UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Alecsander Cucke

Avaliação de técnicas de machine learning aplicada ao processo de classificação de potenciais clientes de uma empresa SAAS

Florianópolis
2023

Alecsander Cucke

Avaliação de técnicas de machine learning aplicada ao processo de classificação de potenciais clientes de uma empresa SAAS

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Rodolfo C. C. Flesch, Dr.
Supervisor: Danilo Zili Pavei, Eng.

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Cucke, Alecsander
Avaliação de técnicas de machine learning aplicada ao processo de classificação de potenciais clientes de uma empresa SAAS / Alecsander Cucke ; orientador, Rodolfo César Costa Flesch, 2023.
76 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Aprendizado de máquina. 3. Dados. 4. Classificação de clientes. I. Flesch, Rodolfo César Costa. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Alecsander Cucke

Avaliação de técnicas de machine learning aplicada ao processo de classificação de potenciais clientes de uma empresa SAAS

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 11 de julho de 2023.

Prof. Hector Bessa Silveira, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Rodolfo C. C. Flesch, Dr.
Orientador
UFSC/CTC/DAS

Danilo Zilli Pavei, Eng.
Supervisor
Empresa Asksuite Tecnologia Ltda.

Prof. Jean Panaioti Jordanou, Doutorando.
Avaliador
UFSC/CTC/DAS

Prof. Eduardo Camponogara, Dr.
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus colegas de classe e
aos meus queridos pais.

AGRADECIMENTOS

Quero expressar minha sincera gratidão aos meus pais, Jailson Cucke e Luci de Costa, por seu constante incentivo e apoio em seguir meus sonhos, independentemente das circunstâncias. Eles têm sido uma fonte de inspiração, impulsionando-me a perseguir meus objetivos. Tenho imenso orgulho de ser filho deles.

Gostaria também de expressar minha profunda gratidão às minhas tias Silvia de Costa e Jaquelaine Cucke, que, admiráveis por seus valores de respeito, persistência e paixão, desempenharam um papel fundamental na realização do meu sonho de estudar na Universidade Federal de Santa Catarina.

Agradeço também aos colegas de curso e amigos que conheci ao longo do caminho. Compartilhamos momentos e aprendizados preciosos e construímos laços fortes. Essa rede de apoio foi fundamental para enfrentar desafios de maneira equilibrada. Agradeço à minha namorada Catarina Guedin, por todo apoio fornecido, pela sua simplicidade e pelo cuidado, que foram fundamentais.

Também desejo expressar meus agradecimentos ao Professor Rodolfo César Flesch pela orientação neste trabalho.

À Asksuite, sou imensamente grato pela oportunidade de adquirir experiência profissional na área de tecnologia. A empresa foi uma escola que me permitiu crescer profissionalmente e aplicar os conhecimentos adquiridos durante minha graduação.

Por fim, dedico este trabalho a Alcioni Panatto (*in memorian*), uma das melhores pessoas que já conheci e que me ensinou valiosas lições de vida. Sua memória será sempre uma inspiração para mim.

“Inteligência é a capacidade de absorver informação em tempo real.
De fazer perguntas que façam sentido. É ter boa memória.
É traçar pontes entre assuntos que não
parecem estar relacionados e inovar ao fazer essas conexões.”
(GATES, 2020)

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 30 de junho de 2023.

Na condição de representante da Asksuite Tecnologia Ltda. na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o autor, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.

_____

Danilo Zilli Pavei
Asksuite Tecnologia Ltda.

declaracao_sigilo_alecsander.pdf

Documento número #64f95238-6ac0-421d-8bdd-9ddfe2d04d76

Hash do documento original (SHA256): c79df96e4fc00faf6844fbf3b3ac6fce84fe24915dba4f69898543a2e634b24e

Assinaturas

Danilo Zilli Pavei
CPF: 006.068.339-27
Assinou em 02 jul 2023 às 16:37:37

Log

02 jul 2023, 16:34:40	Operador com email danilo@asksuite.com na Conta f42256f5-25b1-4102-91bd-3d07671e4628 criou este documento número 64f95238-6ac0-421d-8bdd-9ddfe2d04d76. Data limite para assinatura do documento: 01 de agosto de 2023 (16:33). Finalização automática após a última assinatura: habilitada. Idioma: Português brasileiro.
02 jul 2023, 16:34:42	Operador com email danilo@asksuite.com na Conta f42256f5-25b1-4102-91bd-3d07671e4628 adicionou à Lista de Assinatura: danilo@asksuite.com para assinar, via E-mail, com os pontos de autenticação: Token via E-mail; Nome Completo; CPF; endereço de IP.
02 jul 2023, 16:37:37	Danilo Zilli Pavei assinou. Pontos de autenticação: Token via E-mail danilo@asksuite.com. CPF informado: 006.068.339-27. IP: 177.174.240.8. Componente de assinatura versão 1.531.0 disponibilizado em https://app.clicksign.com.
02 jul 2023, 16:37:37	Processo de assinatura finalizado automaticamente. Motivo: finalização automática após a última assinatura habilitada. Processo de assinatura concluído para o documento número 64f95238-6ac0-421d-8bdd-9ddfe2d04d76.

Documento assinado com validade jurídica.
Para conferir a validade, acesse https://validador.clicksign.com e utilize a senha gerada pelos signatários ou envie este arquivo em PDF.
As assinaturas digitais e eletrônicas têm validade jurídica prevista na Medida Provisória nº. 2200-2 / 2001

Este Log é exclusivo e deve ser considerado parte do documento nº 64f95238-6ac0-421d-8bdd-9ddfe2d04d76, com os efeitos prescritos nos Termos de Uso da Clicksign, disponível em www.clicksign.com.

RESUMO

O projeto consiste no desenvolvimento de ferramentas que permitam a automação do processo de classificação de potenciais clientes de uma empresa de software como serviço. O processo originalmente é custoso, realizado de forma imprecisa e suscetível a erro humano. Inicialmente, foi feita uma análise exploratória de dados para visualizar características dos dados registrados no sistema de relacionamento com o cliente da empresa. Uma vez pré-processados, os dados foram tratados por modelos e técnicas de aprendizado de máquina para realizar classificação, com característica de multiclasse em dados desbalanceados. Uma prova de conceito é proposta em busca de automatizar o processo utilizando o modelo com melhor desempenho para realizar predições consistentes na base de dados já conhecida.

Palavras-chave: Aprendizado de máquina. Dados. Classificação de clientes.

ABSTRACT

The project consists of the development of tools that managed to automate the process of classifying potential customers of a software-as-a-service company. The process is originally costly, performed inaccurately and susceptible to human error. Initially, an exploratory data analysis was carried out to visualize characteristics of the data recorded in the company's customer relationship system. Once pre-processed, the data were treated by models and machine learning techniques to perform classification, with multiclass characteristic in unbalanced data. A proof of concept is proposed in order to automate the process using the model with the best performance to make consistent predictions in the already known database.

Keywords: Machine learning. Data. Client classification.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACATE	Associação Catarinense de Tecnologia
ADR	*Average Daily Rate*
API	*Application Programming Interface*
BPMN	Notação de modelagem de processo de negócio
CRISP DM	*Cross Industry Standard Process for Data Mining*
CRM	*Customer Relationship Management*
CSV	*Comma Separated Values*
EDA	*Exploratory Data Analysis*
EMEA	*Europe, Middle East, and Africa*
GPTW	*Great Place to Work*
GPU	*Graphics Processing Unit*
HTTP	*Hypertext Transfer Protocol*
ICP	*Ideal Customer Profile*
KNN	*K-nearest neighbor*
LATAM	*Latin America*
ML	*Machine Learning*
NMRR	*New Monthly Recurring Revenue*
OvR	*One Versus Rest*
RNA	Rede neural artificial
SaaS	*Software as a Service*
SMOTE	*Synthetic Minority Over-sampling Technique*
TPU	*Tensor Processing Unit*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo é abordada a motivação principal do projeto, bem como o seu impacto atual na empresa. Nele também são discutidas a importância de se ter dados confiáveis no processo de classificação de um potencial cliente e a relação dessa classificação com o potencial de receita da empresa. Além disso, são expostos os objetivos gerais e os objetivos específicos que nortearam o desenvolvimento do trabalho. Por fim, é apresentada a metodologia utilizada em cada fase do projeto, momento em que se discutem os métodos de análise de dados, o *pipeline* para desenvolvimento de projetos de aprendizado de máquina e o plano de execução utilizado no trabalho.

1.1 MOTIVAÇÃO

Nos últimos anos, as empresas, especialmente no ramo de tecnologia, têm gerado em suas estruturas um grande registro de dados focados na jornada do cliente. Esses dados ficam centralizados em sistemas de *Customer Relationship Management* (CRM), conhecidos em português como sistemas de gerenciamento de relacionamento com o cliente. Com o mercado cada vez mais exigente e competitivo, é de grande importância tomar decisões fundamentadas em dados, principalmente no mercado de tecnologia, que é volátil. Os registros de dados podem variar de acordo com mercado e características que cada empresa pode vir a definir. Com os registros realizados, é possível analisar e decidir com base em comportamentos e ter maior previsibilidade nas decisões de valor, bem como nos processos, incluindo o de tomada de decisão estratégica (SERGUE, 2020).

O processo de geração e classificação de potenciais clientes (tipicamente conhecidos pelo termo em inglês *leads*) é feito em diferentes etapas do processo de vendas de uma empresa. No caso específico da Asksuite, empresa na qual este projeto de fim de curso foi desenvolvido, por vezes a classificação do porte de *leads* só é realizada nas fases finais e com grande dependência dos agentes envolvidos no processo. Além disso, as regras atuais para classificação foram construídas por meio de inferência no mercado de atuação da empresa.

Por conta dessas limitações, o processo de classificação vem se tornando pouco preciso, extremamente manual e associado a um alto custo de operação para a empresa. Adicionalmente, a empresa está em processo internacionalização e necessita de um melhor entendimento das classes de clientes para apoiar a tomada de decisão.

1.2 OBJETIVOS

Esta seção descreve o objetivo geral e os objetivos específicos do trabalho.

1.2.1 Objetivo Geral

O projeto tem como objetivo geral avaliar modelos de *Machine Learning* (ML), ou aprendizado de máquina em português, para classificação do porte de potenciais clientes da empresa Asksuite, de modo que a empresa possa ter uma classificação de potenciais clientes mais confiável e aprimorar o processo de geração de *leads*, tornando-o mais eficiente e direcionado para as oportunidades com maior potencial de conversão em clientes.

1.2.2 Objetivos Específicos

O projeto possui os seguintes objetivos específicos:

- realizar análise exploratória dos dados para entender o perfil dos potenciais clientes e identificar possíveis padrões. Essa análise permitirá compreender melhor as características da base de dados e facilitará a escolha dos modelos de aprendizado de máquina;
- avaliar modelos de aprendizado de máquina com base em métricas de desempenho;
- criar prova de conceito com o objetivo de automatizar a classificação dos potenciais clientes;
- criar um protótipo de *dashboard* para visualização dos dados de predição para uso da liderança da empresa.

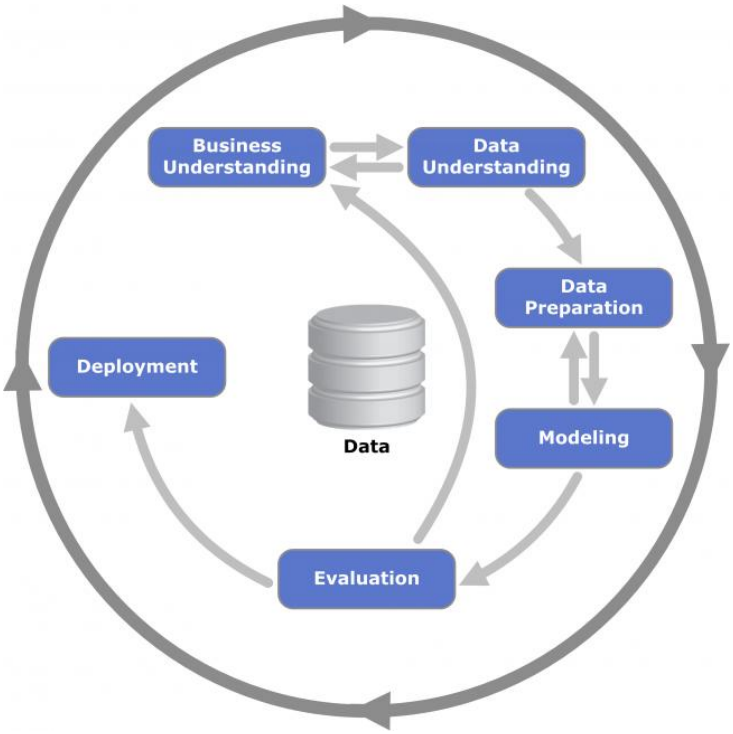
1.3 METODOLOGIA

Este projeto teve como base metodológica a *Cross-industry standard process for Data Mining* (Processo interprofissional padrão para mineração de dados, ou), comumente utilizada em projetos de mineração de dados, mas que pode ser empregada em projetos de ML. Essa metodologia consiste em um *framework* para utilizar problemas de negócios em tarefas de mineração de dados (HUBER *et al.*, 2019). No caso deste projeto, ela realiza a tradução do problema de negócio da classificação de potenciais clientes em tarefas de análise exploratória de dados e ML.

A figura 1 ilustra a estrutura da metodologia CRISP-DM. Como pode ser observado, esta metodologia possui seis fases principais em seu ciclo de vida, que são detalhadas na sequência.

1. Entendimento do negócio (*business understanding*): deve-se compreender os objetivos e requisitos do projeto e formular hipóteses de como o problema será abordado.

Figura 1 – Ciclo de vida da metodologia CRISP-DM.



Fonte: Retirado de (HUBER *et al.*, 2019).

2. Compreensão dos dados (*data understanding*): esta fase se inicia com a coleta dos dados, identificação de suas características e formação de hipóteses iniciais. Ela tem ligação direta com o entendimento do negócio, pois a junção desses conhecimentos ajuda na formulação do problema a ser resolvido.

3. Preparação dos dados (*data preparation*): compreende todo conjunto de atividades necessárias para construção do *dataset* que será utilizado nas fases subsequentes. Esta etapa pode ser executada várias vezes com objetivo de se ter um conjunto robusto para uso. Aqui entram atividades como limpeza de dados, identificação de *outliers* (valor atípico) e transformação dos dados.

4. Modelagem (*modeling*): nesta fase, dá-se o emprego de diferentes técnicas de modelagem adequadas para o problema em questão, a criação dos modelos de ML, o ajuste de seus parâmetros e a avaliação de sua performance. Por exemplo, alguns modelos requerem parametrizações específicas para problemas de classificação multiclasse.

5. Avaliação (*evaluation*): após construção dos modelos, deve-se avaliá-los com base em métricas apropriadas para verificar se podem ser encaminhados para a implantação final. Deve-se aqui também revisar as etapas desenvolvidas do modelo e averiguar se esse atinge os objetivos de negócio traçados no início do

processo. Nesta fase é importante ter uma visão crítica dos resultados e definir a utilização dos modelos desenvolvidos.

6. Implantação (*deployment*): a mera criação de um modelo que atenda a determinados requisitos não é suficiente. A depender dos requisitos de projeto, esta fase pode se desenrolar de maneira simples – por exemplo, quando realiza a geração de relatórios –, como também pode se apresentar uma tarefa mais desafiadora – como quando executa a implementação de um processo a ser utilizado por outros usuários (WIRTH; HIPP, 2000).

1.4 ESTRUTURA DO DOCUMENTO

O documento é dividido em sete capítulos. O capítulo 1 faz uma introdução à motivação do projeto e à sua relevância. Nele também são exposto os objetivos que o projeto deve alcançar, contextualizando a avaliação de modelos de aprendizado de máquina para classificação multiclasse.

O capítulo 2 concentra-se na apresentação da empresa em que o projeto foi desenvolvido, introduzindo sua estrutura organizacional, bem como o escopo da área de *Sales Operations* e sua importância e atuação em atividades de dados, processos e inteligência de mercado.

No capítulo 3 é exposto o embasamento teórico utilizado para o desenvolvimento do projeto.

Já no capítulo 4 é apresentado o processo AS-IS, a estrutura geral do projeto, bem como seus requisitos funcionais e não funcionais, e a estrutura da prova de conceito aplicada neste trabalho.

O capítulo 5 mostra o desenvolvimento das etapas do pipeline do projeto de aprendizado de máquina e a prova conceitual de uso dessas técnicas no processo comercial. Os resultados obtidos serão expostos de uma forma aprofundada no capítulo 6.

Por fim, no capítulo 7 é feita a síntese final do projeto e a indicação dos próximos passos a serem explorados em projetos futuros.

2 EMPRESA ASKSUITE

A Asksuite é uma empresa de software como serviço (*Software as a Service* (SaaS)) com sede em Florianópolis, que tem como objetivo simplificar a comunicação direta entre hotéis e viajantes por meio de uma plataforma *omnichannel*. A plataforma conecta diferentes canais de comunicação, como chat em *websites*, WhatsApp e Facebook Messenger, com um assistente virtual para atendimento on-line dentro do mercado hoteleiro.

Fundada em 2018, a empresa foi pioneira no uso de inteligência artificial em comunicação on-line voltada para o mercado hoteleiro global, tendo conquistado o prêmio de melhor assistente virtual para hotelaria do mundo por quatro vezes consecutivas, segundo a Hotel Tech Report, um *hub* de tecnologia hoteleira que realiza essa competição por meio de votos e *feedbacks* dos próprios clientes das empresas.

Nos primeiros anos de operação, a empresa utilizou o modelo *bootstrapping*, ou seja, usou recursos próprios para crescer, até ser incubada pela MIDITEC, que está entre as cinco melhores incubadoras de *startups* do mundo e faz parte da Associação Catarinense de Tecnologia (ACATE).

A Asksuite também participou de programas importantes, como o Scale Up da Endeavor, principal portal de empreendedorismo nacional, e já foi classificada como uma das melhores empresas para as quais trabalhar via o *Great Place to Work* (GPTW)) tanto pelo Hotel Tech Report quanto pelo Best Place to Work. Em 2020, a empresa recebeu seu primeiro aporte financeiro por meio de uma rodada *seed* da ABseed.

