos as técnicas Random Undersampling (RU) e SMOTE.

Random Undersampling, ou Subamostragem aleatória, condiz em realizar uma subamostragem aleatoriamente de instâncias da classe dominante até atingir o balanceamento pretendido (NICOLA; LAURETTO; DELGADO, 2019). Para essa situação, um subamostragem de transações não fraudulentas será realizada até igualar a quantidade de transações fraudulentas. Uma desvantagem nítida para esse caso é que uma maciça quantidade de transações será negligenciada no nosso modelo com essa técnica, conforme pode ser observado na Figura 14.

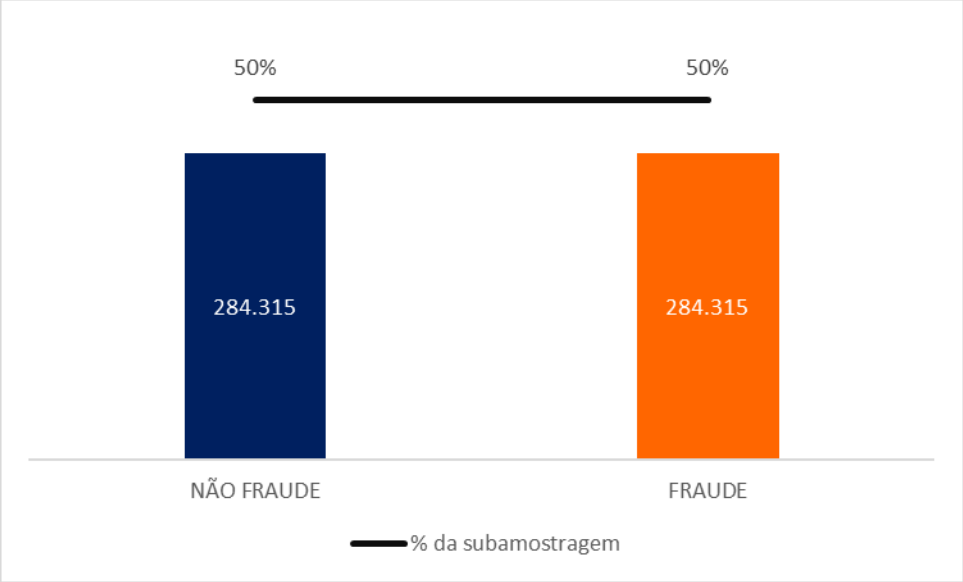
Figura 15 – Gráfico da quantidade de transações fraudulentas e não fraudulentas, e sua representatividade (%) perante a subamostragem via Random Undersampling



Fonte: Autor

A técnica SMOTE, ou Synthetic Minority Oversampling Technique, consiste em criar dados sintéticos da instância minoritária, aumentando assim sua quantidade até ao ponto de balancear o novo conjunto de dados (NICOLA; LAURETTO; DELGADO, 2019). Em suma, utiliza do artifício do classificador KNN, elucidado na seção anterior. Sendo assim, novos dados, semelhantes aos originais, serão criados da instância de transações fraudulentas até equilibrar a quantidade de transações não fraudulentas. A desvantagem dessa técnica consiste em, mesmo sendo similares, não consiste em dados totalmente reais. O novo dataset pode ser observado na Figura 15.

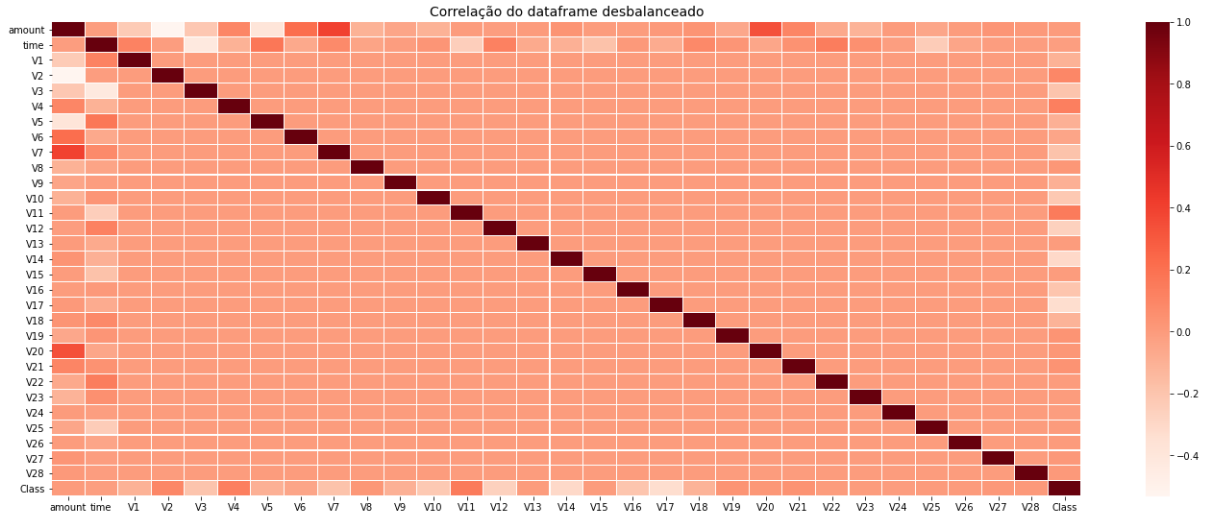
Figura 16 – Gráfico da quantidade de transações fraudulentas e não fraudulentas, e sua representatividade (%) perante a subamostragem via SMOTE



