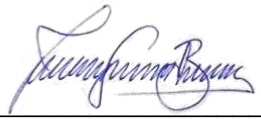
Universidade Federal do Rio De Janeiro

Escola Politécnica

MBA em Big Data, Business Intelligence e Business Analytics (MB3B-2)

UTILIZAÇÃO DE MACHINE LEARNING NO GERENCIAMENTO DE SERVIÇOS DE TI BASEADO NA ITIL 4

Autor:



Filipe Siqueira Burguez

Orientador:

Electronically signed by: Cláudio Latta
claudiolatta@gmail.com
23/07/2023 19:30:21
ID: FGQ2TEMRWGU-GEYTQMRZGU3Q-XCA7YGLM4GZNQ

Cláudio Luiz Latta de Souza, M. Sc.

Examinador:

Electronically signed by: Vinicius Drumond Gonzaga
viniciusdgonzaga@gmail.com
24/07/2023 15:35:29
ID: CGQ2TEMRWGU-GEYTQMRZGU3Q-IGQKIB2FRTQTV

Vinicius Drumond Gonzaga, M. Sc.

Examinador:

Electronically signed by: Vinicius Teixeira do Nascimento
vinicius.tn@yahoo.com
23/07/2023 20:45:41
ID: BGQ2TEMRWGU-GEYTQMRZGU3Q-4MKMTTHZDHV3N

Vinicius Teixeira do Nascimento, M. Sc.

Examinador:

Electronically signed by: Norberto Ribeiro Bellas
Norberto Ribeiro Bellas
norbertobellas@gmail.com
24/07/2023 23:24:33
ID: EGQ2TEMRWGU-GEYTQMRZGU3Q-DZIJHHM6ZU2RA

Norberto Ribeiro Bellas, M. Sc.

Rio de Janeiro
Junho/2023

Declaração de Autoria e de Direitos

Eu, **Filipe Siqueira Burguez,** CPF 118.882.947-52, autor da monografia ***UTILIZAÇÃO DE MACHINE LEARNING NO GERENCIAMENTO DE SERVIÇOS DE TI BASEADO NA ITIL 4***, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na defesa da monografia do curso de Pós-Graduação, Especialização MBA em Big Data, Business Intelligence e Business Analytics da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1 eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

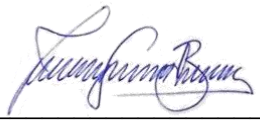
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

5. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

6. Por ser verdade, firmo a presente declaração.

Rio de Janeiro, 03 de junho de 2023.



Filipe Siqueira Burguez

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Av. Athos da Silveira, 149 - Centro de Tecnologia, Bloco H, sala - 212,
Cidade Universitária Rio de Janeiro – RJ - CEP 21949-900.

DEDICATÓRIA

À Deus, pelo seu infinito amor por nós.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por ter me sustentado todos os dias, me dando forças para continuar quando não achei.

Agradeço também aos meus pais, João Carlos e Lourdes, que moldaram toda a minha história e abriram mão de muitas coisas para que eu chegasse até aqui.

Ao meu irmão, Fernando, meu agradecimento não poderia deixar de ser feito, pois luta a cada dia para conquistar seus objetivos e sendo uma fonte de inspiração.

À minha amada esposa, Marcelle, que foi a fonte de motivação para o meu ingresso nesse curso. Sem ela, essa decisão não seria tomada.

Agradeço, aos demais familiares, em especial a minha tia Lucia, ao meu tio Roberto, a minha tia Alecir e a minha Avó Berenice, que sempre me colocam em suas orações à Deus com pedidos de proteção e direção.

Meu reconhecimento ao estimado amigo, Roberto Kane, que no começo desse curso dedicou uma parte do seu tempo retirando dúvidas da disciplina Banco de Dados.