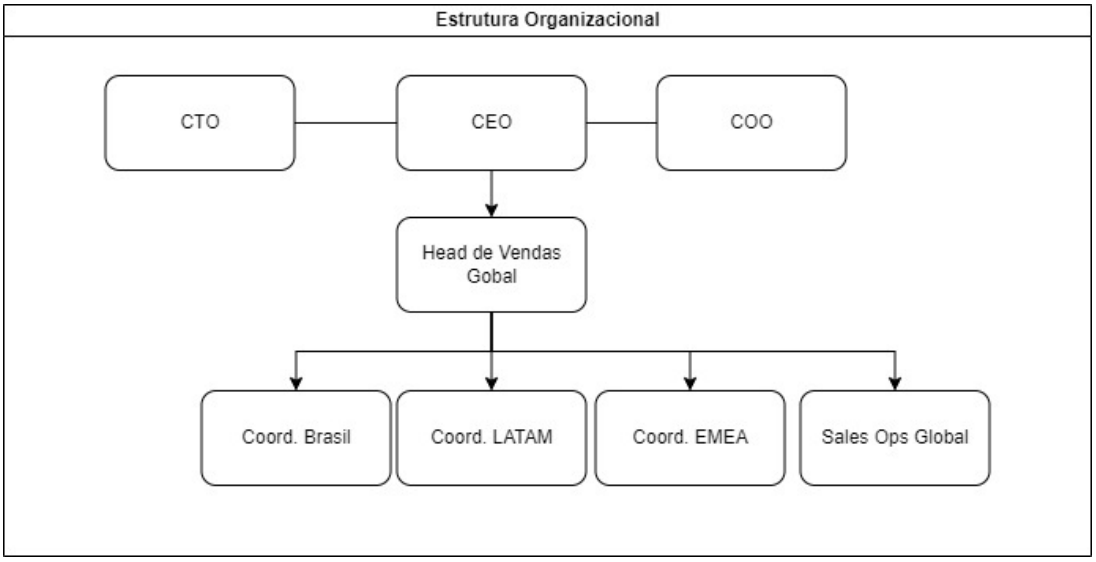
Atualmente, a Asksuite possui mais de 3000 clientes em 40 países e conta com uma equipe de 115 colaboradores presentes em 7 países.

2.1 ESTRUTURA ORGANIZACIONAL

A estrutura organizacional é composta pelos cargos executivos de *Chief Executive Officer* (diretor geral da empresa, ou CEO), *Chief Technology Officer* (diretor de tecnologia, ou CTO) e *Chief Operating Officer* (diretor operacional, ou COO). Eles têm objetivo de condução da estratégia, tecnologia e operações da empresa.

Abaixo do CEO há duas diretorias, representadas por Marketing e Vendas. Dentro da diretoria de vendas atualmente existem quatro coordenações. Juntas, elas têm o objetivo de desenvolver as estratégias de vendas e elaborar o planejamento com objetivo de alcançar as metas de receita da empresa. Dentro dessa estrutura, existe uma coordenação para cada uma das principais regiões de atuação da empresa, sendo elas o Brasil, a América Latina (*Latin America* (LATAM)) e o conglomerado Europa, Oriente Médio e África (*Europe, Middle East, and Africa* (EMEA)), como detalhado na figura 2.

Figura 2 – Estrutura Organizacional Simplificada.



Fonte: Autor.

Para dar apoio operacional a toda essa estrutura, existe a área de *Sales Operations*, que atua de forma global. Ela tem o objetivo de garantir eficiência dos processos, sistemas e coleta de dados utilizados pela estrutura de vendas, e também de gerar análises da operação para a criação de *insights* do desempenho e de estratégia da área. O autor, atua como coordenador global da área. Sendo responsável pela estruturação dos processos do setor de vendas da empresa e responsável das frentes de atuação detalhadas na seção 2.2.

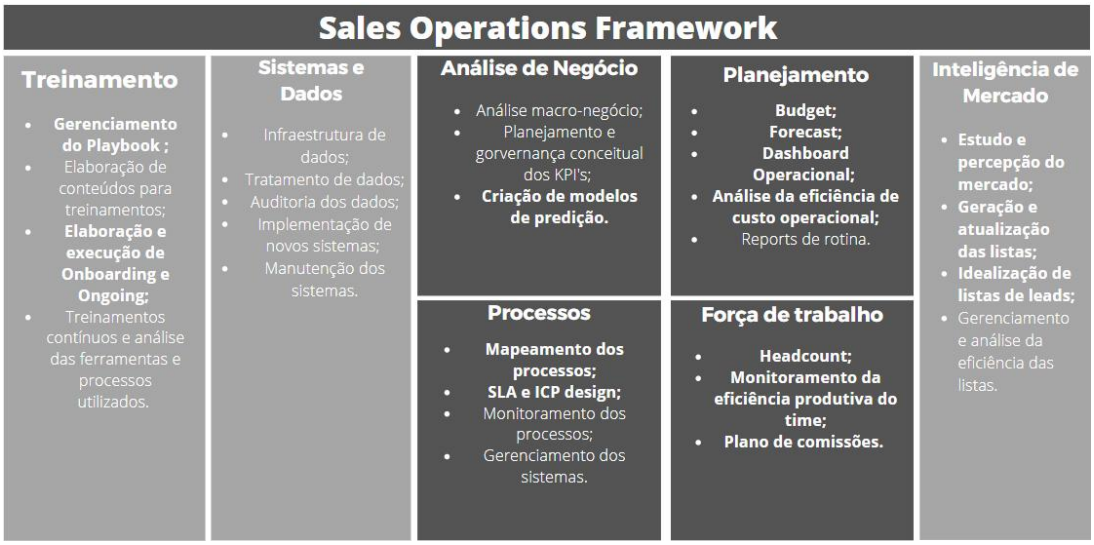
2.2 A ÁREA SALES OPERATIONS

A área de Sales Operations da Asksuite tem frentes de atuação bem claras para suportar operacionalmente uma área de vendas com atuação global. Essas frentes são:

- governança e análise dados;
- governança de processos;
- inteligência de mercado.

Cada pilar de atuação contribui de maneira específica para a evolução da área de vendas e, consequentemente, da empresa. Na figura 3 é possível ver as diferentes atividades desempenhadas por cada uma delas. A frente de governança e análise de dados atua desde a concepção estrutural dos dados nos sistemas utilizados até a análise deles, com o objetivo de levantar informações que sejam relevantes para

Figura 3 – Framework da área de Sales Operations.



Fonte: Autor.

utilização cotidiana e para o planejamento estratégico. A frente de processos é encarregada da criação, manutenção e melhoria contínua, garantindo assim que haja clareza e eficiência na área. Por fim, a frente de inteligência de mercado tem como foco as atualizações e os estudos sobre o mercado, bem como a garantia da geração de demanda de *leads*.

3 FUNDAMENTAÇÃO TEÓRICA

Como foi apresentado nos capítulos anteriores, a problemática do projeto é a dependência de classificação humana dos portes de potenciais clientes. Os objetivos do projeto buscam avaliar técnicas mais sofisticadas para resolução desse problema.

Neste capítulo é introduzida a fundamentação teórica utilizada para o desenvolvimento deste trabalho. Nele são expostas a análise exploratória de dados e as abordagens de aprendizado de máquina, que incluem regressão logística, árvores de decisão, *random forest*, *K-nearest neighbor* (KNN) e Rede neural artificial (RNA). Também são expostas técnicas para lidar com classificação multiclasse e classes desbalanceadas.

3.1 ANÁLISE EXPLORATÓRIA DE DADOS

A Análise Exploratória de Dados (*Exploratory Data Analysis* (EDA), na sigla em inglês) é uma técnica utilizada para analisar e obter informações de um conjunto de dados que possam ser relevantes (SAHOO *et al.*, 2019). Com o uso dessa técnica é possível identificar tendências e padrões de dados, bem como apontar suas anomalias.

Em um projeto de análise de aprendizado de máquina, a EDA tem grande importância, pois ajuda a identificar problemas na medição dos dados, como valores nulos e *outliers*. Essa identificação possibilita a correção dessas anomalias antes da modelagem dos dados, o que garante uma maior qualidade e, consequentemente, confiança nos resultados obtidos.

A EDA possui diferentes técnicas estatísticas e de visualização de dados, tais como histogramas, gráficos de dispersão, *box plots* e mapas de calor, que permitem o entendimento da distribuição dos dados e das relações entre as variáveis. (SAHOO *et al.*, 2019)

Uma técnica muito importante na EDA é a análise de correlação, por meio da qual a ferramenta permite avaliar a força e direção da relação entre duas ou mais variáveis. No projeto de avaliação de modelos de aprendizado de máquina, essa técnica é importante para entender como os dados utilizados podem afetar o resultado final do trabalho.

3.2 DESBALANCEAMENTO DE DADOS

Alguns modelos de ML assumem que os dados de entrada no modelo se encontram balanceados, isto é, têm uma proporção similar de registros. Porém esta distribuição nem sempre será possível em exemplos práticos (SKRYJOMSKI; KRAWCZYK, 2017). Para se analisar o desempenho dos algoritmos, normalmente é utilizada a acurácia, no entanto apenas a utilização desta métrica não é suficiente quando há dados

desbalanceados (CHAWLA *et al.*, 2002).

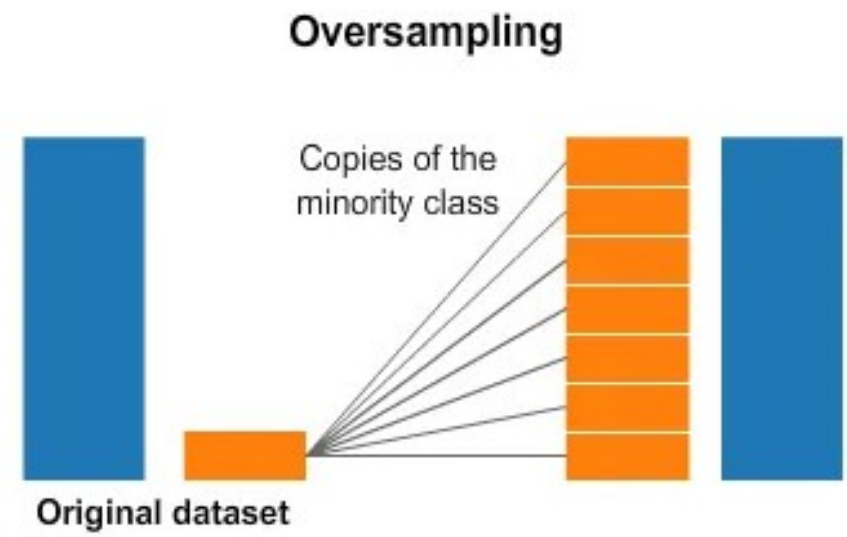
Existem dois tipos de abordagens a serem utilizados para resolução desse tipo de problema. O primeiro é atuar no pré-processamento de dados e o segundo é utilizar modelos de ML com maior robustez. Algoritmos do tipo *ensemble*, por exemplo, são muito populares quando os dados têm característica desbalanceada em sua distribuição (SKRYJOMSKI; KRAWCZYK, 2017).

No projeto em questão, este problema foi solucionado utilizando as duas abordagens, a depender do modelo estudado. A comparação entre os resultados do emprego das duas abordagens pode ser encontrada no capítulo 6.

3.2.1 Synthetic Minority Over-sampling Technique (SMOTE)

A ideia por trás do método *Synthetic Minority Over-sampling Technique* (SMOTE) é equilibrar as classes presentes de entrada quando se tem desbalanceamento de dados. Essa técnica gera dados sintéticos na classe minoritária com o objetivo de aproximar esses dados criados dos vizinhos mais próximos, interpolando suas características (CHAWLA *et al.*, 2002). Essa aplicação ajuda o modelo a obter mais equilíbrio na fase de teste, pois naturalmente suas predições poderiam pender para a classe majoritária e assim sofrer com *overfitting* (sobreajuste, ou ajuste exagerado, do modelo aos dados ou amostras).

Figura 4 – Funcionamento SMOTE.



Fonte: Retirado de (DAS, s.d.).

No entanto, vale a ressalva de que o SMOTE pode não ter desempenho tão satisfatório em todas as aplicações. Os exemplos sintéticos podem vir a se aproximar

de vizinhos que não representam a classe minoritária, além de poder ocorrer perda de informação. O uso dessa técnica é sugerido quando há caso de problemas de classificação binária. Porém, combinado com modelos mais robustos e outras técnicas, os resultados podem vir a ser positivos (DEMŠAR, 2006).

3.3 MODELOS DE APRENDIZADO DE MÁQUINA

Esta seção apresenta os modelos de ML empregados neste trabalho. A seção 3.3.1 trata da regressão logística, a seção 3.3.3 de redes neurais artificiais, a seção 3.3.4 de *K-nearest neighbor*, a seção 3.3.5 de árvores de decisão e a seção 3.3.6 da técnica *random forest*.

3.3.1 Regressão Logística

A regressão logística é um algoritmo de aprendizado de máquina supervisionado amplamente utilizado na resolução de problemas de classificação binária, por meio do qual se realiza a predição de uma variável de saída categórica (BISHOP, 2006). No entanto, este algoritmo também pode ser aplicado a problemas de classificação multiclasse, nos quais a variável de saída possui mais de duas classes.

Este algoritmo pode ser compreendido como uma aproximação linear das *features* de entrada do modelo. Essa aproximação é realizada pela função sigmoide, definida como $f(x) = \frac{1}{1+e^{-x}}$. A função sigmoide é fundamental, pois mapeia números reais para o intervalo de [0, 1], permitindo a interpretação probabilística das saídas do modelo (STANFORD UNIVERSITY, 2017).

Por trás do modelo de regressão logística multiclasse, encontra-se a função de custo conhecida como "função de custo logístico multiclasse" representada por:

$$J_{\text{rl}} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right), \tag{1}$$

em que J_{rl} representa o custo, N representa o número total de exemplos de treinamento, y_i é o vetor binário com 1 na posição da classe correta do exemplo i e 0 nas outras posições, e p_i é a probabilidade prevista para a classe correta do exemplo i. A soma é realizada sobre todos os N exemplos de treinamento. Ela tem como objetivo medir o quão bem o modelo está fazendo as predições em relação às classes reais. Uma função comumente utilizada é a função de entropia cruzada.

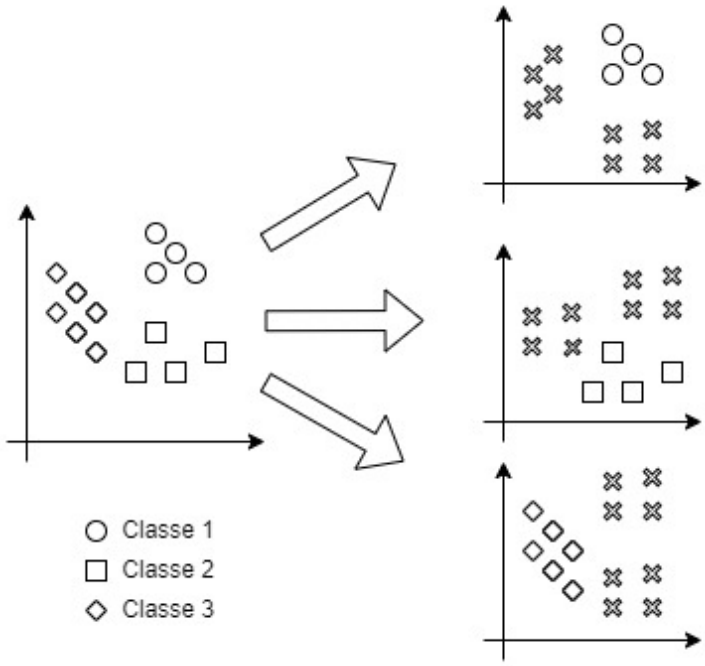
Durante o treinamento, o modelo de regressão logística multiclasse busca encontrar os melhores valores para os coeficientes (pesos) que multiplicam as variáveis independentes, de modo que a função de custo seja minimizada. Para esse fim, técnicas de otimização, como o gradiente descendente, são frequentemente empregadas para ajustar iterativamente os pesos com base nos erros de previsão.

Para estender a regressão logística a problemas de classificação multiclasse, existem duas abordagens principais: *One-vs-Rest* e *Softmax* (MITCHELL, 1997).

3.3.2 One vs Rest

A técnica *One-vs-Rest* (OvR), também conhecida como *One-vs-All*, é uma abordagem para solucionar problemas de classificação multiclasse. A ideia por trás dela é a criação de um classificador binário para cada uma das classes referentes às demais (LOGISTICREGRESSION. . . , s.d.). Na figura 5 vê-se o funcionamento da técnica em uma aplicação em que há a presença de três classes distintas. Em sequência, o algoritmo trata uma classe como positiva e as demais como negativas, e assim por diante. Essa técnica, no entanto, pode ter seu desempenho afetado quando a base de dados traz classes desbalanceadas (XU, 2011).

Figura 5 – Estrutura One-vs-Rest.



Fonte: Autor.

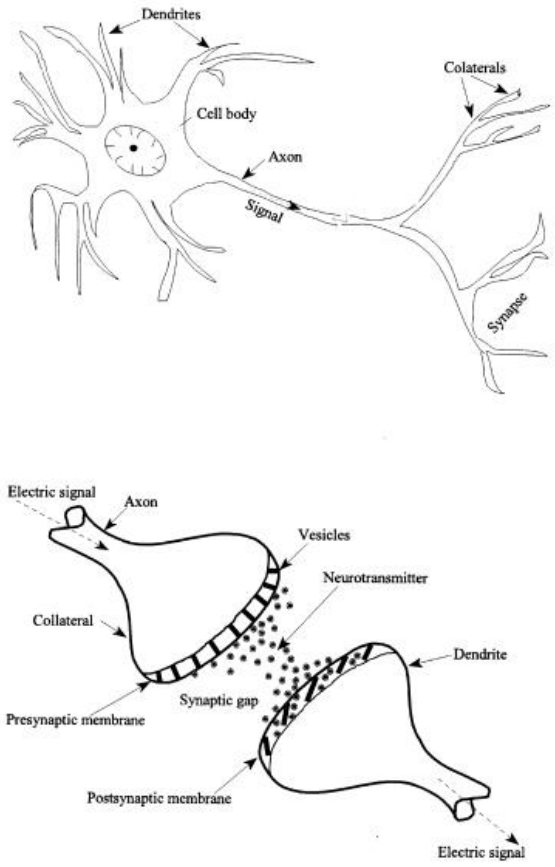
3.3.3 Redes neurais artificiais

Rede neural artificial (RNA) é uma abordagem computacional com inspiração no funcionamento do cérebro humano que vem se mostrando de grande importância na resolução de problemas complexos reais (BASHEER, I.A; HAJMEER, 2000). A RNA tem sido amplamente evocada em diferentes áreas, como classificação, visão computacional e processamento de linguagem natural. Isso se dá pela sua grande

capacidade de aprendizado e generalização através de dados complexos. Em resumo, a RNA pode ser descrita como uma estrutura complexa conectada por elementos chamados neurônios, que são, por sua vez, conectados entre si.