Fonte: Autor

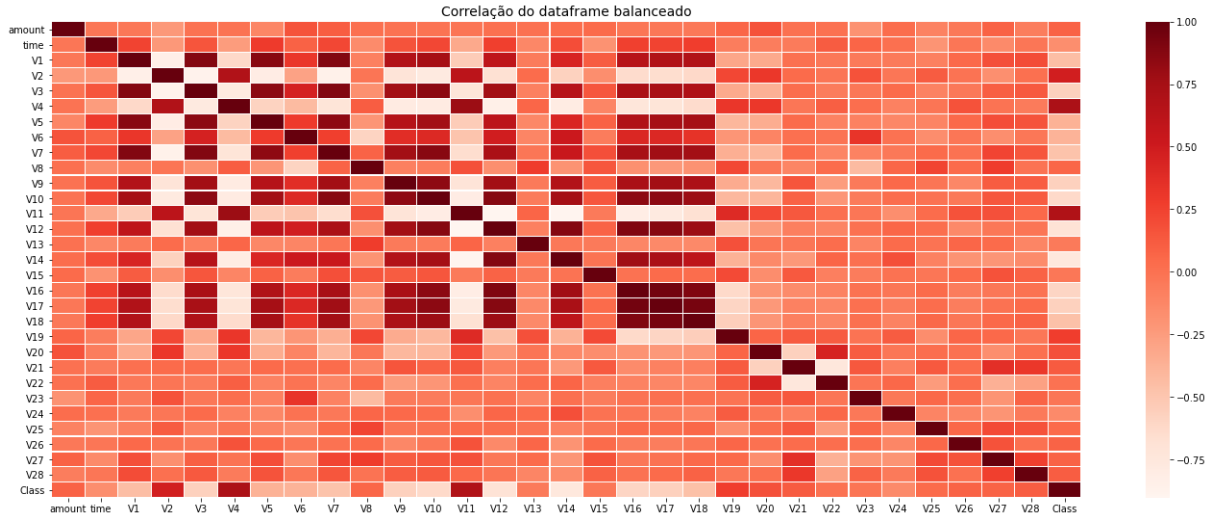
De maneira simplificada, a fim de exemplificação, pode-se observar na Figura 16 a correlação dos atributos no dataframe desbalanceados, onde há pouca correlação, evidenciado pela ausência de cores mais quentes (exceto pela diagonal principal, que sempre terá o valor de um, pois correlaciona o atributo com ele mesmo). Em contrapartida, na Figura 17, onde há o dataframe balanceado (via Random Undersampling) observa-se a grande presença de correlação, nitidamente visto pela quantidade de cores mais quentes na matriz de correlação.

Figura 17 – Matriz de correlação do dataframe original (desbalanceado)



Fonte: Autor

Figura 18 – Matriz de correlação do dataframe balanceado (via Random Undersampling)



Fonte: Autor

3.1.4 Hiperparâmetros

Hiperparâmetro, podem ser definidos como os parâmetros que não são aprendidos diretamente nos modelos (SCIKIT-LEARN, 2020). Dessa forma, estimar os melhores hiperparâmetro é de fundamental importância para a qualidade do produto da modelagem. Ressalta-se que a busca pelos melhores deles não é o objetivo desse projeto e assim sendo, utilizou-se o módulo GridSearchCV, presente

na biblioteca Scikit Learn. Esse módulo auxilia na escolha dos hiperpârametros de forma automatizada, no qual realiza-se todas as combinações possíveis a partir de um subconjunto de dados fornecidos. Após as combinações e testes, o módulo armazena em um único objeto os melhores hiperparâmetros para o algoritmo. (SCIKIT-LEARN, 2020). Por exemplo, para um fictício hiperparâmetro X, fornecemos um range com os possíveis valores inteiros (0,1,2,3,4,5), e o módulo dará, para aquele tipo de algoritmo e dados, qual o melhor valor inteiro entre 0 e 5.

Diversos são os hiperparâmetros para os classificadores aqui estudados. Será utilizado o módulo GridSearchCV para apenas alguns deles, conforme a Tabela 3, os demais utilizaram os valores padrões das bibliotecas.

Tabela 3 – Modelos e hiperparâmetros estudos e suas definições

Classificador	Hiperparâmetro	Definição
Regressão Logística	C	Parâmetro de controle da intensidade da regularização. C alto pode ocasionar Overfit, C baixo pode ocasionar Underfit
	Penalty	Parâmetro de minimização da complexidade do modelo.
KNN	n_neighbors	Parâmetro da quantidade de vizinhos a ser considerado para classificação
	algorithm	Parâmetro do estilo de algoritmo a realizar a classificação dos vizinhos próximos
SVM	C	Parâmetro de controle da intesidade da regularização. C alto pode ocasionar Overfit, C baixo pode ocasionar Underfit
	kernel	Parâmetro do tipo de kernel a ser implementado na classificação
Árvore de decisão	criterion	Parâmetro para medir a qualidade de uma divisão.
	max_depth	Parâmetro da profundidade máxima da árvore
	min_samples_leaf	Parâmetro de quantidade mínima de amostras necessárias para dividir um nó interno

Fonte: (SCIKIT-LEARN, 2020)

Diante dos hiperparâmetros apresentados na Tabela 3, a Tabela 4 traz os possíveis valores/métodos para escolha pelo GridSearchCV.