RESUMO

O Gerenciamento de Serviços de Tecnologia da Informação (GSTI), através das melhores práticas da biblioteca ITIL 4 (*Information Technology Infrastructure Library*), pode otimizar e automatizar tarefas e procedimentos para a melhor gestão de ativos e serviços. O objetivo deste trabalho é apresentar a aplicação de *Machine Learning* para a predição do prazo de conclusão de ordens de serviços de equipamentos de controle de acesso em uma empresa de telecomunicações, utilizando algoritmos de classificação, sendo assim uma abordagem quantitativa para o estudo. Para alcançar o objetivo, utilizou-se a linguagem Python e algumas bibliotecas para tratamento de dados e criação das máquinas preditivas. Criou-se a variável alvo definindo as ordens de serviço fechadas dentro e fora de prazo. A técnica de Label Encoder foi utilizada para transformação de variáveis categóricas em numéricas. Foi realizada uma análise exploratória para conhecimento da base, verificando-se, através da variável alvo, um balanceamento da base de dados, onde 56,67% das ordens de serviço fecharam dentro do prazo de cinco dias e 43,33% foram encerradas fora do prazo. Constatou-se também que 31,92% dos valores da variável 'ResumoRelato' foram classificados como "em branco" e que cerca de 70% dos registros da variável 'TipoProblema' concentravam-se somente nas classificações "Sem conexão – Equipamento off-line" e "Outros", o que pode caracterizar uma influência negativa na eficácia dos algoritmos. Na sequência, foi realizada a análise preditiva do tempo de conclusão das ordens de serviço utilizando os algoritmos *Support Vector Machine (SVM)*, Regressão Logística, Árvore de Decisão, *K-Nearest Neighbors (KNN)* e *Random Forest*. Avaliou-se o resultado obtido em cada algoritmo com a utilização das métricas estatísticas de acurácia, precisão, *recall* e *F1-score*, utilizando os valores da matriz de confusão dos dados preditos e reais. O *Random Forest* apresentou um resultado de 65% de Acurácia e 64% em *F1-Score*, obtendo assim, o melhor desempenho entre os algoritmos utilizados. O estudo cumpriu o objetivo de atender ao princípio de otimização e automatização contido nos princípios orientadores da ITIL 4 e concluiu que o algoritmo com a melhor avaliação poderá ser colocado em operação, mas recomendou melhorias na base de dados e a utilização de outros algoritmos de classificação para que haja a possibilidade de aumento da assertividade dos dados preditos.

Palavras-chave: Manutenção corretiva; métricas estatísticas; Algoritmos de classificação; *Random Forest*; *Python*.

ABSTRACT

The Information Technology Service Management (ITSM), through the best practices of the ITIL 4 (Information Technology Infrastructure Library), can optimize and automate tasks and procedures for better management of assets and services. The objective of this work is to present the application of Machine Learning for the prediction of the deadline for completion of service orders of access control equipment in a telecommunications company, through the application of classification algorithms, thus being a quantitative approach to the study. To achieve the objective, the Python language and some libraries for data processing and creation of predictive machines were used. The target variable was created, defining service orders closed on and off deadlines. The Label Encoder technique was used to transform categorical variables into numerical ones. An exploratory analysis was performed to get to know the base, verifying, through the target variable, a balancing of the database, where 56.67% of the work orders were closed within the five-day deadline and 43.33% were closed after the deadline. It was also found that 31.92% of the values of the variable 'ResumoRelato' were classified as "blank" and that about 70% of the records of the variable 'TipoProblema' were concentrated only in the classifications "Sem conexão – Equipamento off-line" and "Outros", which may characterize a negative influence on the effectiveness of the algorithms. Next, the predictive analysis of work order completion time was performed using the Support Vector Machine (SVM), Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), and Random Forest algorithms. The result obtained from each algorithm was evaluated using the statistical metrics of accuracy, precision, recall, and F1-score using the confusion matrix values of the predicted and actual data. Random Forest presented a result of 65% accuracy and 64% in F1-Score, thus obtaining the best performance among the algorithms used. The study fulfilled the objective of meeting the principle of optimization and automation contained in the guiding principles of ITIL 4 and concluded that the algorithm with the best evaluation could be put into operation, but recommended improvements in the database and the use of other classification algorithms so that there is the possibility of increasing the assertiveness of the predicted data.

Keywords: Corrective maintenance; statistical metrics; Classification algorithms; Random Forest; Python.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CFTV	Circuito Fechado de Televisão
CIS	Controle de Informações de Segurança
FN	False Negative
FP	False Positive
GTSI	Gerenciamento de Serviços de Tecnologia da Informação
IA	Inteligência Artificial
ITIL	Information Technology Infrastructure Library
ITSM	Information Technology Service Management
KNN	K-Nearest Neighbors
OS	Ordens de Serviço
RH	Recursos Humanos
SLA	Service Level Agreement
S/A	Sociedade Anônima
SVM	Support Vector Machine
SVS	Sistema de Valor de Serviços
TI	Tecnologia da Informação
TN	True Negative
TP	True Positive

Sumário

Capítulo 1

Introdução

1.1 – Tema

Este trabalho abordará a utilização de técnicas de *Machine Learning* aplicadas no Gerenciamento de Serviços de Tecnologia da Informação (GSTI) ou, em inglês, *Information Technology Service Management* (*ITSM*), baseado nas boas práticas da biblioteca *Information Technology Infrastructure Library* (*ITIL* 4).

Os dados a serem utilizados são de uma base de Ordens de Serviço (OS) de equipamentos de controle de acesso predial, da empresa de Telecom OI S/A, que estejam em manutenção corretiva com objetivo de identificar padrões que possam auxiliar no processo de cobrança dos atendimentos da prestadora de serviço com objetivo de otimizar a resolução de problemas e diminuição do tempo de *Service Level Agreement* (SLA).