O sistema nervoso possui complexas estruturas de funcionamento que carregam informação (BASHEER, I.A; HAJMEER, 2000). A comunicação entre esses neurônios ocorre por meio de uma estrutura muito bem definida, em que os dendritos recebem impulsos de entrada e são responsáveis por transmiti-los para o corpo celular através do axônio; aqui, esses impulsos se dividem e produzem sinais colaterais e sinapse. A figura 6 mostra a estrutura de um neurônio biológico.

Figura 6 – Estrutura de um neurônio biológico.

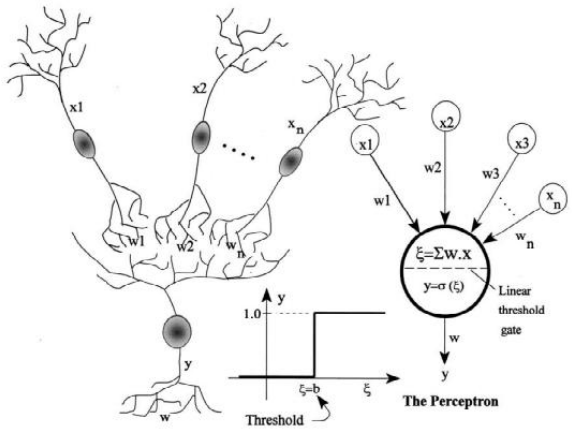


Fonte: Retirado de (BASHEER, I.A; HAJMEER, 2000).

A relação entre a RNA e a rede biológica ocorre em sua estrutura. A analogia é a de que ambas as redes possuem conexões entre dendritos e axônios, e suas comunicações ocorrem por meio do processo de sinapse. Na figura 7 é possível observar a estrutura baseada nesses três elementos principais. Nela, x_i, com $i \in [1, n]$, representam os sinais recebidos, e seus pesos sinápticos são representados por w_i. Esses pesos são responsáveis por determinar a influência dos sinais de entrada. Inicialmente,

os pesos possuem valores aleatórios e passam por ajustes iterativos com o objetivo de otimizar o desempenho do modelo, fazendo com que a saída predita corresponda à saída real, y.

Figura 7 – Estrutura comparativa entre RNA e RN.



Fonte: Retirado de (BASHEER, I.A; HAJMEER, 2000).

Um modelo para a atividade de um neurônio pode ser expresso pelas equações (2) e (3) abaixo. Na equação (2), z representa a entrada ponderada do neurônio, que é calculada através do somatório do produto entre os sinais de entrada (x_i) com os pesos sinápticos (w_i) somado ao *bias* (b). Vale ressaltar que esse cálculo expressa o comportamento de uma única unidade em uma camada da rede neural, que tipicamente é composta por várias unidades e por várias camadas. A saída do neurônio, representada por a em (3), é obtida com emprego de uma função de ativação, $f(\cdot)$.

$$z = \sum_{i=1}^{n} (x_i \cdot w_i) + b \tag{2}$$

$$a = f(z) \tag{3}$$

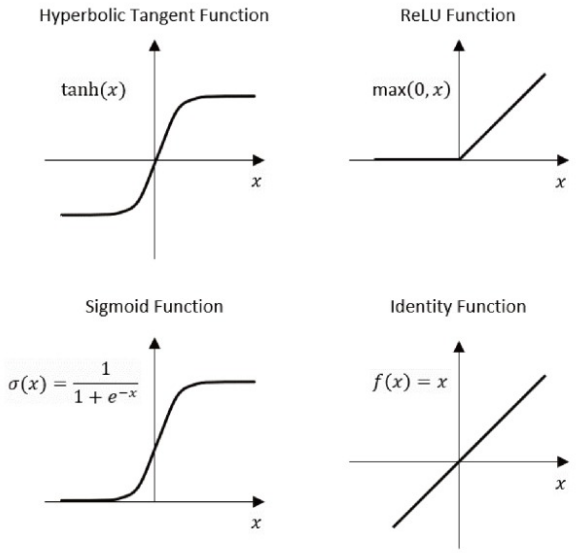
As redes neurais artificiais possuem diferentes funções de ativação a depender de cada problema. Na figura 8 é possível ver as principais funções (BASHEER, Imad; HAJMEER, M.N., 2001). Neste projeto, foi utilizada a função sigmoide (*sigmoid*), dada pela equação (4).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

3.3.4 KNN

O *K-nearest neighbor* (KNN) é um modelo amplamente empregado no reconhecimento de padrões e pode ser utilizado na resolução problemas de classificação

Figura 8 – Funções de ativação típicas.



Fonte: Retirado de (NISHIMOTO, 2018).

quando cada classe é representada por um conjunto de amostras. A ideia por trás do KNN é a de que as classes tendem a ter rótulos semelhantes e essa semelhança é identificada através da distância entre os *k*-vizinhos mais próximos. Vale ressaltar que quando uma nova amostragem de dados precisa ser classificada, seus *k*-vizinhos mais próximos são encontrados a partir de todos os vetores de classes conhecidas e o rótulo da classe é decidido com base na regra da maioria. Para evitar empates em regiões de sobreposição de classe, o valor de *k* deve ser ímpar. Esta regra é simples e elegante e a taxa de erro tipicamente é pequena na prática (LAAKSONEN; OJA, 1996).

O KNN opera de maneira semelhante em problemas de classificação multiclasse e em problema de classificação binária, utilizando abordagens *One-vs-All* ou *One-vs-One* para atribuir classes aos exemplos de teste.

Quando trata com classes desbalanceadas, o KNN pode ser impactado negativamente. O desequilíbrio entre o número de exemplos em diferentes classes pode levar a um viés na classificação, favorecendo as classes majoritárias em detrimento das minoritárias. Isso ocorre porque os exemplos de treinamento das classes majoritárias podem ser mais numerosos nas vizinhanças dos exemplos de teste.

Algumas técnicas para lidar com esse desafio podem ser a reamostragem dos dados para equilibrar as classes, o uso de ponderação para atribuir pesos diferentes às instâncias de diferentes classes e a aplicação de métodos de redução de dimensionalidade para melhorar a separabilidade das classes. No entanto, a escolha da abordagem mais adequada depende do contexto específico do problema e pode exigir

Figura 9 – Estrutura KNN.

Fonte: Retirado de (IBM, s.d.).

experimentação e validação cruzada para determinar a melhor estratégia.

Em resumo, o algoritmo KNN é uma abordagem de aprendizado de máquina que utiliza a vizinhança de exemplos de treinamento para fazer classificações. Ele tem vantagens, como a facilidade de implementação e o fato de não fazer suposições sobre a distribuição dos dados. No entanto, também possui desvantagens, como o alto custo computacional associado e a sensibilidade a ruídos e valores discrepantes. Em problemas multiclasse, o desbalanceamento das classes pode afetar o desempenho do KNN, sendo necessária a aplicação de técnicas específicas para lidar com esse desafio.

3.3.5 Modelo baseado em Árvore de Decisão

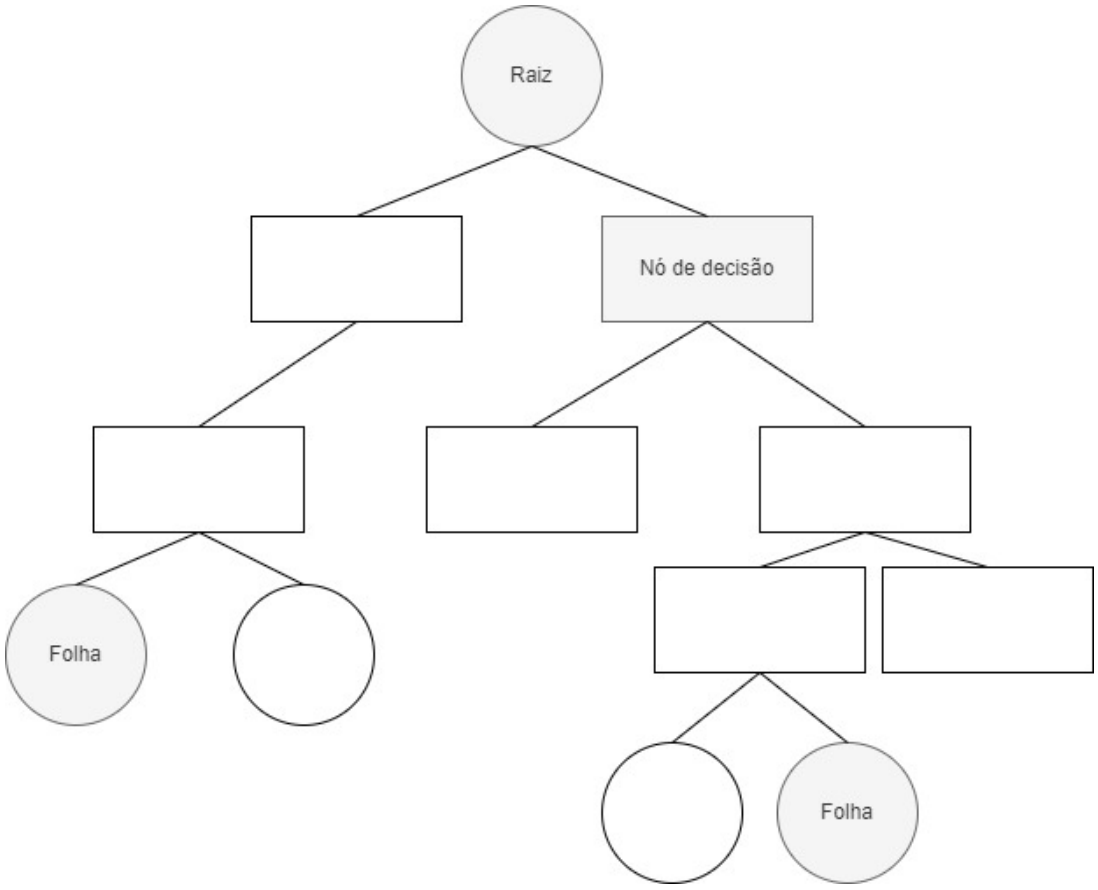
O modelo de aprendizado por árvore de decisão emprega aprendizado supervisionado e é amplamente aplicado na resolução de problemas de classificação. O algoritmo baseia-se em uma estrutura de árvore, composta por nós de decisão, ramos e folhas conectados entre si (MITCHELL, 1997).

Na figura 10 é possível ver a estrutura de uma árvore de decisão. Para problemas de classificação existe um ponto inicial, representado pela raiz. A ideia por trás dessa estrutura é a de que, a cada decisão tomada, adiciona-se um nó à árvore. Os nós de decisão são responsáveis por realizar teste de condições, fazendo assim a divisão em subgrupos menores.

Com o decorrer do algoritmo, os dados percorrem os ramos da árvore, sendo

direcionados pelos nós de decisão, que tendem a realizar agrupamentos específicos. Esse trajeto acaba assim que os dados chegam às folhas de decisão, que representam a classe de saída da predição. Assim, cada folha resultante equivale a uma saída do modelo (BENTO, 2021).

Figura 10 – Estrutura de árvore de decisão.



Fonte: Adaptado de (MITCHELL, 1997).

Uma das vantagens das árvores de decisão é capacidade de lidar com diferentes tipos de dados. Sua estrutura intuitiva permite fácil compreensão do processo de classificação utilizado pelo modelo. No entanto, as árvores de decisão podem sofrer com problema de *overfitting*. Ao construir uma árvore de decisão, o algoritmo utiliza um índice de impureza para aferir a incerteza presente nos dados. Uma medida comum de impureza é a entropia, que avalia a distribuição dos dados. Além disso, o ganho de informação é uma métrica importante nessa abordagem, pois quantifica a redução na entropia após a divisão dos dados (MITCHELL, 1997). No contexto deste projeto de classificação, a entropia (S) e o ganho (Δ) podem ser calculados usando as equações abaixo:

$$S = \sum_{i=1}^{c} -P_i \log P_i. \tag{5}$$

$$\Delta = H - \sum_{i=1}^{m} \frac{n_i}{N_i} \cdot H_i \tag{6}$$

- n é o número de classes;
- P_i é a proporção de instâncias da classe i em relação ao total de instâncias;
- N_i é o número total de instâncias;
- m é o número de divisões dos dados;
- n_i é o número de instâncias na divisão i;
- H_i é a entropia da divisão i.

3.3.6 Random Forest

O *Random Forest* é um algoritmo de ML do tipo *ensemble* que vem sendo amplamente utilizado na resolução de problemas de classificação. A proposta dessa abordagem é fazer a combinação de várias árvores de decisão, com o intuito de realizar uma predição mais robusta (PAUL *et al.*, 2018).

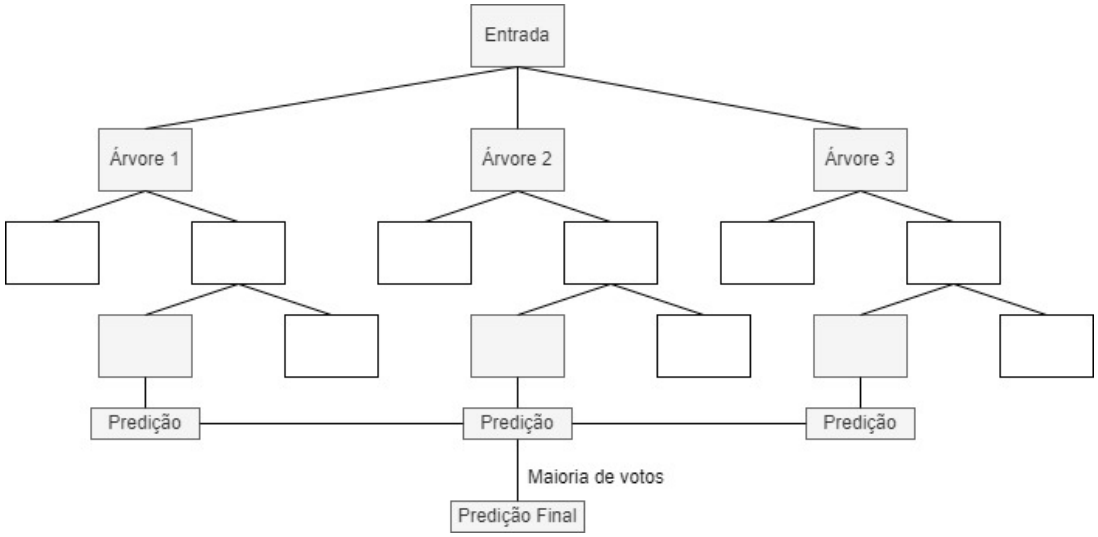
O modelo tem como vantagem a possibilidade de lidar com problema multiclasse, pois em sua estrutura as árvores são construídas e combinadas com aleatoriedade. Desta forma, cada árvore é treinada com um subconjunto aleatório, valendo-se de amostras e características distintas, o que faz com que o seu conjunto se proteja dos erros individuais (YIU, 2019).

Ao combinar as predições de várias árvores de decisão, o *Random Forest* é capaz de reduzir o impacto do desbalanceamento das classes. As árvores individuais podem capturar informações relevantes de classes minoritárias, contribuindo para a tomada de decisão coletiva do modelo. Além disso, o algoritmo atribui pesos às predições de cada árvore, considerando sua precisão e confiabilidade. Isso permite que o *Random Forest* leve em conta as diferenças de distribuição entre as classes e tome decisões mais equilibradas.

Em resumo, o *Random Forest* é uma escolha poderosa para a solução de problemas de classificação multiclasse com classes desbalanceadas, pois sua abordagem de *ensemble* e combinação de árvores de decisão independentes permitem lidar de forma eficaz com a disparidade na distribuição das classes, resultando em predições mais confiáveis e precisas.

A combinação das previsões das árvores individuais é feita por votação na classificação. Cada árvore emite sua própria predição e a classe mais frequente entre as

Figura 11 – Estrutura de *Random Forest*.



Fonte: Adaptado de (YIU, 2019).

árvores é escolhida como a predição final do *Random Forest*. Esse processo de votação permite que o algoritmo beneficie-se da sabedoria coletiva das árvores, reduzindo assim a probabilidade de que uma árvore específica seja dominante e enviesada em relação às classes majoritárias.

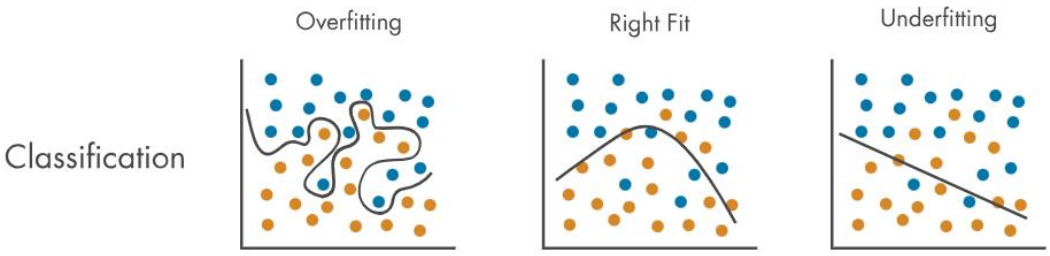
A figura 11 fornece uma ilustração do funcionamento do *Random Forest*, destacando as etapas de treinamento das árvores individuais e a combinação de suas predições. Essa figura mostra como as árvores são construídas e como suas predições são agregadas para obter uma predição final mais robusta e confiável.

3.4 VALIDAÇÃO DO MODELO

No *Pipeline* de desenvolvimento de projetos de ML, muitas vezes é realizada a divisão dos dados em dois conjuntos: o de treino e o de teste. Em alguns casos, há um terceiro conjunto para validação do modelo com o intuito de realizar as predições necessárias. Ao se fazer isso, é importante verificar se o modelo está sofrendo *underfitting* ou *overfitting* (BRONSHTEIN, 2017).

Underfitting ocorre quando o modelo não consegue se adequar aos dados de treinamento, perdendo a tendência deles. Por consequência, o modelo não consegue realizar uma generalização à entrada de novos dados. Em contraste, e como já foi previamente comentado, o *overfitting* ocorre quando modelo fica superajustado aos dados: de forma geral, o modelo aprendeu muito sobre dados de treinamento, e tem um desempenho praticamente perfeito, porém no conjunto de teste ou validação não consegue ter o mesmo desempenho (BRONSHTEIN, 2017). Na figura 12 é possível ver um exemplo da característica desses dois fenômenos.