Tabela 4 – Possíveis valores para os hiperparâmetros

Classificador	Hiperparâmetro	Valor
Regressão Logística	C	0.001, 0.01, 0.1, 1, 10, 100, 1000
	Penalty	'l1', 'l2'
KNN	n_neighbors	2,3,4
	algorithm	'auto', 'ball_tree', 'kd_tree', 'brute'
SVM	C	0.5, 0.7, 0.9, 1
	kernel	'rbf', 'poly', 'sigmoid', 'linear'
Árvore de decisão	criterion	'gini', 'entropy'
	max_depth	2,3
	min_samples_leaf	5,6

Fonte: Autor

3.2 Treinamento, teste e avaliação dos modelos

Os algoritmos classificadores utilizados nesse contexto (Regressão Logística, KNN, SVM e Floresta Aleatória) possuem similaridade em relação ao fluxo do dado quando aplicada a biblioteca utilizada nesse projeto (Scikit-learn).

No início, o treinamento ocorre ao atribuir-se duas matrizes, X e Y. A matriz X recebe os dados de treinamento do modelo, tendo esse a dimensão f (quantidade de variáveis) por g (quantidade de atributos). A matriz Y, por sua vez, recebe os valores esperados da saída do conjunto treino, com dimensão f (quantidade de variáveis) por 1. Analogamente, as matrizes de teste possuem as mesmas características das matrizes de treinamento.

A próxima etapa, dá-se pela predição do modelo. Para isso, invoca-se o classificador desejado, que em linguagem python, têm-se: *LogisticRegression(),*

KNeighborsClassifier(), DecisionTreeClassifier(), SVC(), conforme observado na Figura 19.

Figura 19 – Invocação dos classificadores em Python.

```
#Classificadores
classifiers = {
    'LogisticRegression': LogisticRegression(),
    'KNearest': KNeighborsClassifier(),
    'SVC': SVC(),
    'DecisionTreeClassifier': DecisionTreeClassifier()
}
```

Fonte: Autor

Após os classificadores já chamados, usa-se o método *fit(X_treino,Y_treino)* para realizar o treinamento de cada classificador. Com isso, após o treinamento, o modelo já se torna apto para realizar as predições na fase de teste (SOUZA, 2019).

Por consequência, as predições na fase de teste ocorrem através do método *predict(X_teste)*. Para isso, apenas a matriz de teste é fornecida ao método. Com isso, o output do método predict será uma matriz com as predições.

Ressalta-se que os dados possuem a proporção 70/30 entre as matrizes de treinos e testes.

Para avaliar a performance dos modelos aqui estudados, serão utilizadas as métricas apresentadas anteriormente.

4. RESULTADO E DISCUSSÃO

Na realização da modelagem dos classificadores desse projeto, utilizou-se um computador com as seguintes configurações: processador Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz, memória RAM 8,00 GB (utilizável: 7,82 GB), tipo de Sistema operacional de 64 bits, processador baseado em x64 com sistema operacional Windows 10 Home Single Language versão 21H1. O primeiro quesito analisado são os hiperparâmetros, conforme demonstrado no próximo tópico.

4.1 Escolha dos hiperparâmetros

Conforme explanado na seção 3.1.4, utilizou-se o módulo GridSearchCV, para selecionar os melhores hiperparâmetros para os modelos de classificação. Ressalta-se ainda que, os demais hiperparâmetros mantiveram-se com os seus valores padrões. O quesito escolhido para escolha dos melhores valores foi a acurácia da melhor média *(Mean_test_score).*

4.1.1 Hiperparâmetros - Regressão Logística

Com o intuito de selecionar o melhor valor para C, a busca ocorreu entre os valores 0.001, 0.01, 0.1, 1, 10, 100, 1000. Para o hiperparâmetro de Penalidade, a busca restringiu-se em l1 e l2. Conforme observado na Tabela 4, a melhor combinação para C, com o dataframe RU, foi C=0.1 e Penalty= l2, com a mean_test_score = 0.958817. Para o dataframe SMOTE, obteve-se C= 0.001 e Penalty= l2, com a mean_test_score = 0.946783.

Tabela 5 – Hiperparâmetros para Regressão Logística.

Dataframe	Hiperparâmetro	Valor	Mean_test_score
Regressão Logística - RU	C	0.1	0.958917
	Penalty	l2	
Regressão Logística - SMOTE	C	0.001	0.946383
	Penalty	l2	

Fonte: Autor

4.1.2 Hiperparâmetros – KNN

Para selecionar os melhores valores para o classificador KNN, o conjunto de possíveis valores para n_neighbors limitou-se em 2, 3 e 4. Para o hiperparâmetro algorithm, os possíveis modos foram 'auto', 'ball_tree', 'kd_tree', 'brute'. A melhor combinação para o dataframe RU encontrada foi n_neighbors = 3 e algorithm = ‘auto’ com mean_test_score = 0.945167. Para o dataframe SMOTE, obteve-se a melhor combinação para n_neighbors = 3 e algorithm = ‘auto’ com mean_test_score = 0.978659.