1.2 – Justificativa

Com grande importância no mercado de telecomunicações, a OI S/A possui diversos imóveis em todo o território nacional que possuem, internamente, uma ampla quantidade de equipamentos primordiais ao funcionamento da atividade-fim da Companhia.

O controle físico de acesso nas dependências da OI S/A está dividido em: áreas prediais, que correspondem ao acesso principal de prédios; e áreas restritas, que controlam o acesso a ambientes administrativos ou operacionais, sendo todos vinculados a um rígido *workflow* para aprovação de acessos. Esse processo envolve a aprovação pelo gestor imediato do colaborador para as áreas prediais. Já para as áreas restritas, além da necessidade de aprovação do gestor imediato, há uma segunda aprovação do gestor de área. Esse fluxo de aprovação é gerenciado através do sistema *SAFE* que está integrado com os sistemas de Recursos Humanos (RH), de maneira que somente funcionários ativos consigam solicitar e aprovar acessos, criando-se uma blindagem na prevenção de fraudes relacionadas a acessos indevidos.

Apesar da necessidade de garantia de funcionamento dos equipamentos de controle de acesso (portas, catracas e cancelas), eventualmente, há problemas diversos que impossibilitam o acesso em uma determinada área ou fragilizam a segurança nos ambientes controlados.

De forma terceirizada, contrata-se uma empresa especializada na manutenção desses equipamentos que, por vezes, deixa de atender aos chamados corretivos dentro do tempo de SLA.

A utilização de *Machine Learning* nos processos de GSTI, auxiliará na classificação das Ordens de Serviço (OS) de forma preditiva, de tal forma que haja otimização da força de trabalho de funcionários, dedicando tempo exclusivamente aos tratamentos das OS classificadas como fora do prazo estabelecido, mantendo-se assim, o foco na cobrança de atendimentos específicos, criando uma cultura de melhoria de processos e de alta disponibilidade de serviço dos equipamentos de controle de acesso.

1.3 – Objetivo geral

Esse trabalho tem por objetivo utilizar técnicas de *Machine Learning* em uma base de dados de Ordens de Serviço (OS) para prever quais atendimentos terão o encerramento das manutenções fora do prazo contratual, para subsidiar a melhoria da qualidade da prestação de serviço e evitar a aplicação de sanções administrativas às contratadas dos serviços especializados e, principalmente, a interrupção da segurança predial.

1.4 – Objetivos específicos

Serão apresentados os seguintes objetivos específicos:

- Realizar a apresentação dos princípios orientadores da ITIL 4 no GSTI, como embasamento ao processo de automatização;
- Tratar a base de dados para um bom desempenho dos algoritmos de classificação;
- Escolher o algoritmo capaz de prever, com maior percentual de assertividade, as manutenções corretivas dentro e fora do prazo para priorização nos atendimentos.

1.5 – Delimitação

Esse estudo concentra-se exclusivamente nas atividades da área de Segurança Empresarial da OI S/A, haja vista a utilização de uma base de dados de OS de manutenções corretivas em equipamentos de controle de acesso. Dessa forma, não se recomenda a utilização da mesma metodologia sem uma avaliação criteriosa das variáveis distintas das apresentadas.

Exclui-se desse trabalho, demais ordens de serviço que envolvam manutenção corretiva de outros equipamentos de segurança empresarial como dispositivos de gravação e câmeras de Circuito Fechado de Televisão (CFTV), pois apesar de pertencerem à mesma área de gestão, possuem outras características e custos diferenciados de atendimento.

1.6 – Metodologia

O presente Trabalho de Conclusão de Curso é uma pesquisa de natureza aplicada pois, conforme *Adelaide University* (2008, apud GIL, 2017, p. 17) o estudo caracteriza-se por uma busca de conhecimento em um contexto exclusivo.

É definido, pelo seu objetivo, como descritivo, pois, de acordo com GIL (2017) possui finalidade de vinculação entre variáveis para alcançar um determinado resultado.

Tecnicamente é considerado um trabalho de pesquisa bibliográfica, tendo em vista o embasamento teórico que ampara todo o estudo (GIL, 2017). Serão abordados na parte teórica necessária ao estudo, tópicos relacionados ao Gerenciamento de Serviços de Tecnologia da Informação (GSTI), à análise e tratamento de dados por meio da biblioteca *Pandas* e à aplicação de algoritmos de *Machine Learning*, através da utilização da biblioteca *Scikit Learn* do Python.

Trata-se também de estudo de caso, pois será utilizado o referencial teórico aplicado a uma base de dados, para gerar hipóteses (GIL, 2017).

O estudo considera o recebimento da base de dados da OI S/A, contendo registros de manutenções de equipamentos de controle de acesso, sendo necessário o processo de tratamento e limpeza para possibilitar uma melhor assertividade nos resultados a serem obtidos.