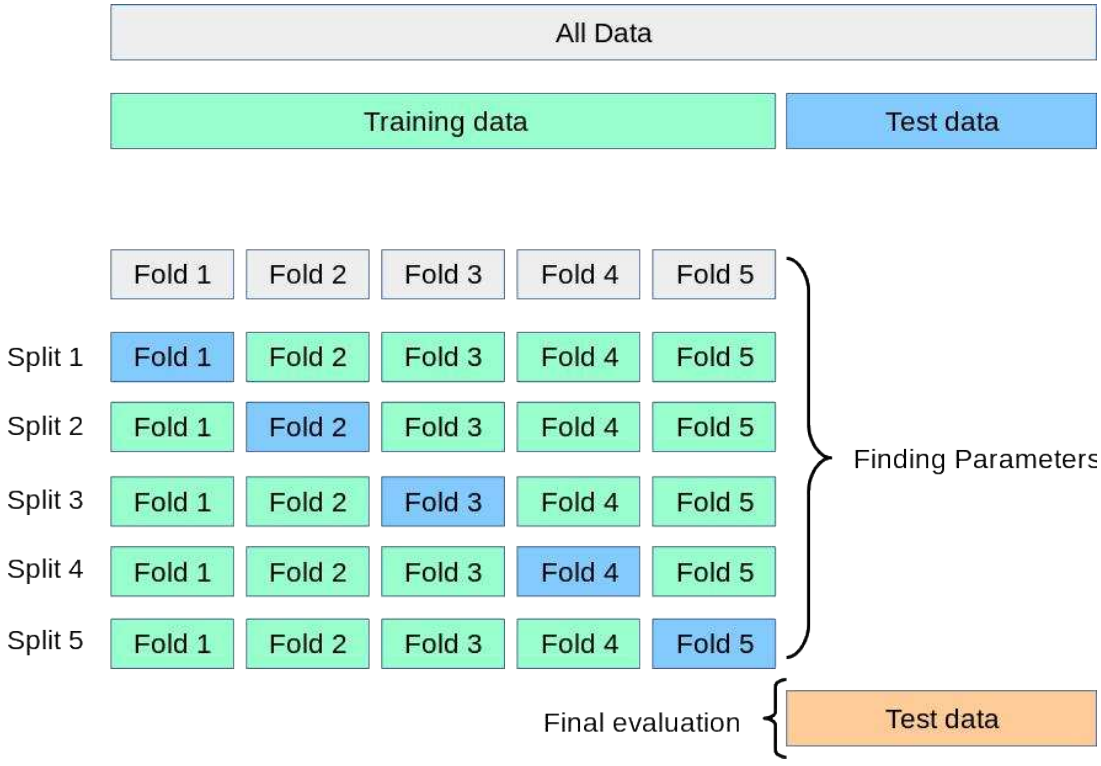
Figura 12 – Comparação entre *underfitting* e *overfitting* em modelos de classificação.



Fonte: Retirado de (MATHWORKS, 2023).

Uma técnica para ajudar na resolução dos problemas vistos anteriormente é a de validação cruzada (*cross validation*). Neste trabalho foi utilizado o método *K-fold*: o *dataset* é dividido K sub-conjuntos, chamados de *fold*. Na figura 13 pode-se ver uma ilustração do funcionamento do algoritmo, em que um sub-conjunto K é selecionado para teste e os $K - 1$ restantes são utilizados para treinamento. O modelo resultante é finalizado quando todos os conjuntos passaram pela fase de teste, criando uma medida de desempenho (SCIKIT-LEARN, Acesso em 2023).

Figura 13 – Processo de validação cruzada *K-fold*.



Fonte: Retirado de (SCIKIT-LEARN, Acesso em 2023).

4 ESPECIFICAÇÕES DO PROJETO

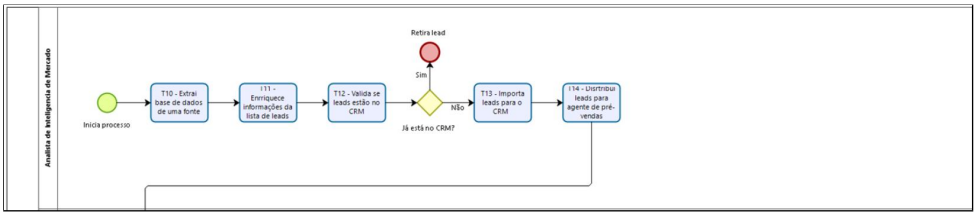
Neste capítulo, são apresentados o mapeamento atual do processo comercial e o modo de realização da aquisição dos dados para classificação durante o processo. Esta representação é feita através de uma modelagem *Business Process Modeling Notation* (Notação de modelagem de processo de negócio, doravante Notação de modelagem de processo de negócio (BPMN)). Em seguida, são apresentadas as especificações gerais do projeto, que tratam da avaliação de técnicas de aprendizado de máquina e da prova de conceito para utilização de *dashboard* para uso operacional. No apêndice A, são expostos o diagrama geral de desenvolvimento deste trabalho e a forma de *pipeline* baseada na metodologia CRISP-DM, que apresenta as principais etapas e a estrutura geral utilizada.

4.1 MAPEAMENTO DO PROCESSO

A empresa tem em uma das suas estratégias principais a metodologia *Outbound Marketing*, uma abordagem ativa sobre potenciais clientes que têm o perfil ideal (*Ideal Customer Profile*, ou *Ideal Customer Profile* (ICP)). Uma das vantagens dessa estratégia é a de que a empresa pode escolher quem vai abordar com base no perfil do potencial cliente.

A primeira etapa para a estratégia é a montagem de uma lista com informações confiáveis para uso dos times de prospecção e vendas (ROSS; TYLER, 2017) (figura 14). Essa etapa do projeto é feita por analistas do time de operações. A primeira atividade é a seleção da fonte de dados – que é pública – de onde posteriormente se extrairão essas informações. É na atividade *T*11, intitulada *Enriquece lista de leads*, que as primeiras informações utilizadas para classificação dos potenciais clientes são obtidas. As informações são o número de quartos, a taxa diária média (*Average Daily Rate* ou *Average Daily Rate* (ADR)), e o tipo de hospitalidade. Também são extraídos outros dados essenciais para a operação, como o *website*, o telefone e o endereço de e-mail da empresa em questão.

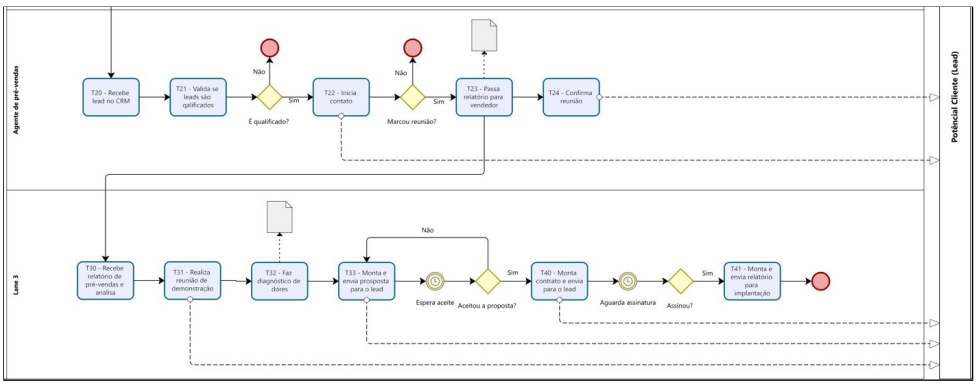
Figura 14 – *Lane* da geração de lista.



Fonte: Autor.

Na atividade de extração dessas informações é utilizado um sistema interno, que possui um conjunto de ferramentas para coleta dos dados de fontes públicas. Após finalizar todas as atividades de geração, os *leads* são distribuídos para os agentes de pré-vendas, que têm como responsabilidade validar as informações recebidas e executar a prospecção desses clientes, com objetivo de apresentar a empresa a eles e conectá-los com o agente de vendas. Na última *lane* do processo, o agente de vendas tem a responsabilidade de apresentar a solução da empresa e de coletar informações relevantes para a montagem de proposta e de dados. As atividades descritas podem ser vistas na figura 15. Atualmente, a variável *visitantes no site* é a última a ser coletada e é também a que impõe mais problemas na coleta, por depender de soluções terceirizadas ou de validação com potencial cliente.

Figura 15 – *Lanes* de pré-vendas e vendas.



Fonte: Autor.

4.2 REQUISITOS FUNCIONAIS

Os requisitos funcionais foram definidos com base nos principais desafios do processo já estabelecido e no problema principal do projeto, utilizando as técnicas de aprendizado de máquina avaliadas. Os requisitos funcionais são dados por:

1. utilizar base de dados presente no CRM de vendas;
2. realizar pré-processamento dos dados antes da utilização dos modelos de ML;
3. classificar os potenciais clientes;
4. avaliar o modelo mais adequado para realizar a classificação automática;
5. fazer backup das classificações para possíveis auditorias;
6. enviar as predições de forma automática para o CRM.

4.3 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais são enumerados abaixo:

1. a interface deve ser de fácil compartilhamento com membros da gestão;
2. o modelo utilizado para classificação deve ter acurácia miníma de 90%;
3. o modelo não deve apresentar *overfitting*;
4. devem-se utilizar ferramentas e recursos de tecnologias já utilizadas pela empresa.

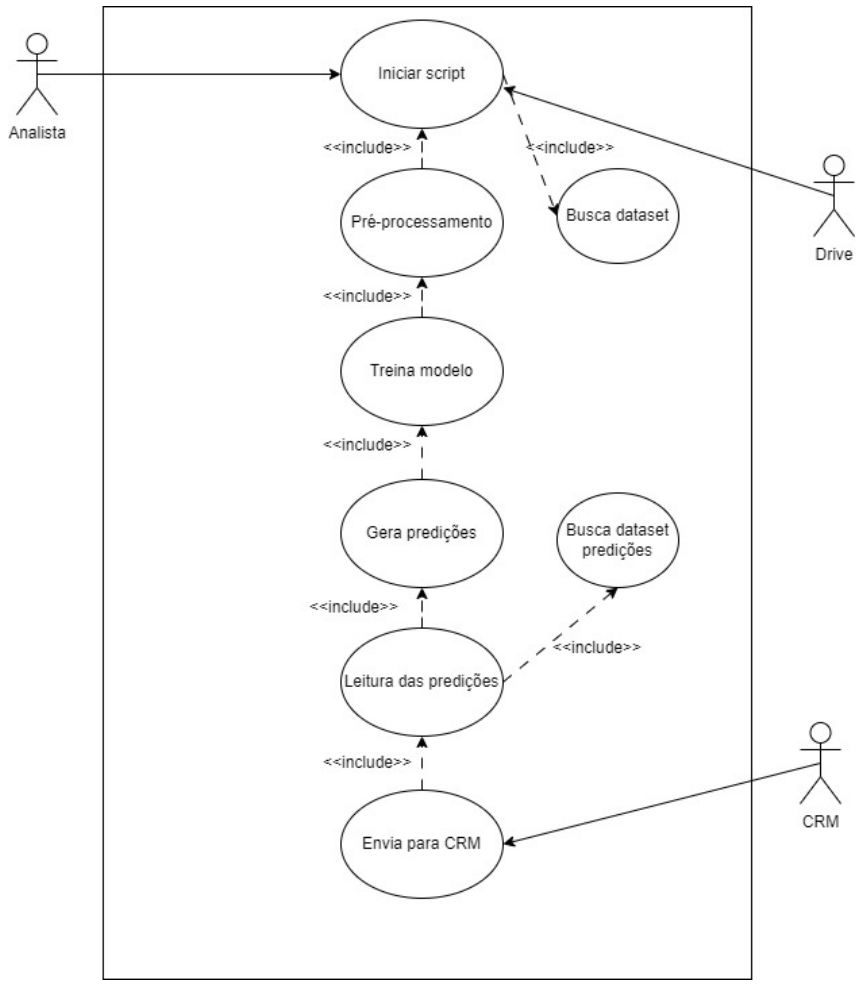
4.4 PROPOSTA DE SOLUÇÃO

Para construção da solução, propõe-se uma prova de conceito utilizando o ambiente *Google Colaboratory* (também referido aqui como *Colab*) para automatizar a classificação de potenciais clientes. Para apoiar a solução implementada é utilizado diagrama de caso de uso e de sequência. No primeiro diagrama de caso de uso, ilustrado na figura 16, é exposta a interação entre atores da solução. O ator principal, intitulado “Analista”, inicializa o *script*, que começa sua atividade pela busca da base de dados a ser utilizada na solução. Com a base de dados adicionada, são efetuados cinco comandos, incluindo pré-processamento dos dados, treinamento do modelo, realização das classificações, leitura das classificações resultantes e envio dessas informações para o CRM.

O segundo diagrama, representado pela figura 17, fornece uma visão geral dos eventos temporais e das interações entre os usuários do sistema. O analista inicia o processo acessando e carregando o *dataset* de interesse. Em resposta, o sistema obtém os dados e os envia para o ambiente do Google Colaboratory. Em seguida, os cinco comandos são executados de forma sequencial. O diagrama de sequência captura essa sequência temporal, destacando uma interação entre o usuário e o sistema em cada etapa do processo.

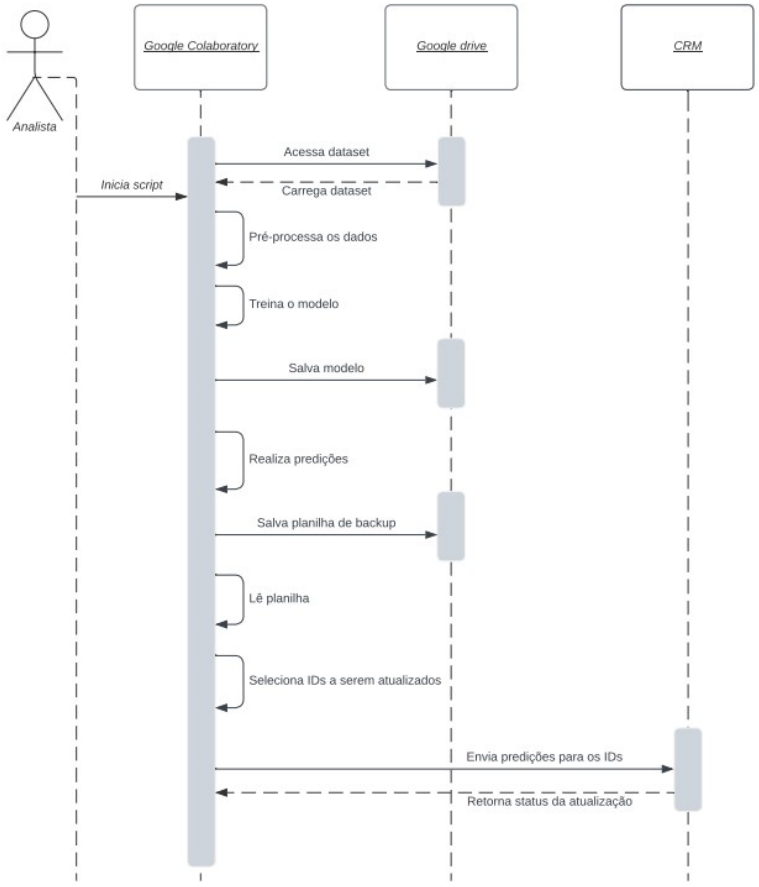
Esses dois diagramas combinados fornecem uma visão de solução clara e aberta para representar o fluxo de execução do processo definido. O diagrama de caso mostra os casos de uso relacionados, seus relacionamentos e os relacionamentos entre o usuário e o sistema. O diagrama de sequência também mostra as interações em ordem temporal, fornecendo uma imagem detalhada do fluxo do processo.

Figura 16 – Caso de uso.



Fonte: Autor.

Figura 17 – Diagrama de sequencia da proposta de solução.



Fonte: Autor.

5 DESENVOLVIMENTO

Nos capítulos anteriores foram vistas diferentes técnicas de aprendizado de máquina para resolução de problema de classificação, e a importância da realização de uma análise exploratória de dados em projetos de ML. Expôs-se também o embasamento teórico da aplicação desses métodos no fluxo de processo. Também foram expostos o mapeamento do processo atual da empresa, os requisitos do projeto e a prova de conceito.

Neste capítulo é apresentado o desenvolvimento de cada etapa do projeto de forma geral. Aqui são comentadas as principais ferramentas utilizadas nele, assim como a forma como os dados foram obtidos. Também são mostradas a análise exploratória, o desenvolvimento e a avaliação dos modelos e técnicas de ML utilizados. Além disso, é apresentada uma prova de conceito de como essas técnicas podem ser implementadas no processo atual da empresa. Por fim, é discutido o desenvolvimento de um *dashboard* para uso operacional e estratégico.

5.1 FERRAMENTAS UTILIZADAS

Esta seção descreve de forma breve as principais ferramentas empregadas durante o desenvolvimento deste trabalho.

5.1.1 Metabase

Para se obter a centralização dos dados utilizados no projeto, foi utilizada a ferramenta *Metabase*, que tem a capacidade de concentrar dados de diferentes fontes e realizar pesquisas (Structured Query Language). A ferramenta também possui uma série de conexões com bancos de dados e serviços de armazenamento em nuvem, de modo que essa centralização permite fácil acesso a informações necessárias para o projeto.

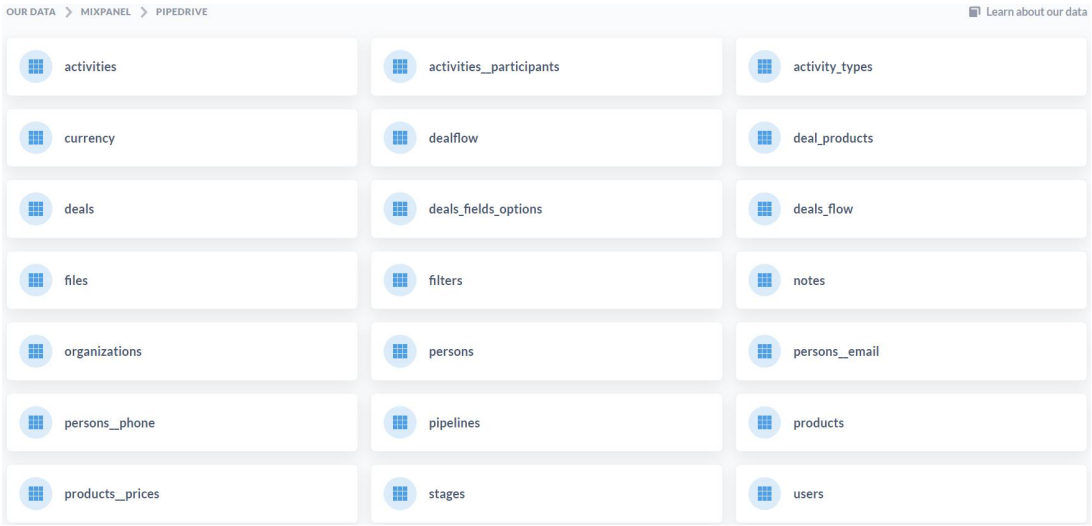
Neste projeto, foram utilizados os dados obtidos via integração do CRM, no qual estão concentrados os dados de interesse para construção do *dataset*. O esquema de tabelas do CRM pode ser visto na figura 18.