Tabela 6 – Hiperparâmetros para KNN

Dataframe	Hiperparâmetro	Valor	mean_test_score
KNN - RU	n_neighbors	3	0.945167
	algorithm	auto	
KNN - SMOTE	n_neighbors	3	0.978659
	algorithm	auto	

Fonte: Autor

4.1.3 Hiperparâmetros – SVM

Buscando o melhor valor para C, os possíveis valores limitaram-se em 0.5, 0.7, 0.9, 1. Para o kernel, os possíveis métodos foram 'rbf', 'poly', 'sigmoid', 'linear'. A melhor combinação obtida para o dataframe RU, foi C= 1 e kernel = ‘linear’ com mean_test_score = 0.954826. Para o dataframe SMOTE, obteve-se C= 0.9 e kernel = ‘linear’ com mean_test_score = 0.900323, conforme a Tabela 6.

Tabela 7 – Hiperparâmetros para SVM

Dataframe	Hiperparâmetro	Valor	mean_test_score
SVM - RU	C	1	0.954826
	kernel	linear	
SVM - SMOTE	C	1	0.900323
	kernel	linear	

Fonte: Autor

4.1.4 Hiperparâmetros – Árvores de Decisão

Para a seleção do hiperparâmetro criterion, a busca limitou-se nos modos ‘giny’ e ‘entropy’. Para max_depth, os possíveis valores foram 2 e 3, assim como para min_samples_leaf foram 5 e 6. No dataframe RU, a melhor combinação obtida foi criterion = entropy, max_depth = 3 e min_samples_leaf = 6, com mean_test_score = 0.936247. No conjunto de dados SMOTE, obteve-se criterion = entropy, max_depth = 3 e min_samples_leaf = 5, com mean_test_score = 0.922940, conforme pode ser observado na Tabela 7.

Tabela 8 – Hiperparâmetros para Árvores de decisão

Dataframe	Hiperparâmetro	Valor	mean_test_score
Árvores de decisão - RU	criterion	entropy	0.936247
	max_depth	3	
	min_samples_leaf	6	
Árvores de decisão - SMOTE	criterion	entropy	0.922940
	max_depth	3	
	min_samples_leaf	6	

Fonte: Autor

Com os hiperparâmetros de melhor performance definidos, resta-se elencar a performance dos modelos preditivos.

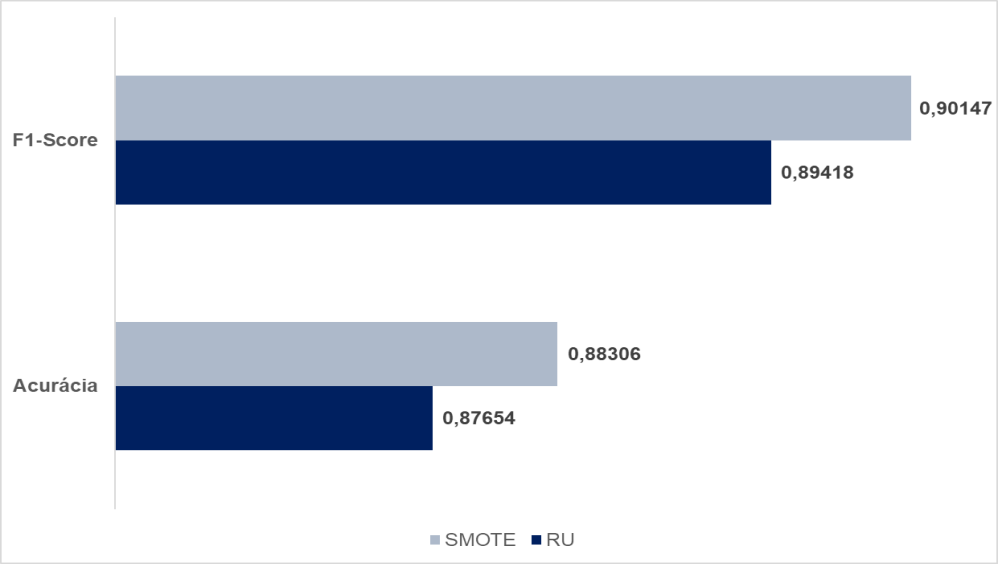
4.2 Modelos preditivos

Posteriormente ao selecionar os hiperparâmetros para cada modelo de classificação, segue-se para o treinamento e teste deles. Com o intuito de avaliar a performance e conseguir evidenciar o melhor modelo e técnica de balanceando entre os aspectos estudados, as métricas de avaliação utilizadas foram a acurácia e F1-Score, além da Curva ROC, conforme os próximos tópicos demonstram.

4.2.1 Regressão logística

A acurácia e F1-Score resultante da regressão logística podem ser vistos na Figura 20. Observa-se que ambas métricas de avaliação de desempenho obtiveram maior valor através do balanceamento SMOTE.