Por fim, com a utilização de técnicas de *Machine Learning*, caracteriza-se como uma pesquisa quantitativa, pois utiliza o estudo estatístico para classificar e analisar os dados (KAUARK; MANHÃES; MEDEIROS, 2010).

1.7 – Descrição

Este trabalho está estruturado em cinco capítulos.

O capítulo dois é composto pela parte teórica que fundamenta esse trabalho. Apresentou-se os princípios orientadores da biblioteca ITIL para melhoria nos processos de Gestão de Serviços de Tecnologia da Informação através da otimização e automatização, os conceitos de Inteligência Artificial e de Machine Learning, os métodos de classificação, as métricas de avaliação e os algoritmos utilizados nesse trabalho.

No capítulo três são descritas as tecnologias utilizadas para aplicação da teoria exposta no capítulo anterior. Demonstrou-se o uso da linguagem de programação Python e algumas bibliotecas para o pré-processamento e análise dos dados e para a criação das máquinas preditivas.

No capítulo quatro foram apresentados os resultados da análise exploratória e a avaliação dos algoritmos utilizados com base nas métricas estatísticas apresentadas no capítulo dois.

Por fim, no capítulo 5, foram consolidadas as conclusões obtidas através das tecnologias utilizadas e as oportunidades identificadas para uso no futuro.

Capítulo 2

Embasamento Teórico

Nessa seção serão abordadas as fundamentações teóricas que direcionam esse trabalho acadêmico. Inicialmente, será apresentado o uso da biblioteca ITIL 4 no Gerenciamento de Serviços de Tecnologia da Informação (GSTI), o conceito de Sistema de Valor de Serviços (SVS) e os sete princípios orientadores. Além disso, será feita uma abordagem da relação de um dos princípios apresentados com a utilização de algoritmos de *Machine Learning* para a predição de variáveis categóricas. Por fim, serão feitas as apresentações das funcionalidades de cada algoritmo a ser utilizado nesse estudo de caso.

2.1 – Gerenciamento de Serviços de TI com ITIL 4

Gerenciamento de Serviços de TI (GSTI) é, segundo Freitas (2013, p. 85), "um conjunto especializado de habilidades organizacionais para fornecer valor a clientes na forma de serviços" e, portanto, torna-se primordial a existência de regras que permitam que a entrega de valor seja realizada dentro de processos, princípios e boas práticas, aliados à governança corporativa.

ITIL é um conjunto das melhores práticas voltadas ao Gerenciamento de Serviços de TI, promovendo atualizações no setor de TI baseadas em modelos mais atuais (BON; VERHEIJEN, 2006).

A aplicação das recomendações contidas na biblioteca ITIL 4 no Gerenciamento de Serviços de TI (GSTI) é fundamental para que as empresas mantenham em operação os serviços e a infraestrutura tecnológica, pois sua utilização já é amplamente discutida e padronizada dentro do mercado de TI.

As versões anteriores eram focadas em processos. Já na versão 4, publicada em 2019, há uma mudança na estrutura, onde o foco passa a ser no Sistema de Valor de Serviço (SVS) (CHIARI, 2019). O SVS pode ser descrito como a soma de componentes e atividades de uma organização para a geração de valor ao cliente (AXELOS, 2019). Fazem parte da estrutura SVS (Figura 2.1) os seguintes componentes: princípios orientadores, governança, cadeia de valor de serviço, práticas e melhoria contínua.