5.1.2 Google Colaboratory

O Goole Colaboratory foi a ferramenta empregada no desenvolvimento dos algoritmos desenvolvidos neste projeto. A ferramenta permite desenvolver e compartilhar códigos em Python, que foi a linguagem de programação utilizada.

O uso dessa ferramenta se fez interessante uma vez que ela utiliza a estrutura do *Google Drive* com acesso a *Graphics Processing Unit* (GPU) e *Tensor Processing Unit* (TPU), permitindo treinamento de modelos de ML. Além disso, ela já possui

Figura 18 – Coleção de dados do CRM.



Fonte: Autor.

bibliotecas pré-instaladas, como *Scikit-Learn*, que foram utilizadas no desenvolvimento do projeto.

Ademais, a importância do Google Colab no desenvolvimento de ML é significativa. O recurso remove muitas barreiras e complexidades associadas à configuração de ambientes de desenvolvimento e hardware de alto desempenho. Com o Colab, é possível começar rapidamente a explorar dados, prototipar modelos e treiná-los em escala. Além disso, o Colab facilita a colaboração e o compartilhamento de projetos com outras pessoas.

5.1.3 Scikit-learn

Scikit-learn, também conhecido como *Sklearn*, é uma biblioteca em Python amplamente utilizada em projetos que envolvem ML. Ela fornece uma ampla gama de algoritmos e ferramentas para realizar tarefas comuns de aprendizado de máquina, como classificação, regressão, agrupamento, pré-processamento de dados e seleção de modelos.

A principal força do Scikit-learn é sua facilidade de uso e sua integração com o ecossistema de bibliotecas científicas do Python, como *NumPy* e *Pandas*. Ele oferece uma interface consistente e intuitiva para treinar modelos de aprendizado de máquina, avaliar seu desempenho e fazer classificações e classificações com novos dados.

A biblioteca Scikit-learn possui uma ampla variedade de algoritmos implementados, incluindo algoritmos populares como regressão linear, regressão logística, árvores de decisão, *random forest*, máquinas de vetores de suporte (da sigla em inglês SVM), redes neurais e muitos outros. Esses algoritmos são altamente otimizados e eficientes,

permitindo o processamento de grandes volumes de dados.

Além dos algoritmos de aprendizado de máquina, o Scikit-learn também oferece ferramentas para pré-processamento de dados, como tratamento de valores ausentes, codificação de variáveis categóricas e dimensionamento de recursos. Ele também fornece recursos de seleção de modelo, como validação cruzada, busca de hiperparâmetros e avaliação de desempenho.

A importância do Scikit-learn no desenvolvimento de aprendizado de máquina é significativa. Ele simplifica o processo de desenvolvimento de modelos, fornecendo uma interface consistente e abstrações de alto nível. Além disso, o Scikit-learn é amplamente adotado pela comunidade de aprendizado de máquina e é bem documentado, o que facilita a aprendizagem e o compartilhamento de conhecimentos.

5.2 AQUISIÇÃO DOS DADOS

Para construção de um *dataset*, foi realizada uma consulta também conhecida como Linguagem de Consulta Estruturada em português) na base de dados do CRM presente na ferramenta Metabase. Os dados requisitados para utilização no projeto, bem como sua descrição, podem ser vistos no quadro 1. Vale ressaltar que alguns dados foram utilizados em todo o projeto, enquanto outros foram utilizados como apoio para a realização de outras execuções. Também é importante dizer que o dado pode ser numérico ou categórico.

Quadro 1 – Descrição dos dados.

Variável	Descrição	Tipo
ID	Dado para conseguir fazer mapeamento e enviar dados de saída para CRM e Sistema de BI	Numérico
Rooms	Número de quartos do hotel	Numérico
ADR	*Average Daily Rate*, receita média por um quarto ocupado por dia	Numérico
Website visitors	Quantidade de visitantes únicos no site do hotel	Numérico
Company type	Segmento de atuação de potenciais clientes definidos pela empresa	Categórico
Size	Variável de saída para classificação do porte dos potenciais clientes	Categórica
Price	Valor de mensalidade	Numérico

Fonte: Autor.

Durante a análise dos dados foi identificada uma anomalia significativa para o projeto: cerca de 60% dos potenciais clientes que tinham registro no CRM apresentavam campos com valores nulos ou com má formatação. Para valores faltantes ou com erro de preenchimento, uma solução encontrada foi a utilização de uma plataforma

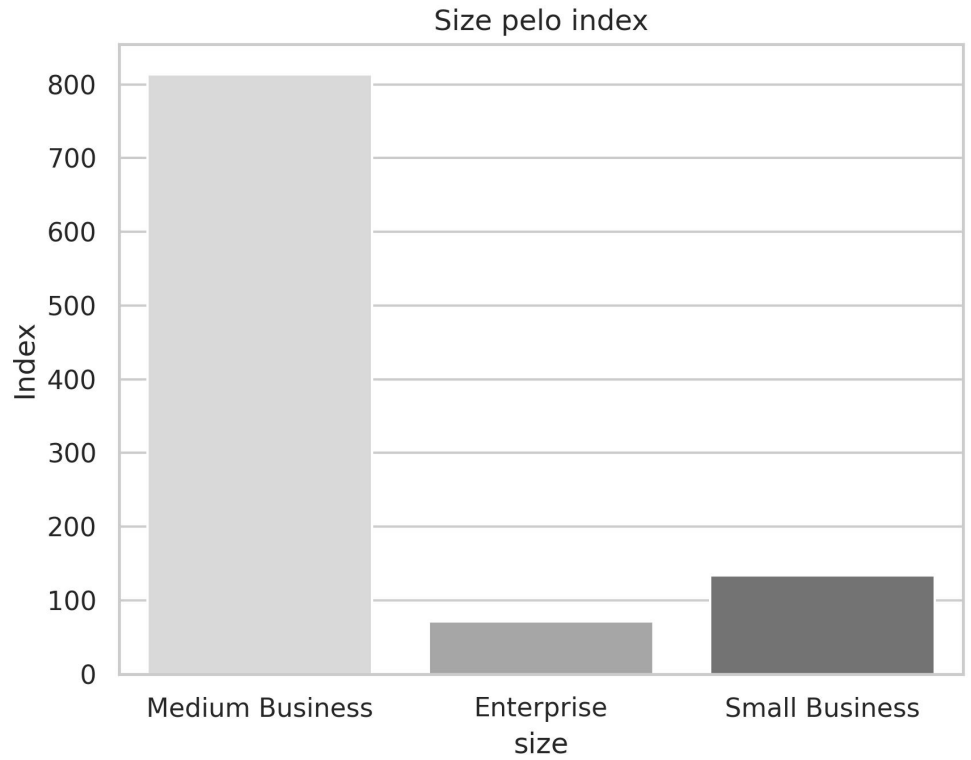
interna de extração de dados para enriquecer a base de dados. Este sistema funciona como um conjunto de ferramentas para extração e validação de dados disponíveis em fontes públicas.

5.3 ANÁLISE EXPLORATÓRIA

Para o bom desenvolvimento do *pipeline* deste projeto, foi essencial o uso de técnicas de EDA. Com a utilização dessas técnicas, foi possível identificar características de importante entendimento para a escolha e avaliação dos modelos de aprendizado de máquina.

O primeiro passo foi verificar a distribuição da variável de interesse *Size*. Observando o gráfico da figura 19, uma característica se evidencia: a classe *Medium* apresentou um maior número de registros em comparação com as classes *Small* e *Enterprise*. Como visto na seção 3.2, um conjunto com classes desbalanceadas pode ter impacto considerável no desempenho de modelos de ML. Na seção 5.4 é apresentado o desenvolvimento de algumas técnicas para lidar com esse tipo de problema.

Figura 19 – Distribuição da variável *Size* no *dataset*.



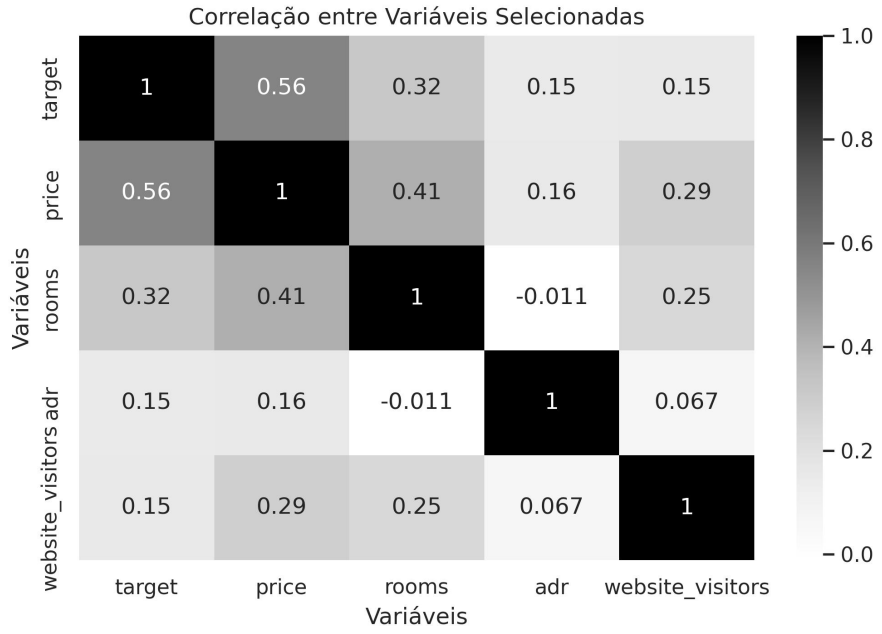
Fonte: Autor.

Outro passo importante foi verificar a correlação entre as variáveis. Para realizá-la, foi necessária a criação de rótulos de dados no *dataset* para a variável de interesse

Size e para de entrada *Type*, uma vez que as duas são categóricas. Para a primeira, foram criados rótulos, e para a segunda foi implementado o método *get dummies* da biblioteca Pandas para transformar os atributos em valores binários.

Para testar a correlação entre as variáveis usou-se *heatmap* e o resultado pode ser visto na figura 20. Analisando a figura, pode-se notar que as variáveis de entrada não têm uma correlação linear forte com a de saída *target* (alvo), de modo que a relação mais próxima de linearidade foi a variável *Price* (preço).

Figura 20 – heatmap entre váriveis.



Fonte: Autor.

5.4 DESENVOLVIMENTO DOS MODELOS DE ML

Nesta seção é apresentado as principais etapas no desenvolimento dos modelos. Estas etapas se basearam no ciclo de vida *Cross Industry Standard Process for Data Mining* (CRISP DM) apresentado na seção 1.3. Onde temos o pré-processamento de dados, modelagem dos modelos utilizados neste projeto e suas avaliações.

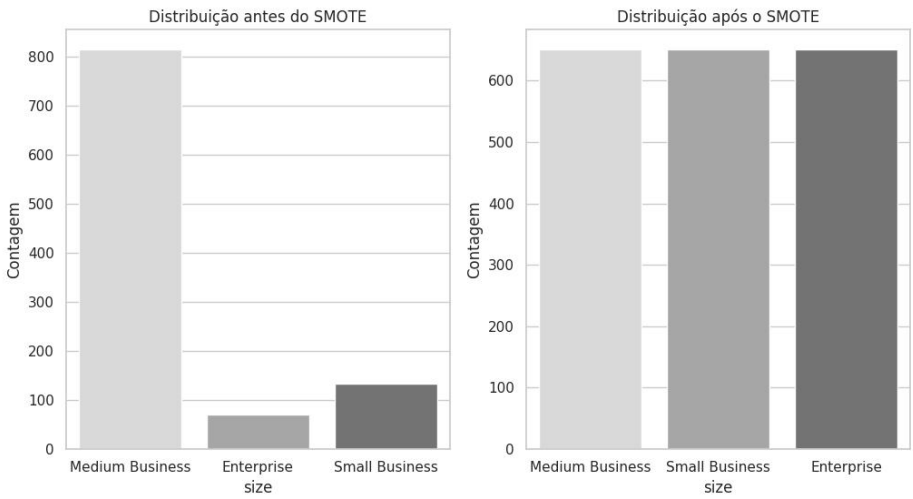
5.4.1 Pré-processamento

Para desenvolvimento dos modelos, o conjunto de dados foi inicialmente dividido em dois conjuntos: o de treino, com representatividade de 70%, e o de teste, de 30%. Essa separação foi realizada utilizando o método *train-test-split* da biblioteca Scikit-learn, que permite uma divisão adequada do conjunto de dados. O conjunto de

treinamento é utilizado para treino dos modelos avaliados, enquanto o de teste para a realização da validação dos modelos, verificando seus resultados e capacidade de uso no processo de classificação. Além disso, para aprofundamento na capacidade dos modelos, foram feitos testes experimentais utilizando uma divisão de três conjuntos, sendo, 70% para treino, 20% para teste e 10% para validação.

Na etapa de pré-processamento, também foi empregada em paralelo a técnica SMOTE para a coleta de resultados experimentais, o que permitiu comparar o desempenho dos modelos com e sem a técnica. Para isso, foi utilizada a biblioteca *Imbalanced learn*, que permite realizar o *oversampling* nas classes minoritárias do conjunto de dados. Para parametrização da técnica, foram utilizadas configurações padrão com $k = 5$ vizinhos e balanceamento automático. Na figura 21 é possível ver o antes e depois do conjunto de treino.

Figura 21 – Comparação utilizando *oversampling*.



Fonte: Autor.

5.4.2 Desenvolvimento geral dos modelos

Para desenvolvimento dos modelos empregados no projeto, os hiperparâmetros de cada modelo foram testados de forma experimental. Após esses testes, também se fez o uso do ajuste desses hiperparâmetros utilizando a classe *Randomized SearchCV* da biblioteca Scikt-learn. Essa classe realiza uma validação cruzada aleatória nos hiperparâmetros, com objetivo de encontrar a combinação com melhor acurácia para os modelos em que a classe é utilizada (PEDREGOSA *et al.*, 2023).

Como apresentado na seção 3.3.1, o modelo de regressão logística é indicado para resolução de problema de classificação binária. Porém, através da classe *LogisticRegression*, a implementação da estrutura *One Versus Rest* (OvR) 3.3.2 é feita na

configuração de hiperparâmetros, adaptando o modelo de problemas multiclasse em classificação binária. Similar à regressão logística, o modelo KNN também necessita da estrutura OvR, que é implementada de forma automática quando o número de classes é maior do que dois.

Para desenvolvimento do modelo de rede neural artificial, foram necessários alguns testes experimentais para parametrização. Os hiperparâmetros utilizados tiveram suas definições através da classe *Randomized SearchCV* foram 100 neurônios na camada oculta e número de máximo de interações igual 1000, o algoritmo ajusta os pesos da rede neural em cada iteração com o objetivo de minimizar a função de perda. O número máximo de iterações especifica quantas vezes esse processo de ajuste será repetido. A função de ativação utilizada na camada oculta foi a *logistic*, que retorna a função *sigmoid* apresentada na expressão 4.

Por fim, os modelos *decision tree* e *random forest* tiveram seus hiperparâmetros definidos diretamente através da classe Randomized SearchCV. Para o primeiro modelo, alguns parâmetros como amostras mínimas por folha e profundidade máxima da árvore foram selecionadas. Para o segundo, número de árvores na *random forest*, amostras mínimas para dividir um nó e profundidade máxima foram alguns parâmetros utilizados. A parametrização completa de cada modelo pode ser observada com mais detalhes no apêndice B.

5.4.3 Avaliação dos modelos

Na seção 5.4, foi apresentado o desenvolvimento dos modelos selecionados para este projeto. Para avaliar qual modelo é mais adequado para resolver o problema definido, foram consideradas diferentes métricas de desempenho para modelos de aprendizado de máquina. Neste projeto, as seguintes métricas foram utilizadas: acurácia, *recall*, precisão e *F1-Score*. Também foi realizada a construção de matriz de confusão de cada modelo, para se ter visualização do desempenho das classificações.

A acurácia (A) permite medir a proporção em que um modelo está acertando em relação ao total de classificações realizadas. Neste trabalho, como existem três classes, sendo que uma delas é predominante perante as outras, apenas o uso dessa métrica não é suficiente para uma boa avaliação, uma vez que a classe dominante pode ter uma influência em seu resultado. A acurácia é definida por:

$$A = \frac{T_P + T_N}{T_P + F_N + F_P + T_N}, \tag{7}$$

em que

- T_P (Verdadeiro Positivo): indica o número de observações corretamente classificadas como positivas pelo modelo;

- T_N (Verdadeiro Negativo): indica o número de observações corretamente classificadas como negativas pelo modelo;
- F_P (Falso Positivo): indica o número de observações erroneamente classificadas como positivas pelo modelo. Esses são os casos em que o modelo classificou incorretamente uma observação negativa como positiva;
- F_N (Falso Negativo): indica o número de observações erroneamente classificadas como negativas pelo modelo. Esses são os casos em que o modelo classificou incorretamente uma observação positiva como negativa.

O *recall* (R) determina a fração de amostras de uma classe que são preditas corretamente pelo modelo, dada pela equação (8).

$$R = \frac{T_P}{T_P + F_N}. \tag{8}$$

A precisão (P) mede a proporção de instâncias classificadas corretamente como positivas em relação ao total de instâncias previstas como positivas. Essa métrica é útil para avaliar a capacidade do modelo em evitar falsos positivos.

$$P = \frac{T_P}{T_P + F_P}. \tag{9}$$

O *F1-Score* (F_1) é uma métrica combinada que considera tanto a precisão quanto o *recall*. É a média harmônica dessas duas métricas e fornece uma única medida de desempenho que leva em consideração tanto falsos positivos quanto falsos negativos. O *F1-Score* é útil quando se deseja equilibrar a precisão e o *recall*. Ele é definido como:

$$F_1 = \frac{2PR}{P + R}. \tag{10}$$

A matriz de confusão não é uma métrica, mas tem relevância na avaliação dos modelos. A matriz consiste em uma representação matricial que mostra as classificações corretas e incorretas do modelo. No caso de um problema multiclasse, no qual tem-se três classes de interesse, a matriz é representada por uma dimensão 3×3. Cada linha da matriz de confusão representa a classe verdadeira, enquanto cada coluna representa a classe predita pelo modelo. Os elementos da diagonal principal representam os casos em que a classe verdadeira e a classe predita são iguais. Os elementos que estão fora da diagonal principal representam os casos em que ocorrem classificações incorretas.