Figura 20 – Gráfico da acurácia de F1-Score obtidos através da regressão logística



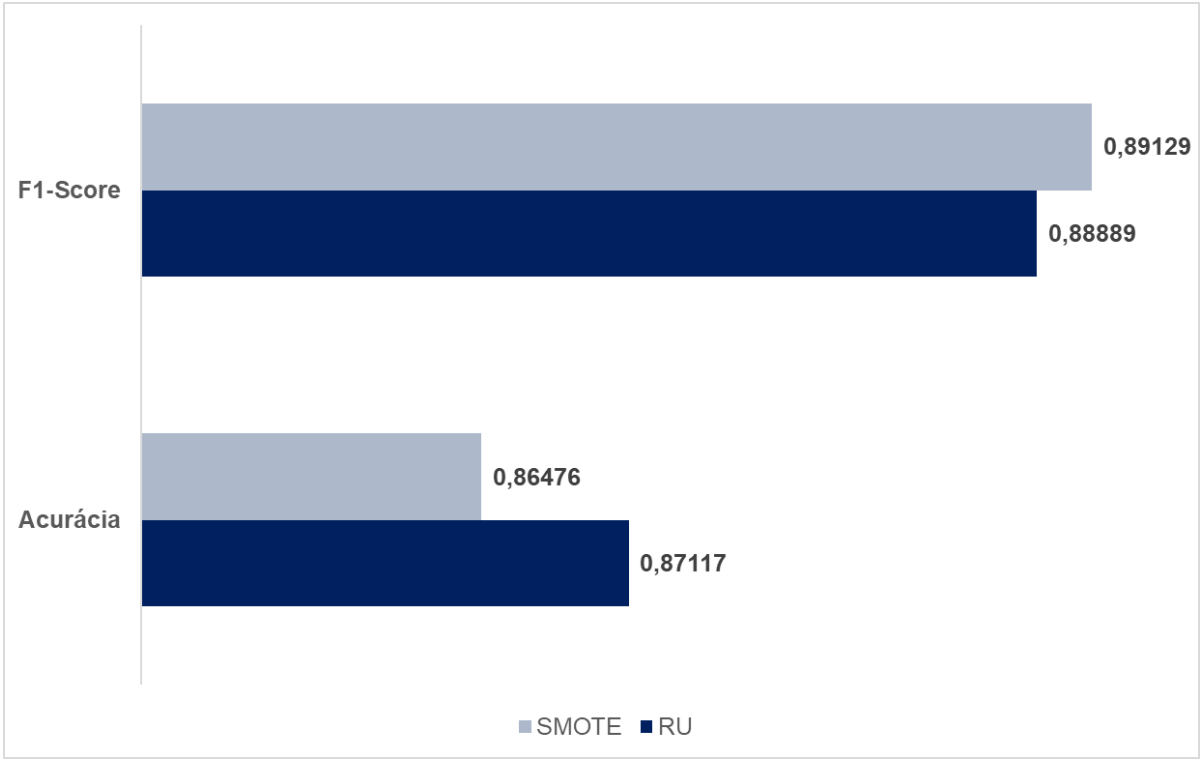
Fonte: Autor

Na Figura 20, observa-se os valores para técnica SMOTE em cinza e em azul escuro para a técnica RU. Nota-se que há pouca diferença entre as técnicas. Entretanto, mesmo pequeno, esse valor é significativo quando pensando em grande escala e espaço de tempo.

4.2.2 KNN

Para o classificador KNN, a acurácia e F1-Score podem ser observados na Figura 21. Nota-se que ambas métricas de avaliação de desempenho obtiveram maior valor do também pelo balanceamento SMOTE.