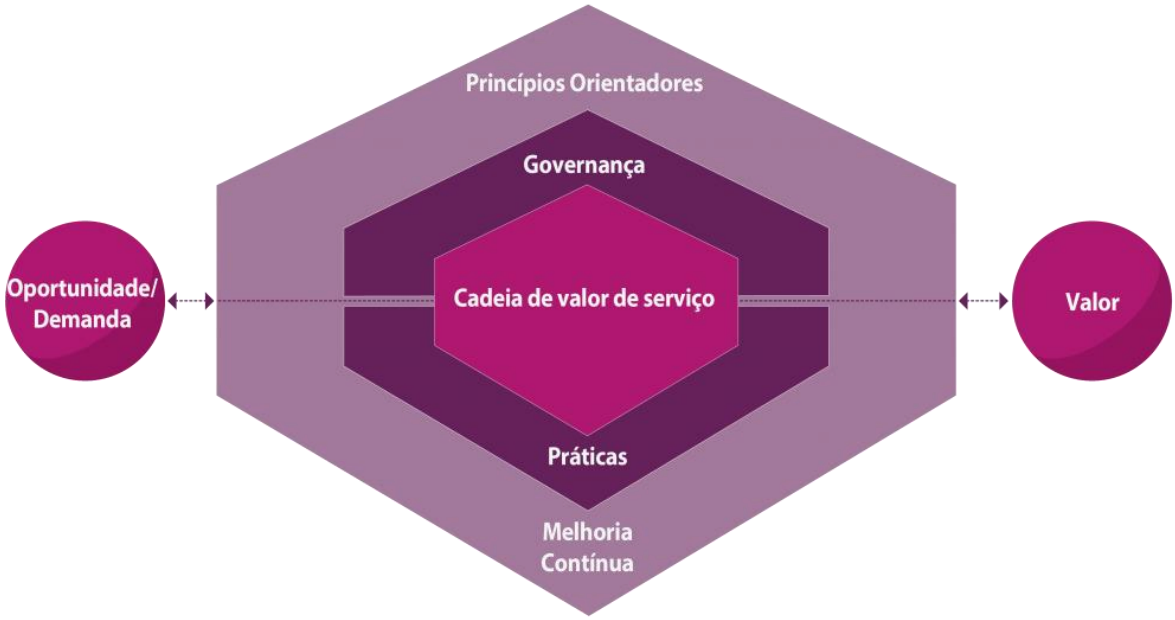


Figura 2.1 – Sistema de Valor de Serviço (SVS)
Fonte: Axelos (2019)

De acordo com AXELOS (2019), os sete Princípios Orientadores ITIL são:

- Foco no valor
- Começar de onde está
- Progredir iterativamente com *feedback*
- Colaborar e promover visibilidade
- Pensar e trabalhar holisticamente
- Manter de forma simples e prática
- Otimizar e automatizar

O último princípio apresentado, otimizar e automatizar, está fortemente ligado com as melhorias que serão propostas nesse estudo através da utilização de Inteligência Artificial, mais especificamente com a utilização de técnicas de *Machine Learning* para a predição de classificação.

2.2 – Inteligência Artificial e Machine Learning

A Inteligência Artificial (IA) é um campo da ciência da computação que pode ser caracterizado como um sistema que pensa e atua como pessoas (RUSSELL; NORVIG, 2013).

Tecnicamente, a IA pode ser caracterizada, segundo o Google Cloud (2023, online), como "um conjunto de tecnologias baseadas, principalmente, em *machine learning* e aprendizado profundo, usado para análise de dados, previsões, categorização de objetos, processamento de linguagem natural, recomendações, recuperação inteligente de dados".

O conceito de *Machine Learning* (Aprendizado de Máquina, em português) está dentro da IA (Figura 2.2) e utiliza técnicas para processar uma grande quantidade de dados, fornecendo aos computadores a capacidade de identificar padrões, possibilitando a realização de uma análise preditiva. Dessa forma, a aplicação de *Machine Learning* na automatização de processos em Gerenciamento de TI poderá gerar benefícios operacionais e orçamentários para uma empresa.

De forma prática, o processo de *Machine Learning* pode utilizar uma diversidade de algoritmos para aprender, de forma iterativa, com dados de treinamento e de teste fornecidos pelo homem, e produzir resultados satisfatórios considerando novas entradas de dados para predição (HURWITZ; KIRSCH, 2018).