Para a utilização dessas métricas, a biblioteca Scikit-learning possui uma classe contendo cada uma das métricas utilizadas neste projeto, permitindo uma avaliação ágil dos modelos. Os resultados de cada modelos são apresentados no capítulo 6.

5.5 SOLUÇÃO PARA USO NO PROCESSO

Para implementação da prova de conceito, optou-se pela utilização do modelo Random Forest devido a seus resultados, que são apresentados com mais detalhes no capítulo 6. Essa abordagem se mostrou promissora, devido à sua robustez na resolução de problemas multiclasse com dados desbalanceados. A estrutura da prova de conceito segue o fluxo do projeto apresentado no apêndice A, de modo que o usuário deve atuar no ambiente do Google Colaboratory para realizar o processo de classificação utilizando o algoritmo do modelo para gerar as classificações.

Para desenvolvimento foi aproveitado o desenvolvimento inicial do modelo *random forest*, em que são realizados o pré-processamento dos dados e a divisão dos conjuntos. Para salvar o modelo treinado é utilizada a biblioteca *Joblib*, que possui duas funções principais na prova de conceito: salvar o modelo e carregá-lo para a realização de classificações em um novo conjunto de dados. É importante ressaltar, que para o carregamento de um novo *dataset*, esse precisa ter a mesma formatação do *dataset* utilizado para treino, caso contrário não será possível garantir a execução da solução.

Ao realizar as classificações, é gerado um arquivo *Comma Separated Values* (CSV) (Comma Separated Values, em português, valores separados por vírgulas) que possui dois objetivos: o primeiro é ser utilizado como *backup* caso haja necessidade de algum ajuste manual; o segundo é ser empregado na leitura dos dados para envio automático ao CRM. Esse processo ocorre através da biblioteca *Requests*.

Figura 22 – Estrutura de código para mapeamento das classificações

```
# Leitura do arquivo CSV com as predições
csv_file = 'hotel_predictions.csv'
csv_reader = pd.read_csv(csv_file)

# Iterar sobre as linhas
try:
    for row in csv_reader.itertuples(index=True, name='Pandas'):
        org_id = row.org_id
        rating = row.predictions
        cluster_id = '26f88fe5462396af584f9007c03c65f5f14ea46a'

        # Mapeamento das predições e os IDs no campo cluster
        if rating == 'Enterprise':
            cluster_id = 1088
        elif rating == 'Medium Business':
            cluster_id = 1089
        elif rating == 'Small Business':
            cluster_id = 1090
        else:
            print(f"Invalid rating: {rating}. Skipping the row.")
            continue
```

Fonte: Autor.

A biblioteca *Requests* permite enviar solicitações *Hypertext Transfer Protocol* (HTTP) interagindo com a API fornecida pelo CRM para utilização do método *PUT* na atualização dos dados. O HTTP é um conjunto de protocolos usado na comunicação

de um cliente e um servidor *web*. A utilização de uma chave de *Application Programming Interface* (API), conhecida em português como Interface de Programação de Aplicativos, válida é necessária para realizar a autenticação entre o algoritmo e a API.

Figura 23 – Estrutura de código para envio da solicitação *PUT* para a API do CRM

```
# Construir payload para a API do Pipedrive
        payload = {
            'chave_id_do_campo_cluster': cluster_id
        }

        endpoint =
f'https://api.pipedrive.com/v1/organizations/{org_id}?api_token=token_do_C
RM'

        print(endpoint)
        print(payload)

        # Envio da solicitação PUT para atualizar o cluster no Pipedrive
        response = requests.put(endpoint, data=payload)
```

Fonte: Autor.

Por fim, o código lida com exceções genéricas. Ao realizar as solicitações *PUT*, é impressa uma mensagem de sucesso para toda atualização bem-sucedida.

Figura 24 – Tratamento de exceção e mensagem de retorno

```
        # Envio da solicitação PUT para atualizar o cluster no Pipedrive
        response = requests.put(endpoint, data=payload)

        # Verifica o status da resposta
        response_data = response.json()
        if 'data' in response_data and 'id' in response_data['data']:
            updated_org_id = response_data['data']['id']
            if updated_org_id == org_id:
                print(f"Cluster for organization {org_id} was successfully
updated in Pipedrive.")
            else:
                print(f"Cluster update for organization {org_id} in
Pipedrive returned unexpected data.")
        else:
            print(f"Failed to update cluster for organization {org_id} in
Pipedrive. Response data: {response_data}")

except Exception as e:
    print(f"An error occurred: {str(e)}")
```

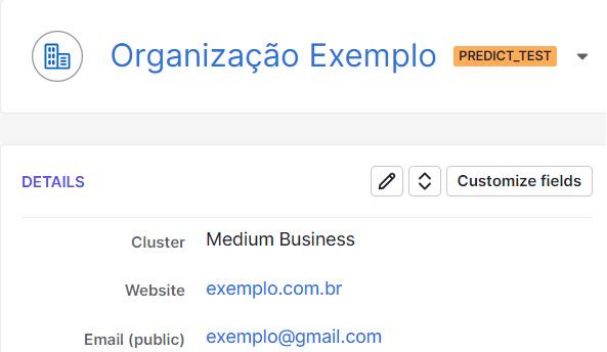
Fonte: Autor.

Já no CRM é utilizada uma estrutura de automação *low-code* para adicionar uma etiqueta a cada nova informação recebida da classe. Essa identificação foi importante para uso na etapa de criação do *dashboard*. A figura 25 reproduz a etiqueta desenvolvida no CRM.

5.6 *DASHBOARD* PARA AVALIAÇÃO DE DADOS

Foi desenvolvido um *dashboard* para monitorar e analisar potenciais clientes, cuja classificação foi determinada por um modelo de ML. O *dashboard* oferece uma visão detalhada dos *leads* presentes no CRM, mostrando a quantidade de potenciais

Figura 25 – Etiqueta de identificação no CRM



Fonte: Autor.

clientes de cada classe, quantos estão em aberto e quantos se tornaram efetivamente clientes, e comparando esses resultados com as expectativas iniciais de vendas.

Para essa etapa do projeto, utilizou-se a plataforma *Google Data Studio*, que proporciona uma variedade de conexões com a base de dados e oferece visualizações interativas e de fácil acesso, permitindo uma interpretação simplificada dos dados pelos usuários. Após enviar as classificações para o CRM, é estabelecida uma conexão com o banco de dados *Postgres* por meio de consulta SQL.

No ambiente do Google Data Studio foram construídos os seguintes indicadores de desempenho: quantidade de *leads* abertos, quantidade de *leads* ganhos, potencial de venda baseado no *ticket* médio histórico da classe e quantidade de venda gerada por classe. Para a construção desses indicadores, foi utilizada a função *case*, com exemplos de uso ilustrados na figura 26.

Vale ressaltar que para a construção do *dashboard* foram utilizadas as classificações geradas pela prova de conceito, porém seus dados foram transformados de modo a preservar o sigilo das informações da empresa. Os indicadores apresentados posteriormente na seção de 6.4 são sintéticos, para poder mostrar os resultados de desenvolvimento.

Figura 26 – Exemplo de uso da função *case* no projeto

```
#exemplo uso case para soma do potencial de venda por classe
 SUM(
  CASE
    WHEN Organization - Cluster = 'Small Business'
    THEN Average ticket expected
    ELSE 0
  END
)

#exemplo uso case para quantidade total de leads de uma classe
CASE
WHEN Organization - Cluster = 'Medium Business'
Then 1
ELSE 0
END

#exemplo uso case para identificar o status dos leads no CRM
CASE
  WHEN Organization - Cluster = 'Enterprise' and Deal - Status='Won'
  THEN 1
  ELSE 0
END
```

Fonte: Autor.

6 ANÁLISE DE RESULTADOS

Neste capítulo, são apresentados os resultados das etapas propostas no projeto. São demonstradas algumas características observadas na EDA, bem como sua importância para o trabalho. São também demonstrados, de forma comparativa, os resultados obtidos dos modelos de ML. Por fim, são apresentados o resultado da prova de conceito para utilização no processo de classificação e o *dashboard* para uso operacional e estratégico.

6.1 RESULTADOS DA EDA

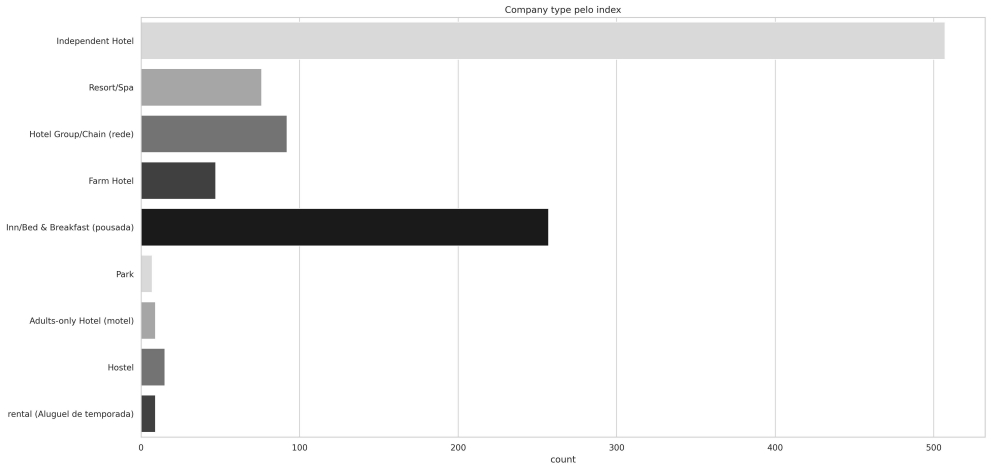
A etapa de EDA teve grande importância no projeto. A partir dessa exploração, foi possível conhecer características importantes dos dados que estavam registrados no CRM. Uma observação muito importante a se fazer referente ao *dataset* trabalhado é a existência de grande dispersão nos dados de entrada que seriam utilizados no modelo. As variáveis de entrada possuem uma característica de distribuição de cauda longa. Esse tipo de distribuição ocorre porque dados extremos são identificados no *dataset*, resultando nessa característica. É importante salientar que esse tipo de distribuição pode ter impactos em modelos de ML, fato considerado no desenvolvimento dos modelos utilizados neste projeto.

Tabela 1 – Estatísticas do conjunto de dados

Índice	**price**	**adr**	**rooms**	**website_visitors**
count	1019,00	1019,00	1019,00	1019,00
mean	720,27	385,39	67,86	5077,54
std	829,37	669,71	83,14	19105,56
min	1,00	10,00	5,00	1,00
25%	400,00	172,00	19,00	452,00
50%	499,00	239,00	42,00	1152,00
75%	759,50	381,50	87,00	3499,00
max	13582,00	12400,00	912,00	411176,00

Uma hipótese inicial do projeto era a de que as variáveis tivessem uma correlação linear entre si. No entanto, ao observar a figura 20, apresentada anteriormente, é possível ver que isso não ocorre. Apenas a variável *Price* apresenta uma correlação linear significativa. Essa descoberta foi extremamente importante para a escolha dos modelos, uma vez que modelos capazes de lidar com problemas de não linearidade seriam mais adequados para a realização da classificação de múltiplas variáveis. Vale destacar que as características encontradas na etapa de EDA foram essenciais para uma maior clareza na utilização de modelos de ML e serviram como um sinal de melhoria para o processo de geração de dados no CRM.

Figura 27 – Distribuição *company type* pelo *index*.



Fonte: Autor.

6.2 RESULTADOS DOS MODELOS DE APRENDIZADO DE MÁQUINA

Os resultados dos modelos de ML se mostraram satisfatórios ao obter métricas de performance gerais. Os modelos Regressão Logística, Árvore de Decisão e *Random Forest* obtiveram acurácia acima de 90%, o que era um dos requisitos do projeto, mesmo sem a utilização da técnica SMOTE. Os resultados experimentais dos modelos sem utilização de uma técnica de balanceamento de dados podem ser observados na tabela 2. Com a utilização da técnica SMOTE, foi obtida uma melhora nos resultados dos modelos Regressão Logística e KNN, que podem ser vistos na tabela 3. Ressalta-se que o modelo Regressão Logística mostrou-se um bom candidato ao uso na prova de conceito.

Tabela 2 – Performance geral dos modelos sem técnica SMOTE

Modelo	**Acurácia**	**Precisão**	**Recall**	**F1-Score**
Regressão Logística	0,921	0,928	0,921	0,912
Árvore de decisão	0,975	0,975	0,977	0,975
Random Forest	0,983	0,983	0,983	0,983
KNN	0,898	0,897	0,898	0,893
RNA	0,826	0,822	0,826	0,788

Mesmo que os resultados globais observados tenham sido satisfatórios, houve um aprofundamento de análise para se entender o desempenho dos modelos em cada classe alvo do projeto. Quando é observada a acurácia, medida geral da precisão do modelos, todos os modelos apresentaram bons resultados, variando entre 80% e 98%.

Tabela 3 – Performance geral dos modelos com técnica SMOTE

Modelo	**Acurácia**	**Precisão**	**Recall**	**F1-Score**
Regressão Logística	0,950	0,954	0,893	0,952
Árvore de decisão	0,980	0,980	0,977	0,980
Random Forest	0,983	0,980	0,983	0,983
KNN	0,859	0,899	0,898	0,869
RNA	0,750	0,757	0,741	0,753

Isso indica que, em média, a classificação da maioria das amostras está sendo feita corretamente. No entanto, como as classes são desbalanceadas, a acurácia pode não ser a melhor métrica para avaliar o desempenho do modelo. Portanto, é importante examinar as métricas por classe para se obter *insights* mais detalhados.

Analisando as métricas por classe, foi possível observar diferenças significativas no desempenho dos modelos para cada uma delas. Inicialmente foi analisado o desempenho sem a utilização da técnica SMOTE. Para a classe *Enterprise*, o modelo Regressão Logística apresenta uma precisão perfeita de $1,0$, o que significa que todas as previsões positivas estão corretas. No entanto, o *recall* é de apenas $0,7$, indicando que o modelo deixou de identificar corretamente 30% das amostras dessa classe. O *F1-score* para essa classe é de $0,8$, que é uma média ponderada de precisão e *recall*. Portanto, embora o modelo apresente uma alta precisão para essa classe, é necessário aperfeiçoar o *recall*.

Para a classe *Medium Business*, os modelos Árvore de decisão, *Random Forest* e KNN têm um desempenho geralmente bom, com valores de precisão, *recall* e *F1-score* acima de $0,9$. Essas métricas indicam que os modelos estão classificando corretamente a maioria das amostras dessa classe. No entanto, o modelo RNA apresenta uma precisão de $0,8$ e um *F1-score* de $0,89$, indicando um desempenho inferior para essa classe.

Para a classe *Small Business*, os modelos Regressão Logística , Árvore de decisão e *Random Forest* têm uma precisão perfeita de $1,0$, o que significa que todas as previsões positivas estão corretas. No entanto, o *recall* é relativamente baixo, variando de $0,53$ a $0,92$, indicando que esses modelos não estão identificando corretamente todas as amostras dessa classe. O *F1-score* para essa classe varia de $0,70$ a $0,94$.

Em geral, os modelos Árvore de decisão e *Random Forest* obteve um desempenho consistente e sólido em todas as classes, com altos valores de precisão, *recall* e *F1-score*. O modelo Regressão Logística também apresenta um bom desempenho em termos de acurácia geral, mas seu desempenho varia por classe, com problemas de *recall* em algumas delas. O modelo KNN tem uma acurácia mais baixa e um desempenho menos consistente nas diferentes classes. O modelo RNA apresenta uma acurácia ainda mais baixa e desempenho insatisfatório em algumas classes.

Após a aplicação da técnica SMOTE para balanceamento dos dados, as métricas de avaliação dos modelos foram atualizadas. A seguir, é apresentada a análise delas.

No Regressão Logística , a acurácia aumentou para 0,950, indicando um bom desempenho geral do modelo. As métricas por classe também melhoraram. A classe *Enterprise* teve um aumento na precisão e no *F1-score*, indicando que o modelo está melhor em identificar corretamente essa classe. No entanto, o *recall* diminuiu, o que significa que o modelo ainda está perdendo alguns casos dessa classe. As métricas para as classes *Medium Business* e *Small Business* também tiveram melhorias, com aumento na precisão, *recall* e *F1-score*.

Tabela 4 – Métricas por Classe para o modelo Logistic Regression com SMOTE

Classe	**Precisão**	**Recall**	**F1-score**
Enterprise	0,703	0,950	0,808
Medium Business	0,983	0,947	0,964
Small Business	0,878	0,923	0,900

Analisando o modelo Árvore de decisão, vê-se que a acurácia se manteve alta (0,980), indicando um bom desempenho geral do modelo. As métricas por classe também se mantiveram consistentes. A classe *Enterprise* teve um aumento na precisão e no *F1-score*, indicando um melhor desempenho na sua identificação. As métricas para as classes *Medium Business* e *Small Business* também se mantiveram altas, com bons valores de precisão, *recall* e *F1-score*.

Tabela 5 – Métricas por Classe para o modelo Árvore de decisão com SMOTE

Classe	**Precisão**	***Recall***	***F1-score***
Enterprise	0,904	0,950	0,926
Medium Business	0,983	0,987	0,985
Small Business	0,972	0,923	0,947

A acurácia se manteve alta (0,980) no modelo *Random Forest*. As métricas por classe também se mantiveram consistentes. A classe *Enterprise* teve um bom desempenho, com precisão, *recall* e F1-score altos. As métricas para as classes *Medium Business* e *Small Business* também se mantiveram altas, com bons valores de precisão, *recall* e F1-score.

Tabela 6 – Métricas por Classe para o modelo *Random Forest* com SMOTE

Classe	**Precisão**	**Recall**	**F1-score**
Enterprise	0,950	0,950	0,950
Medium Business	0,983	0,991	0,987
Small Business	0,972	0,923	0,947

No KNN a acurácia diminuiu para 0, 876, indicando um desempenho geral um pouco mais baixo do modelo em comparação com os demais. As métricas por classe mostram que o modelo ainda tem dificuldades em identificar corretamente a classe *Enterprise*, com valores mais baixos de precisão, *recall* e *F1-score*. Para as classes *Medium Business* e *Small Business*, o modelo apresenta resultados melhores, com valores mais altos nas métricas de precisão, *recall* e *F1-score*.