Figura 21 – Gráfico da acurácia de F1-Score obtidos através da KNN



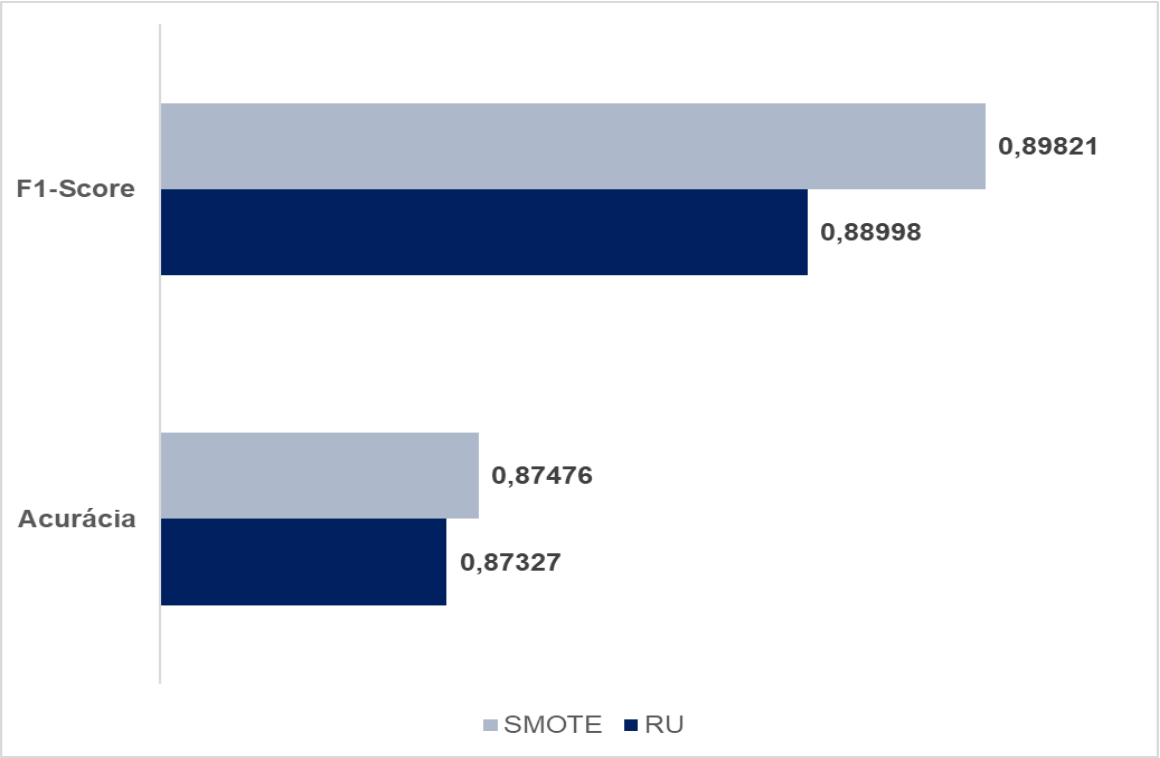
Fonte: Autor

Na Figura 21, observa-se os valores para técnica SMOTE em cinza e em azul escuro para a técnica RU.

4.2.3 SVM

No classificador SVM, pode-se observar o valor da acurácia e F1-score obtido ao final de predição na Figura 22. Novamente, assim como nos outros modelos, obteve-se resultados melhores com o balanceamento SMOTE.

Figura 22 – Gráfico da acurácia de F1-Score obtidos através da SVM



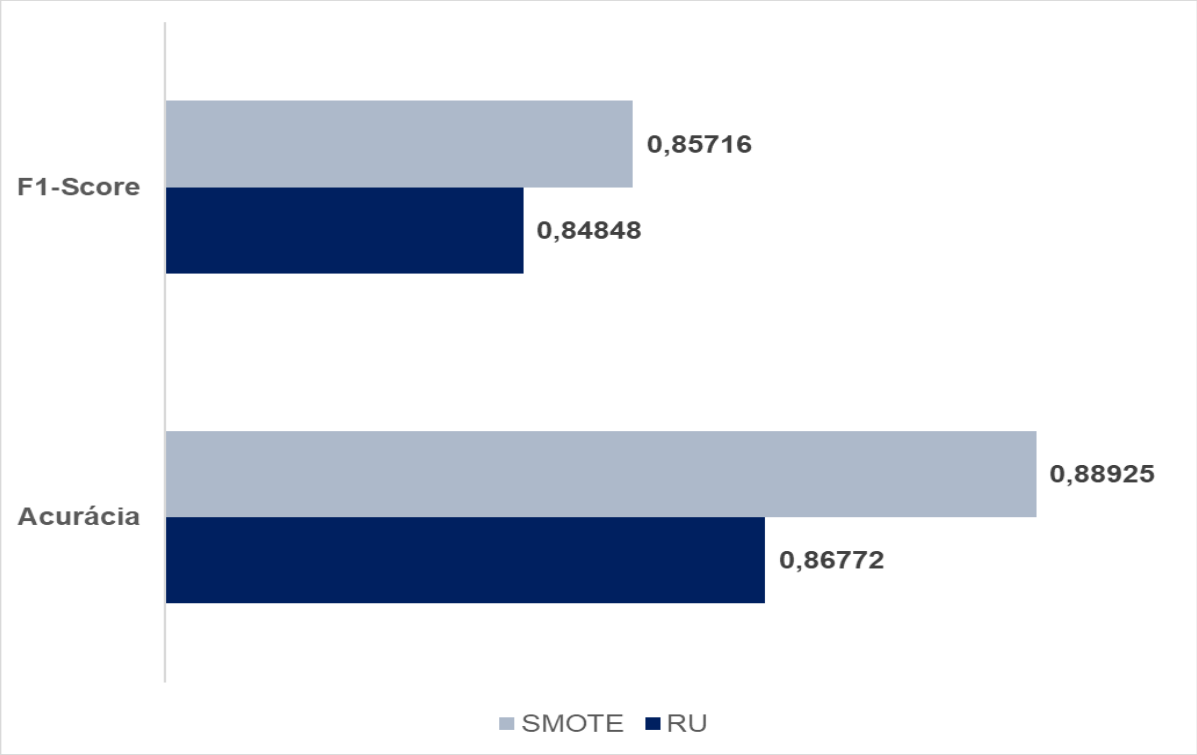
Fonte: Autor

Na Figura 22, observa-se os valores para técnica SMOTE em cinza e em azul escuro para a técnica RU.

4.2.4 Árvores de Decisão

A figura 23, exibe os resultados para as métricas acurácia e F1 – Score para o classificador Árvores de Decisão. Analogamente aos demais modelos, obteve-se melhores resultados com o balanceamento SMOTE.