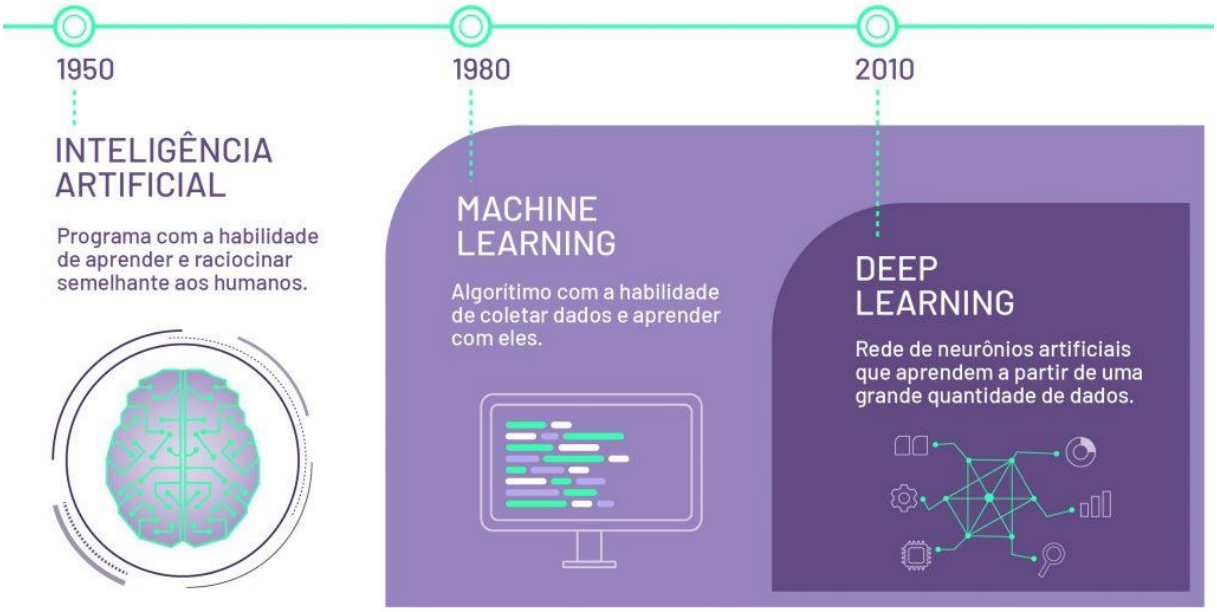


Figura 2.2 – Evolução da Inteligência Artificial
Fonte: Docket (2020)

2.3 – Métodos de Aprendizado de Máquina

O Aprendizado de Máquina pode ter sua aplicabilidade dividida em setores como: aprendizado supervisionado, não supervisionado, por reforço e aprendizado profundo (HURWITZ; KIRSCH, 2018).

O método de aprendizado supervisionado é aquele que utiliza dados rotulados para prever resultados de uma variável alvo contínua ou de um conjunto finito de valores.

Já o não supervisionado, não existe a variável alvo rotulada nos dados (GRUS, 2016), ou seja, a predição concentra-se em segmentar os dados em grupos (HURWITZ; KIRSCH, 2018).

No aprendizado por reforço, o algoritmo não trabalha com uma base de dados para obter resultados, como acontece no modelo supervisionado. A forma de aprendizado ocorre no método de tentativa e erro, onde o algoritmo melhora os resultados através de recompensas, ou seja, quando o algoritmo acerta, ele pontua e otimiza sua base de dados (HURWITZ; KIRSCH, 2018).

E, por fim, o aprendizado profundo (*deep learning*, em inglês), é um método que trabalha com dados de entrada, muitas vezes, não estruturados, fazendo uso de redes neurais distribuídas em camadas consecutivas (Figura 2.3) para que, de forma repetitiva, aprenda os padrões necessários para obter resultados de saída previstos (HURWITZ; KIRSCH, 2018).

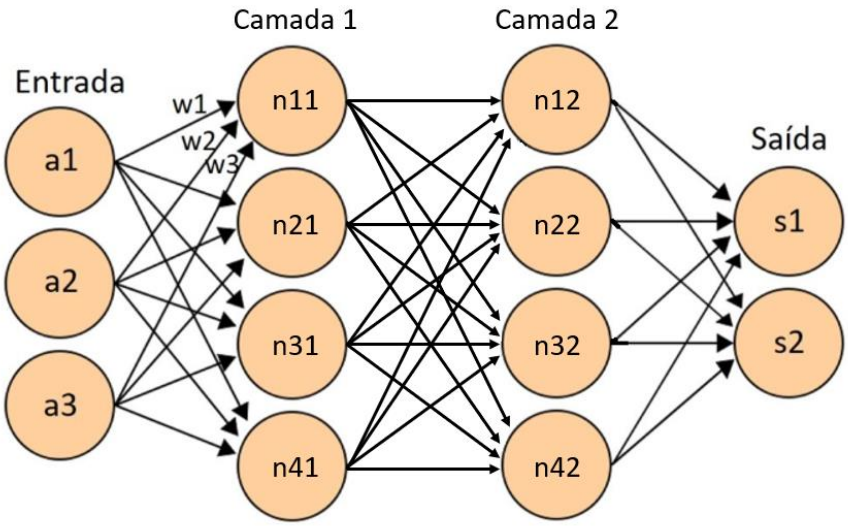


Figura 2.3 – Rede Neural
Fonte: Hardware (2022)

2.4 – Classificação

Classificação é um processo dentro modelo supervisionado que busca definir o rótulo de uma variável categórica com base em dados históricos das observações (LANTZ, 2013). Geralmente, o classificador é binário, como por exemplo: positivo / negativo, doente / não doente, spam / não spam (Figura 2.4).