Tabela 7 – Métricas por Classe para o modelo KNN com SMOTE

Classe	**Precisão**	**Recall**	**F1-score**
Enterprise	0,545	0,900	0,679
Medium Business	0,968	0,878	0,921
Small Business	0,693	0,871	0,772

Por fim, quanto ao RNA, sua acurácia diminuiu significativamente para 0,750, indicando um desempenho geral muito baixo desse modelo. As métricas por classe mostram um desempenho demasiadamente fraco para todas as classes. A classe *Enterprise* tem uma precisão muito baixa, enquanto que as classes *Medium Business* e *Small Business* têm valores baixos em todas as métricas.

Em resumo, após a aplicação da técnica SMOTE para o balanceamento dos dados, houve melhorias gerais nas métricas de avaliação na maioria dos modelos, com exceção do modelo RNA.

Tabela 8 – Métricas por Classe para o modelo RNA com SMOTE

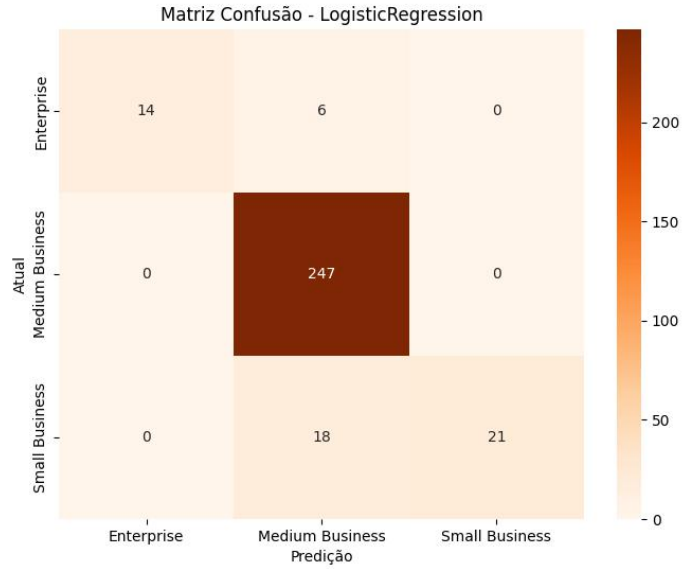
Classe	**Precisão**	**Recall**	**F1-score**
Enterprise	0,253	0,800	0,385
Medium Business	0,873	0,445	0,589
Small Business	0,205	0,615	0,307

Ao observar as matrizes de confusão de cada modelo nas figuras 28, 29 e 30, é possível ver os erros e os acertos de cada modelo, e com isso confirmar o melhor desempenho dos modelos Árvore de Decisão e Random Forest.

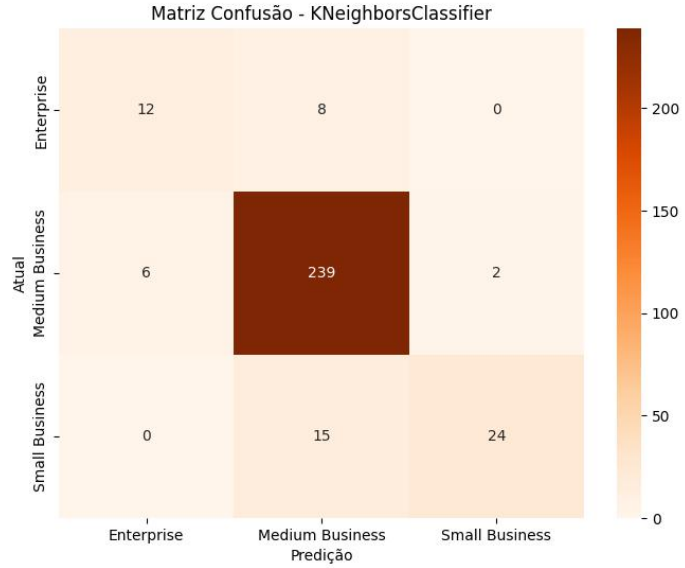
Com base nos resultados experimentais dos modelos estudados, foi possível observar que o modelo *Random Forest* apresentou um desempenho superior em relação a outros modelos analisados. Em geral, ele teve bons resultados para as métricas de forma global e do ponto de vista local, observado o seu desempenho em cada classe. Por esses motivos, foi o modelo selecionado para produção e para a execução da prova de conceito do projeto.

6.3 RESULTADOS DA PROVA DE CONCEITO

Para a idealização da prova de conceito, foi utilizado o modelo *Random Forest*. Com o intuito de obter um bom resultado, utilizou-se o método RandomizedCV para a

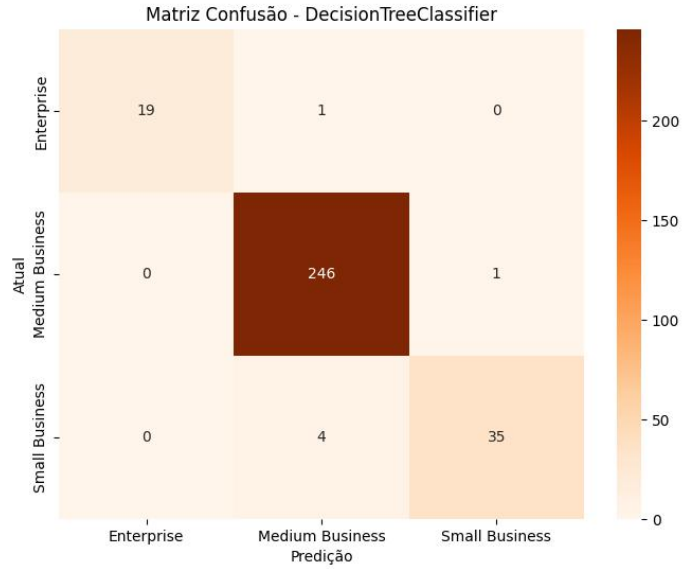


(a) Matriz de confusão da Regressão Logística

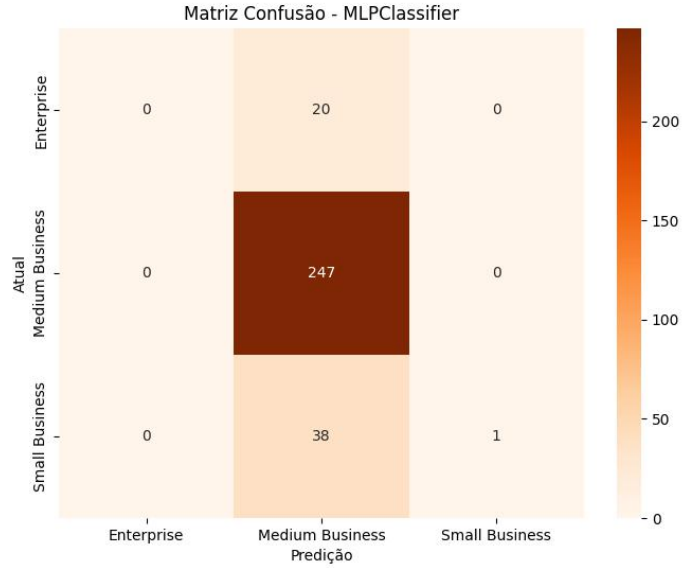


(b) Matriz de confusão do KNN

Figura 28 – Matriz de confusão dos modelos Regressão Logística e KNN



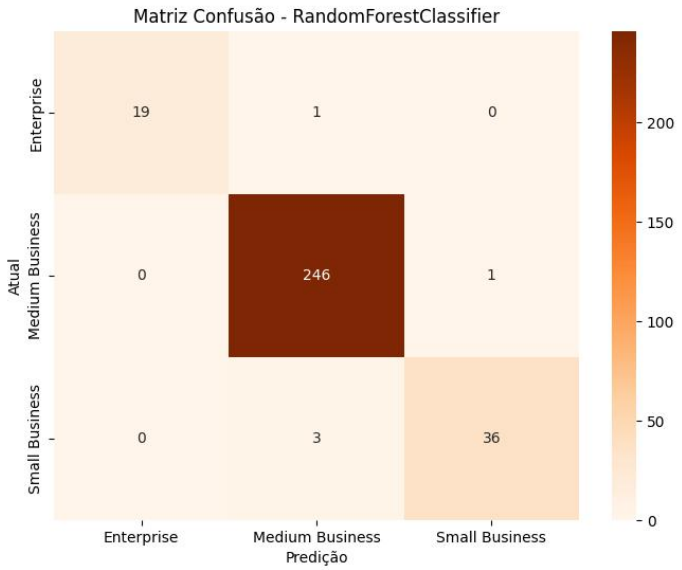
(a) Matriz de confusão de *Árvore de Decisão*



(b) Matriz de confusão da RNA

Figura 29 – Matriz de confusão dos modelos *Árvore de Decisão* e RNA

Figura 30 – Matriz de confusão da *Random Forest*.



Fonte: Autor.

definição de hiperparâmetros e o SMOTE para o *oversampling*. O resultado atingido estava dentro das expectativas. Na figura 30 podemos ver a matriz de confusão das predições realizadas, em que a taxa de acerto foi extremamente positiva. Para a classe *Enterprise*, o modelo foi capaz de obter 100% de acerto, enquanto que para a classe *Medium Business* ele acertou a sua grande maioria, embora tenha trazido alguns valores errôneos (2 para *Enterprise* e 3 para *Small*). Por fim, a classe *Small Business* apresentou apenas um erro de predição.

A integração com o CRM foi bem sucedida, pois foi capaz de integrar com a API do Pipedrive e atualizar o campo *Cluster* da entidade Organização com base nas predições do modelo *Random Forest*. Em termos de funcionalidade, o algoritmo assume que as predições são válidas e pode realizar o mapeamento para os IDs do campo *Cluster*. É importante garantir que o modelo esteja gerando as predições certas e que a lógica de mapeamento esteja correta para assegurar que as atualizações no CRM sejam precisas. Com a geração do arquivo CSV, as métricas de avaliação do modelo e o *dashboard* para monitoramento das predições, é possível ter segurança no envio das informações.

Apesar dos resultados positivos, alguns aspectos de atenção devem ser levados em conta. Primeiramente, para se conseguir uma execução correta, a qualidade dos dados de entrada é de extrema importância. A solução não é capaz de realizar predições para diferentes configurações de dados ou com dados de entrada faltantes, evidenciando a importância do processo de pulverização de mais dados na base de

potenciais clientes da empresa.

Vale ressaltar também que a prova de conceito construída não aborda aspectos de escalabilidade. Para implementá-la em produção, é necessário considerar a capacidade da solução em lidar com grande volume de predições e atualizações no CRM de forma eficaz. Por fim, a solução ainda precisa evoluir em aspectos de segurança e proteção de dados. O algoritmo necessita de um trabalho de adequação para proteger informações confidenciais, como as credenciais e os dados que são enviados ao CRM.

Com a utilização da prova de conceito, o processo de classificação de potenciais clientes passa a ter uma significativa melhoria. Originalmente, a realização da classificação do porte do cliente estava totalmente dependente dos ciclos de pré-venda e vendas combinados, podendo, por isso, levar dias para ser concluída, e, além disso, apresentar uma margem considerável de erros humanos e falhas no preenchimento das informações.

No entanto, com a introdução desta nova abordagem, o tempo necessário para se obter as classificações é reduzido significativamente. Agora, o processo é realizado de forma mais eficiente e rápida, permitindo que a equipe possa obter as classificações em apenas um dia.

Além disso, a confiabilidade do processo foi aprimorada consideravelmente. Os erros humanos e as falhas de preenchimento de informações, que eram frequentes anteriormente, foram minimizados. A integração do modelo *Random Forest* ao CRM proporcionou uma maior precisão na classificação dos potenciais clientes, além de ter eliminado boa parte dos erros cometidos na seriação de clientes com valores já preditos.

Esses resultados representam um avanço significativo para a empresa, pois reduzem o tempo necessário para se obter informações importantes sobre os potenciais clientes. Além disso, a confiabilidade aprimorada garante que as decisões de venda e as estratégias de acompanhamento possam ser baseadas em dados mais precisos e confiáveis. É importante ressaltar a relevância de se investir continuamente na melhoria da produção do modelo para projetos futuros. Essa etapa é crucial para garantir que o modelo seja implementado de forma eficiente e escalável, possibilitando seu uso em um ambiente de produção real. Existem diversas técnicas e ferramentas mais sofisticadas que podem ser exploradas posteriormente, visando aprimorar ainda mais o processo de produção.

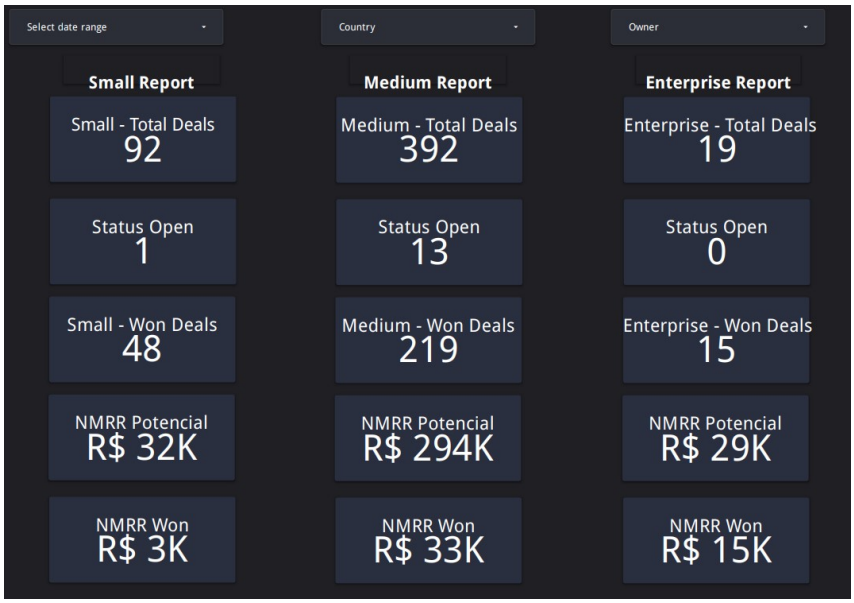
6.4 RESULTADOS DO *DASHBOARD*

O *dashboard* desenvolvido apresenta uma estrutura inicial que demonstra o poder de análise proveniente da classificação de potenciais clientes. As visualizações criadas fornecem *insights* valiosos seja no uso operacional quanto no estratégico. No entanto, para complementar e aprimorar ainda mais a análise, é necessário, em

atividades futuras, pulverizar mais informações no CRM.

A figura 31 exibe o número seriado de *leads* que se encontram na base, bem como o status deles — ou seja, se estão em aberto ou já são clientes —, o potencial de *New Monthly Recurring Revenue* (NMRR) (New Monthly Recurring Revenue) conhecido em português como Nova Receita Recorrente Mensal de que se tem conhecimento e o quanto já foi vendido. Vale ressaltar que os dados foram alterados para preservação de informações sigilosas e que a figura, portanto, é apenas uma representação do processo real.

Figura 31 – Indicadores chaves do *dashboard*.

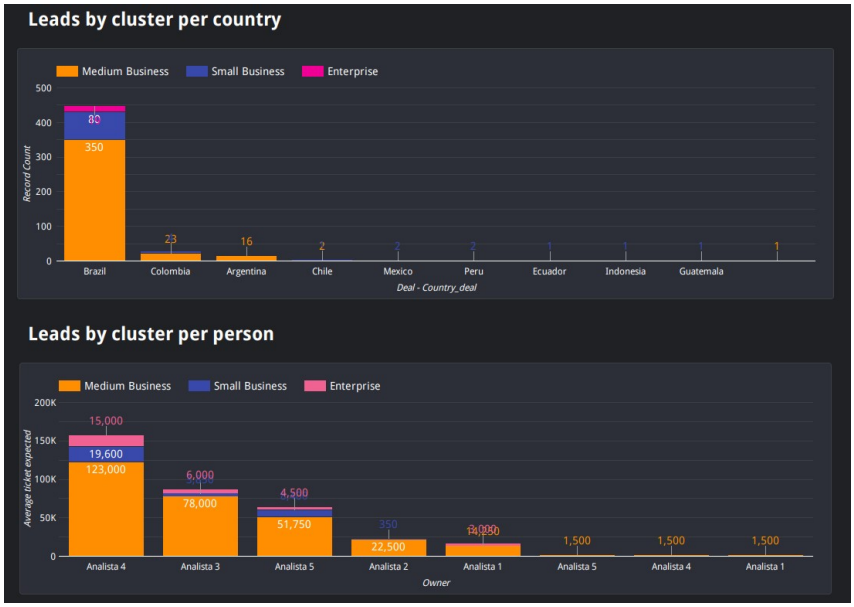


Fonte: Autor.

Também foi criado um recurso de visualização para mensurar os *leads*, que os seria por país e por analista comercial. Uma representação desse recurso em uso pode ser observada na figura 32, em que foram utilizados gráficos de barras empilhadas. Para o aprimoramento do *dashboard* em trabalhos futuros, uma adição útil será a inclusão de gráficos de funil de vendas, que mostram a quantidade de *leads* em cada etapa do processo de vendas, desde o primeiro contato com eles até a sua conversão final. Isso auxilia na identificação de gargalos, na análise da eficácia das ações em cada etapa do funil e no consequente direcionamento estratégico de esforços com vistas ao aumento da taxa de conversão.

Adicionado esse novo recurso, o *dashboard* será enriquecido com informações complementares, o que proporcionará uma análise mais completa e robusta dos potenciais clientes. Isso permitirá uma tomada de decisão mais embasada e aprimorará tanto a gestão operacional quanto a estratégica da equipe responsável pela captação e conversão de *leads*.

Figura 32 – Gráficos de barra do *dashboard*



Fonte: Autor.

7 CONCLUSÃO

Neste projeto foi explorado o uso de abordagens de aprendizado de máquina para o desafio de classificação multiclasse de potenciais cliente de uma empresa SaaS. Para isso, foram avaliados cinco modelos e foi criada uma solução como prova de conceito para aplicar no processo comercial da empresa.

Na etapa da análise exploratória de dados, foram identificados problemas não mapeados inicialmente, como desbalanceamento de classes, presença de *outliers* e valores nulos. Após tratamento dos dados, que se valeu de um conjunto de dados composto por 1019 registros para treino e teste na etapa de desenvolvimento de modelos, foi possível encontrar resultados promissores no fluxo de processo da empresa. Os objetivos gerais e específicos traçados no início do projeto foram cumpridos.

O uso de ferramentas como a biblioteca Scikt-learn foi de grande valia para acelerar o desenvolvimento dos modelos, assim como o ambiente Google Colaboratory contribuiu para uma maior fluidez do projeto.