Figura 23 – Gráfico da acurácia de F1-Score obtidos através das Árvores de Decisão



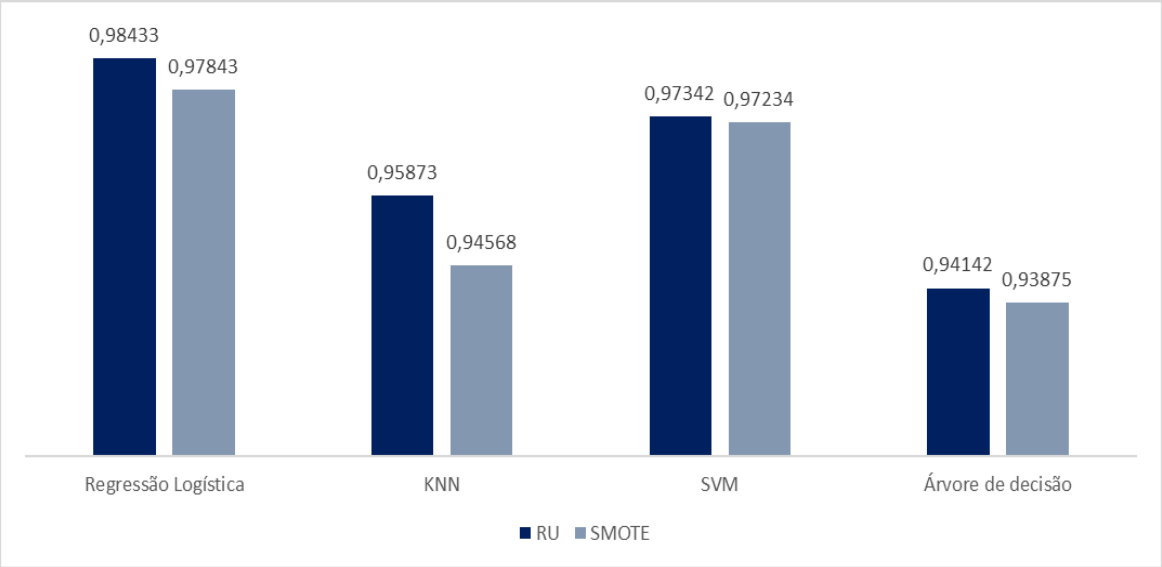
Fonte: Autor

Na Figura 23, observa-se os valores para técnica SMOTE em cinza e em azul escuro para a técnica RU.

4.2.5 Curva ROC

A área da curva ROC, como elucidada anteriormente, possibilita o diagnóstico rápido dos possíveis modelos com melhor performance em relação ao erro do modelo, ou seja, o quanto confiável é o classificador em relação as predições falsas-positivas. Dessa forma, observa-se os valores dessa métrica de avaliação na Figura 24 para os dois métodos de balanceamento.

Figura 24 – Gráfico da área sobre a curva ROC dos classificadores testados com os dataframes RU e SMOTE



Fonte: Autor

Como evidenciado na Figura 24, tem-se que os valores obtidos pelo dataframe com balanceamento RU obteve-se melhores resultados em relação a técnica SMOTE. Esse resultado pode ser explicado como uma das desvantagens do modo SMOTE, no qual cria dados fictícios do atributo minoritário. Dessa forma, não condizendo fielmente a realidade. Além disso, o dataframe original, como anteriormente citado, possui grande desbalanceamento, contribuindo para possíveis casos de overfiting.

Em relação aos classificadores, evidencia-se melhores performances para a Regressão Logística.

4.2.6 Overview

Em suma, pode-se destacar que o modelo de Regressão Logística possuiu melhores performances diante dos demais. Além disso, a técnica SMOTE também se obteve performance melhor em relação a RU. Ressalta-se também que, por mais que a métrica da área da curva ROC diagnosticou a técnica RU com melhor eficácia, o valor não pode ser conclusivo, uma vez que o dataframe é altamente desbalanceado e, evidenciou-se não ser uma a melhor métrica para tal cenário. Dessa forma, e de acordo com a metodologia e hiperparâmetros aqui utilizados, a melhor combinação entre método e técnica de balanceamento foi a regressão logística com o balanceamento SMOTE.

5. CONCLUSÃO

Em virtude do cenário atual, o mundo digital e dinâmico crescente, o pagamento através de cartões, principalmente do estilo crédito, tornou-se uma das principais formas de pagamento. Por consequência, a quantidade de transações fraudulentas em seus diversos estilos cresceu simultaneamente, tornando-se cada vez mais necessário obter métodos de prevenção a fraude. Um dos principais métodos utilizados pelas instituições bancárias é o emprego de modelos preditivos que consigam prever instantaneamente se a transação é fraudulenta ou não fraudulenta. Com isso, esse trabalho visou avaliar quatro modelos preditivos de classificação, KNN, SVM, Árvores de decisão e Regressão Logística. O dataframe utilizado está disponibilizado na rede, com transações reais em certo período na Europa. Para realizar a análise, também foram utilizadas duas técnicas de balanceamento.

A primeira atividade desse trabalhou consistiu no levantamento do referencial teórico de modo sucinto em relação aos modelos preditivos aqui abordados e a metodologia a ser executada. O ambiente trabalhado utilizado foi o Jupyter, com utilização da linguagem Python e suas bibliotecas (Anexo A).