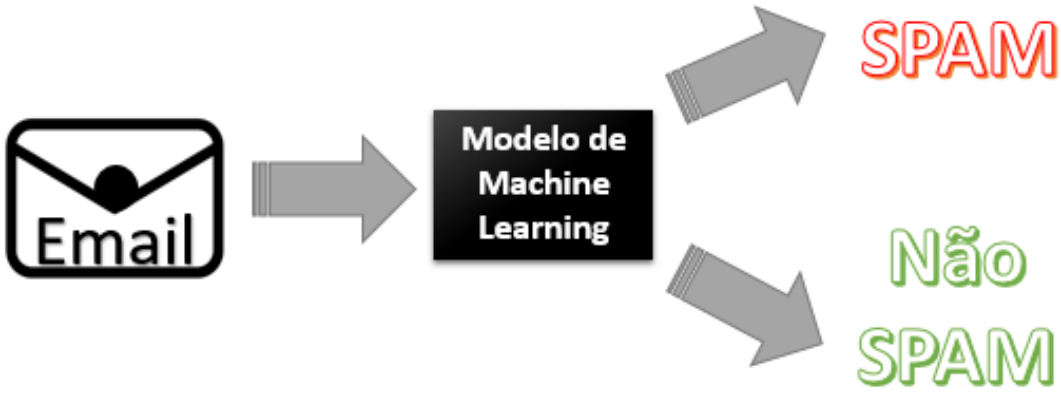


Figura 2.4 – Classificação de SPAM
Fonte: Autor (2023)

Para a utilização de algoritmos de classificação é necessário que a base de dados contenha registros anteriormente classificados na variável alvo de forma correta, de tal forma que o processo de aprendizado utilize esses dados para treinar os modelos e, posteriormente, realizar testes e avaliar a performance da predição.

2.5- Avaliação de modelos de classificação

2.5.1 – Matriz de Confusão

O processo de avaliar a eficácia de um modelo de classificação envolve o desenvolvimento de um comparativo entre as classes reais e as preditas, realizando-se a contagem absoluta de cada grupo e demonstrando a porcentagem em relação às classes reais (Figura 2.5).

Uma forma bastante eficaz de representação dos resultados da classificação é através da Matriz de Confusão. Nessa matriz, pode-se perceber, de forma simples, a relação entre as classes previstas e as classes reais.

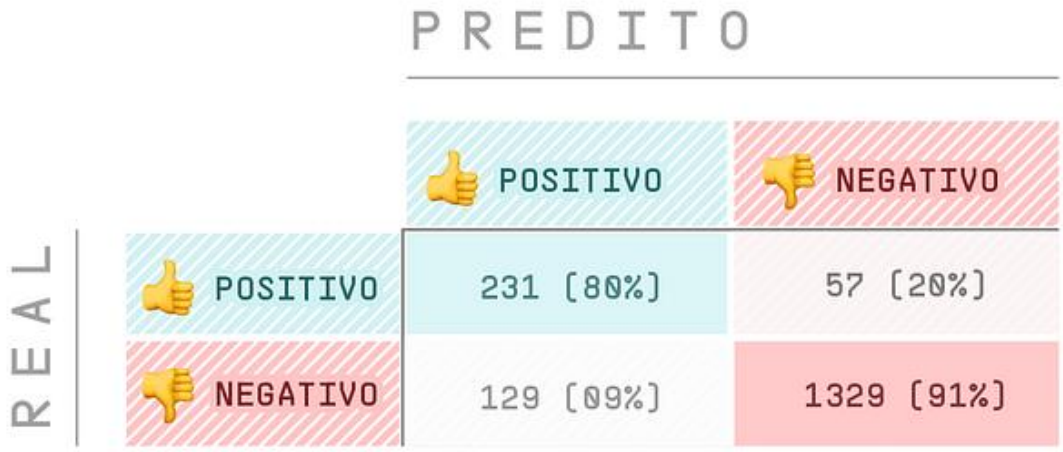


Figura 2.5 – matriz de confusão com valores
Fonte: Kunumi (2020)

Os resultados da avaliação são divididos em quatro grupos (Figura 2.6):

- Verdadeiro Positivo (*true positive* - TP) – classificação positiva e dados reais positivos;
- Verdadeiro Negativo (*true negative* - TN) – classificação negativa e dados reais negativos;
- Falso Positivo (*false positive* - FP) – classificação positiva e dados reais negativos;
- Falso Negativo (*false negative* - FN) – classificação negativa e dados reais positivos.

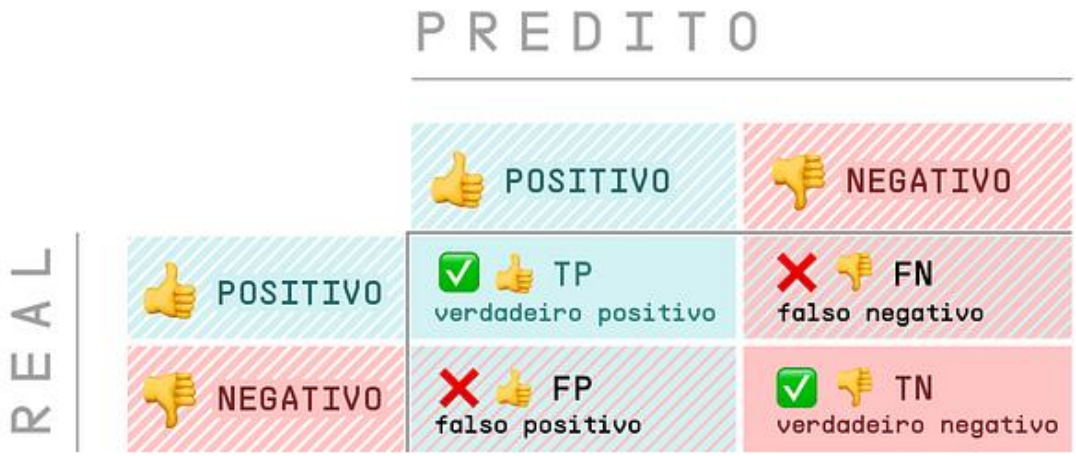


Figura 2.6 – matriz de confusão por grupos
Fonte: Kunumi (2020)