Na avaliação dos modelos, a abordagem *ensemble Random Forest* teve melhor desempenho que outros algoritmos, por ter em sua estrutura natural uma maior robustez para a resolução de problemas de classificação multiclasse, mesmo com classes desbalanceadas. Porém, o estudo e a avaliação de técnicas usadas nos outros algoritmos demonstrou que, com uma correta combinação de abordagens, como *oversampling* para balanceamento dos dados, é possível obter resultados promissores na solução desse tipo de problema.

Além disso, a confiabilidade do processo foi aprimorada consideravelmente. Os erros humanos e as falhas de preenchimento de informações, que eram frequentes anteriormente, puderam ser minimizados com o uso da solução. A integração do modelo *Random Forest* ao CRM proporcionou maior precisão na classificação dos potenciais clientes, eliminando boa parte dos erros que eram cometidos no processo anterior.

Esses resultados representam um avanço significativo para a empresa, pois reduzem o tempo necessário para se obter informações importantes sobre os potenciais clientes. Ademais, a confiabilidade aprimorada garante que as decisões de venda e as estratégias de acompanhamento possam ser baseadas em dados mais precisos e confiáveis.

Para o uso dessas técnicas no fluxo de processo, decidiu-se utilizar a estrutura de integração da API do CRM para o envio de predições obtidas através do modelo escolhido.

No geral, a implementação bem sucedida dessa nova prova de conceito resultou em um processo mais ágil, confiável e eficiente para a classificação de potenciais clientes. Isso muito provavelmente se converterá em um impacto positivo nos esforços de vendas da empresa, permitindo uma tomada de decisão mais informada e uma

abordagem mais direcionada para se alcançar o sucesso comercial.

Apesar dos requisitos traçados serem cumpridos no decorrer do desenvolvimento do projeto, é notória a necessidade de maior enriquecimento na base de dados para se colher resultados cada vez mais precisos e abrangentes. Esta é uma atividade que será abordada constantemente para melhorar os resultados obtidos.

Em trabalhos futuros será explorada uma estratégia *ensemble* de modelos, em que, ao se combinar diferentes modelos, a predição final possa ser feita em uma votação da predição dos modelos, com o objetivo de minimizar os erros locais de forma geral.

Será importante a evolução dessa estrutura, cogitando ferramentas mais sofisticadas para o *deployment* de modelos, com isso garantindo maior eficiência na usabilidade dos resultados obtidos neste projeto. Além disso, melhorias na aquisição e organização dos dados no CRM serão exploradas para fornecimento de uma visão ampla do potencial de mercado da empresa.

Por fim, o projeto mostrou-se bastante desafiador, uma vez que nele se impôs a necessidade de se entender e utilizar diferentes ferramentas e técnicas para se obter o resultado final. No entanto, o uso de modelos de aprendizado de máquina no processo comercial provou-se uma estratégia eficaz, a ser empregada tanto nas análises quanto nas tomadas de decisão, pois traz maior confiabilidade à estratégia da empresa.

REFERÊNCIAS

BASHEER, I.A; HAJMEER, M. Artificial neural networks: Fundamentals, computing, design, and application. **Journal of Microbiological Methods**, v. 43, n. 1, p. 3–31, 2000. Neural Computting in Micrbiology. ISSN 0167-7012. DOI: https://doi.org/10.1016/S0167-7012(00)00201-3.

BASHEER, Imad; HAJMEER, M.N. Artificial Neural Networks: Fundamentals, Computing, Design, and Application. **Journal of Microbiological Methods**, v. 43, p. 3–31, jan. 2001. DOI: 10.1016/S0167-7012(00)00201-3.

BENTO, Carolina. **Decision Tree Classifier Explained in Real Life: Picking a Vacation Destination**. [*S.l.*: *s.n.*], 2021. https://medium.com/towards-data-science/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575. Acesso em: 10 de junho de 2023.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. [*S.l.*]: Springer, 2006.

BRONSHTEIN, Adi. **Train-Test Split and Cross-Validation in Python**. Medium. 2017. Disponível em: https://medium.com/towards-data-science/train-test-split-and-cross-validation-in-python-80b61beca4b6.

CHAWLA, Nitesh V; BOWYER, Kevin W; HALL, Lawrence O; KEGELMEYER, W Philip. SMOTE: Synthetic Minority Over-sampling Technique. **Journal of Artificial Intelligence Research**, v. 16, n. 1, p. 321–357, 2002.

DAS, Ashesh. **Oversampling**. [*S.l.*: *s.n.*]. Disponível em: https://medium.com/@asheshdas.ds/oversampling-to-remove-class-imbalance-using-smote-94d5648e7d35. Acessado em: 27 de Junho de 2023.

DEMŠAR, Janez. Statistical Comparisons of Classifiers over Multiple Data Sets. **Journal of Machine Learning Research**, JMLR, v. 7, p. 1–30, dez. 2006. ISSN 1532-4435.

HUBER, Steffen; WIEMER, Hajo; SCHNEIDER, Dorothea; IHLENFELDT, Steffen. DMME: Data mining methodology for engineering applications–a holistic extension to the CRISP-DM model. **Procedia Cirp**, Elsevier, v. 79, p. 403–408, 2019.

IBM. **IBM KNN**. [*S.l.*: *s.n.*]. https://www.ibm.com/topics/knn. Acesso em: 21 de junho de 2023.

LAAKSONEN, Jorma; OJA, Erkki. Classification with learning k-nearest neighbors. *In*: IEEE. PROCEEDINGS of International Conference on Neural Networks (ICNN'96). [*S.l.*: *s.n.*], 1996. P. 1480–1483.

LOGISTICREGRESSION - scikit-learn documentation. [*S.l.*: *s.n.*]. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Acesso em: 10 de junho de 2023.

MATHWORKS. **Overfitting**. MathWorks. 2023. Disponível em: https://www.mathworks.com/discovery/overfitting.html.

MITCHELL, Tom M. **Machine Learning**. New York, NY: McGraw-Hill, 1997. ISBN 9780070428072.

NISHIMOTO, Paulo Hiroki. **Analysis of machine learning methods and AutoML tools for indirect measurement of evaporation and condensation temperatures in variable speed compressors**. [*S.l.*: *s.n.*], 2018. https://repositorio.ufsc.br/handle/123456789/222693?show=full. Acessado em: 2 de julho de 2023.

PAUL, Angshuman; MUKHERJEE, Dipti Prasad; DAS, Prasun; GANGOPADHYAY, Abhinandan; CHINTHA, Appa Rao; KUNDU, Saurabh. Improved random forest for classification. **IEEE Transactions on Image Processing**, IEEE, v. 27, n. 8, p. 4012–4024, 2018.

PEDREGOSA, F. *et al.* **RandomizedSearchCV**. [*S.l.*: *s.n.*], 2023. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html. Online; Acessado em 11 de junho de 2023.

ROSS, Aaron; TYLER, Marylou. **Receita Previsıvel: Como implantar a metodologia revolucionária de vendas outbound que pode triplicar os resultados da sua empresa.** [*S.l.*]: Autêntica, 2017.

SAHOO, Kabita; SAMAL, Abhaya Kumar; PRAMANIK, Jitendra; PANI, Subhendu Kumar. Exploratory data analysis using Python. **International**

Journal of Innovative Technology and Exploring Engineering (IJITEE), v. 8, n. 12, p. 2019, 2019.

SCIKIT-LEARN. **Cross-validation**. Scikit-learn. Acesso em 2023. Disponível em: `https://scikit-learn.org/stable/modules/cross_validation.html`.

SERGUE, Marie. **Customer Churn Analysis and Prediction Using Machine Learning for a B2B SaaS Company**. [*S.l.*: *s.n.*], 2020.

SKRYJOMSKI, Przemysław; KRAWCZYK, Bartosz. Influence of minority class instance types on SMOTE imbalanced data oversampling. *In*: PMLR. FIRST International Workshop on Learning with Imbalanced Domains: Theory and Applications. [*S.l.*: *s.n.*], 2017. P. 7–21.

STANFORD UNIVERSITY. **Logistic Regression**. 2017. Disponível em: `https://web.stanford.edu/class/archive/cs/cs109/cs109.1178/lectureHandouts/220-logistic-regression.pdf`.

WIRTH, Rüdiger; HIPP, Jochen. CRISP-DM: Towards a standard process model for data mining. *In*: MANCHESTER. PROCEEDINGS of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining. [*S.l.*: *s.n.*], 2000. P. 29–39.

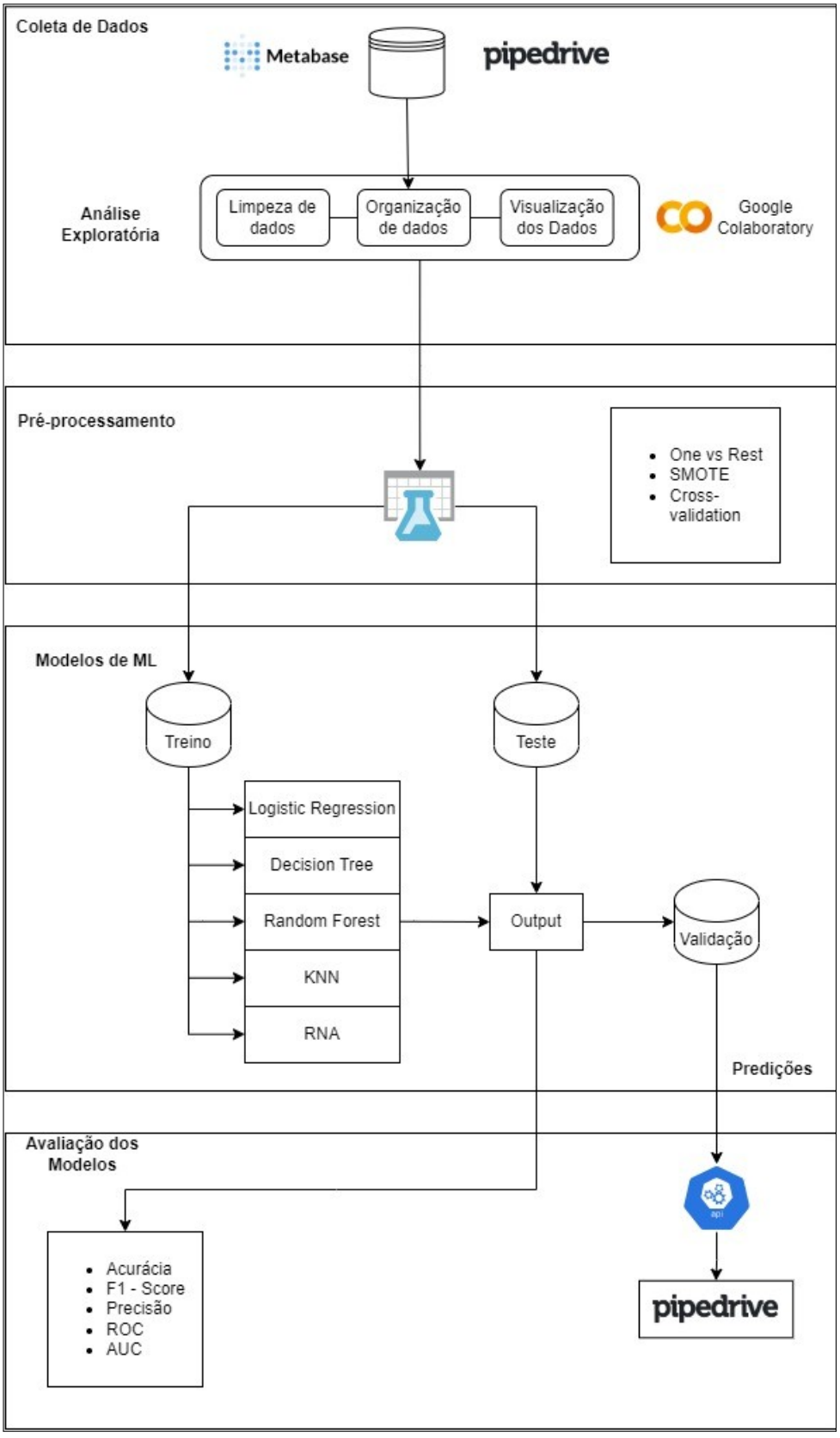
XU, Jianhua. An extended one-versus-rest support vector machine for multi-label classification. **Neurocomputing**, Elsevier, v. 74, n. 17, p. 3114–3124, 2011.

YIU, Tony. **Understanding Random Forest**. Acessado em: June 3, 2023. 2019. Disponível em: `https://towardsdatascience.com/understanding-random-forest-58381e0602d2`.

Apêndices

APÊNDICE A – ESTRUTURA GERAL DO PROJETO

Figura 33 – Estrutura geral do projeto.



Fonte: Autor.

APÊNDICE B – SCRIPT PARA VALIDAÇÃO DOS MODELOS

```
#Imports dados
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

#Imports pré-processamento
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE


#Imports modelos
# Import dos modelos a serem avaliados
from sklearn.linear_model import LogisticRegression  # Logistic regression
from sklearn.tree import DecisionTreeClassifier  #  Decision Tree Classifier
from sklearn.ensemble import RandomForestClassifier  # Random Forest
from sklearn.neighbors import KNeighborsClassifier #KNN
from sklearn.neural_network import MLPClassifier #RNA

#salvar modelo
import joblib


#Imports métricas
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
```

```
# Carrega o dataset
df = pd.read_csv('dataset-analise4.csv', header=0)
```

```
def dados(df):
    # Exclusão de colunas
    df.drop(columns=['deal_id', 'name', 'country'], inplace=True)

    # Transformar coluna categórica em variáveis binárias
    df = pd.get_dummies(df, prefix=['CP'], columns=['type'])
```

```
    # Separar atributos e target
    X = df.drop(['size', 'org_id'], axis=1)
    y = df['size']

    # Dividir os dados em conjuntos de treinamento e teste
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    return X_train, X_test, y_train, y_test

# Pré-processar os dados
X_train, X_test, y_train, y_test = dados(df)
```

```
#aplica técnica SMOTE
def apply_smote(X, y):
    smote = SMOTE()
    X_resampled, y_resampled = smote.fit_resample(X, y)
    return X_resampled, y_resampled
```

```
#função para treinar cada modelo com parametrização definida
def train_model(X, y, model_name):
    if model_name == 'LogisticRegression':
        modelo = LogisticRegression(multi_class='ovr', max_iter=1000)
    elif model_name == 'DecisionTreeClassifier':
        modelo = DecisionTreeClassifier(min_samples_leaf=4, min_samples_split=2,
max_depth=20, criterion='entropy')
    elif model_name == 'RandomForestClassifier':
        modelo = RandomForestClassifier(n_estimators= 200, min_samples_split= 10,
min_samples_leaf= 1, max_depth= 10, criterion= 'gini')
    elif model_name == 'KNeighborsClassifier':
        modelo = KNeighborsClassifier(n_neighbors=5)
    elif model_name == 'MLPClassifier':
        modelo = MLPClassifier(hidden_layer_sizes=(100,), max_iter=680, solver='adam',
activation='logistic')
    else:
        raise ValueError("Modelo inválido: " + model_name)
```

```
    modelo.fit(X, y)
    joblib.dump(modelo, 'modelo_treinado_' + model_name + '.pkl')
    return modelo

#função para realizar predições de cada modelo
def make_predictions(model, X):
    y_pred = model.predict(X)
    return y_pred

#função para validação das metricas
def validate_model(y_true, y_pred, classes):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')

    class_accuracy = accuracy_score(y_true, y_pred)
    class_precision = precision_score(y_true, y_pred, average=None)
    class_recall = recall_score(y_true, y_pred, average=None)
    class_f1 = f1_score(y_true, y_pred, average=None)

    metrics_dict = {
        'Accuracy': accuracy,

        'Precision': precision,
        'Recall': recall,
        'F1-Score': f1,
        'Class Accuracy': class_accuracy,
        'Class Precision': class_precision,
        'Class Recall': class_recall,
        'Class F1-Score': class_f1
    }
    return metrics_dict


def main():
    modelo_pipeline = [
        'LogisticRegression',
        'DecisionTreeClassifier',
        'RandomForestClassifier',
        'KNeighborsClassifier',
        'MLPClassifier'
    ]
```

```
    metrics_dict = {
        'Model': [],
        'Accuracy': [],
        'Precision': [],
        'Recall': [],
        'F1-Score': [],
        'Class Precision': [],
        'Class Recall': [],
        'Class F1-Score': []
    }

#loop principal, aplica oversampling, treina, gera predições e métrica
    for modelo_name in modelo_pipeline:
        # Aplicar SMOTE nos dados de treinamento
        X_train_resampled, y_train_resampled = apply_smote(X_train, y_train)

        modelo = train_model(X_train_resampled, y_train_resampled, modelo_name)
        y_pred = make_predictions(modelo, X_test)

        metrics = validate_model(y_test, y_pred, classes=modelo.classes_)

        metrics_dict['Model'].append(modelo_name)
        metrics_dict['Accuracy'].append(metrics['Accuracy'])
        metrics_dict['Precision'].append(metrics['Precision'])
        metrics_dict['Recall'].append(metrics['Recall'])
        metrics_dict['F1-Score'].append(metrics['F1-Score'])
        metrics_dict['Class Precision'].append(metrics['Class Precision'])
        metrics_dict['Class Recall'].append(metrics['Class Recall'])
        metrics_dict['Class F1-Score'].append(metrics['Class F1-Score'])

        # Gerar e salvar as imagens das matrizes de confusão
        cm = confusion_matrix(y_test, y_pred)
        plt.figure(figsize=(8, 6))
        ax = sns.heatmap(cm, annot=True, cmap='Oranges', fmt='d',
xticklabels=modelo.classes_, yticklabels=modelo.classes_)
        plt.xlabel('Predição')
        plt.ylabel('Atual')
        plt.title('Matriz Confusão - ' + modelo_name)
        plt.savefig('confusion_matrix_' + modelo_name + '.jpg')
        plt.close()

    # Imprimir as métricas para cada modelo
```

```
    for i in range(len(metrics_dict['Model'])):
        print("Métricas para o modelo", metrics_dict['Model'][i])
        print("Acurácia:", metrics_dict['Accuracy'][i])
        print("Precisão:", metrics_dict['Precision'][i])
        print("Recall:", metrics_dict['Recall'][i])
        print("F1-score:", metrics_dict['F1-Score'][i])
        print()

        class_labels = modelo.classes_
        for j, label in enumerate(class_labels):
            print("Métricas para a classe", label)
            print("Precisão da classe:", metrics_dict['Class Precision'][i][j])
            print("Recall da classe:", metrics_dict['Class Recall'][i][j])
            print("F1-score da classe:", metrics_dict['Class F1-Score'][i][j])
            print()

if __name__ == '__main__':
    main()
```