Em relação a metodologia, realizou-se inicialmente a análise exploratória do dataframe original utilizado. Após o diagnóstico, buscou-se balancear o dataframe com as técnicas RU e SMOTE. Por consequência, a próxima etapa deu-se pela obtenção dos melhores hiperparâmetros para cada modelo preditivo. Nesse estágio, não se buscou a obtenção de todos os melhores hiperparâmetros, apenas os principais, uma vez que a obtenção de todos extrapola o escopo desse projeto.

O primeiro modelo avaliado, Regressão Logística, foi o de melhor performance, principalmente pelo balanceamento SMOTE. Com valor de acurácia **0,88306** e F1-Score **0,90147**. Além disso, o valor da área da curva ROC mostrou-se inconclusiva diante das outras métricas de avaliação. Possivelmente influenciada pelo alta desbalanceamento do dataframe original.

Além disso, vale ressaltar que os resultados aqui obtidos servem de insights para a construção de modelos preditivos mais robustos e melhores. Outro aspecto importante é avaliação de tais modelos em outros dataframes reais. A busca por melhores hiperparâmetros é essencial para o sucesso de um modelo. Pode-se

destacar também, que a fusão de técnicas independentes é cada vez mais frequente no mercado atual e possuem performances ainda melhores.

Como sugestão para trabalhos futuros, destaca-se: o uso de maiores dataset reais; utilização de outras técnicas de balanceamento de conjunto de dados; exploração de hiperparâmetros com maiores performances; modelagens preditivas mais robustas e complementares; utilização de métricas de avaliação melhores ao cenário da classificação, como revocação vs precisão.

REFERÊNCIAS BIBLIOGRÁFICAS

Bell, G., Hey, T., e Szalay, A. (2009). Beyond the Data Deluge. Science, 323:1298–1298. Chen, T. M. (2001). **Increasing the observability of internet behavior. Communications of the ACM**, 44(1):93–98

MARSLAND, Stephen. **Machine Learning: An Algorithmic Perspective: An Algorithmic Perspective**. 2. ed. Ashhurst, New Zealand: Crc Press, 2014. 452 p

SPC Brasil, **SPC BRASIL NO ESTUDO DAS FRAUDES**, [S.1]: Site oficial, 2013

YAOHAO, P.; MATION, L. F. **O desafio do pareamento de grandes bases de dados: mapeamento de métodos de record linkage probabilístico e diagnóstico de sua viabilidade empírica**. Instituto de Pesquisa Econômica Aplicada (Ipea), 2018.

MURPHY, Kevin P.. **The Machine Learning: A Probabilistic Perspective**. Cambridge: Mit Press, 2012.

SCIKIT-LEARN. **Linear Models**. 2020. Disponível em: <https://scikit-learn.org/stable/modules/ linear_model.html>. Acesso em: 18 jan. 2021.

IZBICKI, Rafael; SANTOS, Tiago Mendonça dos. **Aprendizado de máquina: uma abordagem estatística**. São Carlos: ., 2020. 272 p

MESQUITA, P. S. B. **Um Modelo de Regressao Logistica para Avaliaçao dos Programas a Pós-Graduação no Brasil**. Dissertação (Mestrado) - Universidade Estadual do Norte Fluminense, Campo dos Goytacazes, 2014.

GONZALEZ, Leandro de Azevedo. **Regressão Logística e suas Aplicações**. 2018. 46 f. Tese (Doutorado) - Curso de Ciências da Computação, Universidade Federal do Maranhão, São Luíz, 2018

SCIKIT-LEARN**. sklearn.metrics.classification_report**. 2021. Disponível em <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>. Acesso em: 03 mar. 2021

OLIVEIRA, Paulo Henrique Maestrello Assad. **Detecção de fraudes em cartões: um classificador baseado em regras de associação e regressão logística**. 2016. 117 f. Tese (Doutorado) - Curso de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016

NICOLA, Victor Gomes de Oliveira Martins; LAURETTO, Marcelo de Souza; DELGADO, Karina Valdivia. **Avaliação empírica de classificadores e métodos de balanceamento para detecção de fraudes em transações com cartões de créditos**. 2019. 12 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade de São Paulo, São Paulo, 2019.

SCIKIT-LEARN. **Tuning the hyper-parameters of an estimator**. 2020. Disponível em: <https://scikit-learn.org/stable/modules/grid_search.html>

APÊNDICES

APÊNDICE A – Bibliotecas Python utilizadas nesse trabalho.

```
import numpy as np
import pandas as pd
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
import time
import matplotlib.pyplot as plt
import seaborn as sns
import warnings


import keras
from keras import backend as K
from keras.models import Sequential
from keras.layers import Activation
from keras.layers.core import Dense
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy

from sklearn.manifold import TSNE
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, accuracy_score, classification_report,
                roc_curve, precision_recall_curve, average_precision_score, confusion_matrix
from sklearn.model_selection import KFold, StratifiedKFold, train_test_split,
            StratifiedShuffleSplit, cross_val_score, GridSearchCV, ShuffleSplit, learning_curve,
            cross_val_predict, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.pipeline import make_pipeline

from imblearn.pipeline import make_pipeline as imbalanced_make_pipeline
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import NearMiss
from imblearn.metrics import classification_report_imbalanced
```