2.5.2 – Métricas de avaliação

A partir da definição dos resultados da matriz de confusão é possível identificar algumas métricas que auxiliam no processo de avaliação dos algoritmos de classificação. A seguir, serão apresentadas as quatro métricas utilizadas nesse estudo de caso, que foram aplicadas diretamente na variável alvo 'DentroPrazo' para a predição das Ordens de Serviço fechadas dentro e fora do prazo de 05 (cinco) dias. Para fins de exemplificação serão apresentadas as métricas de avaliação utilizando os resultados da matriz de confusão para o caso de classificação binária de e-mails com *Spam* (Figura 2.7), onde obteve-se a seguinte representação, considerando o caso hipotético de um conjunto de dados de teste:

		PREDITO	
		E-mails Spam	**E-mails não Spam**
REAL	**E-mails Spam**	**1200**	**50**
	E-mails não Spam	**30**	**800**

Figura 2.7 – matriz de confusão e-mails *Spam*
Fonte: Autor (2023)

2.5.2.1 – Acurácia

Corresponde ao percentual obtido em relação as saídas preditas corretamente e o valor total de previsões possíveis. Muitas vezes, a acurácia, de forma isolada, pode não ser eficaz pois pode apresentar um alto valor e possuir uma performance em torno do problema inaceitável. Como exemplo, para uma base de dados de ordens de serviço, com 1000 registros, sendo 990 com o rótulo de "concluídas dentro do prazo" e 10 com o rótulo de "concluídas fora do prazo", se o algoritmo classificar todos os exemplos como "concluídas dentro do prazo", mesmo assim seria obtido 99% de acurácia, o que na prática, não demonstraria êxito no processo de predição.

$$Acurácia = \frac{TP+TN}{TP+FP+TN+FN} \quad (1)$$

Considerando o caso hipotético de classificação de *Spam* obteve-se o resultado de 96% na métrica de acurácia, o que indica o quão bom o modelo classificou corretamente os dados de testes.

$$Acurácia_{Spam} = \frac{1200+800}{1200+30+800+50} \cong 96\% \text{ (2)}$$

2.5.2.2 – Precisão

Nessa métrica são utilizadas somente as classes previstas como positiva, onde é calculada a razão entre as previsões corretas e o total das previsões positivas. Faz-se a pergunta nos casos de precisão: das classificações positivas, quantas realmente são positivas?

$$Precisão = \frac{TP}{TP+FP} \text{ (3)}$$

No mesmo caso de classificação de *Spam* chegou-se ao resultado de 98% na métrica de precisão, o que demonstra ser uma alta assertividade dentre as previsões positivas. Abaixo, segue o cálculo precisão para o caso de e-mails *Spam*:

$$Precisão_{Spam} = \frac{1200}{1200+30} \cong 98\% \text{ (4)}$$

2.5.2.3 – Revocação

Diferente da precisão, a revocação ou *recall*, em inglês, possui o foco nos erros por falso negativo. O cálculo da revocação é feito através da razão entre as classificações corretas para a classe positiva sobre os valores que de fato são positivos. Para essa análise, cabe a seguinte pergunta: para os casos reais positivos, quantos foram previstos corretamente pelo algoritmo?

$$Revocação = \frac{TP}{TP+FN} \text{ (5)}$$

Ainda no mesmo exemplo de classificação de e-mails *Spam* obteve-se o resultado de 96% na métrica *recall*, o que demonstra uma alta assertividade nas classes reais de e-mails *Spam*. Para o resultado obtido, utilizou a fórmula acima, chegando-se no resultado mencionado conforme cálculo abaixo:

$$Revocação_{Spam} = \frac{1200}{1200+50} \cong 96\% \text{ (6)}$$

2.5.2.4 – F1-Score

A métrica F1-Score é a média harmônica entre precisão e revocação (PROVOST; FAWCETT, 2016). O F1-Score é utilizado para que não haja uma avaliação tendenciosa quando a precisão ou a revocação estiver baixa enquanto a outra alta, pois indicaria problemas na classificação de falsos positivos ou falsos negativos. A fórmula é apresentada da seguinte maneira:

$$F1_{Score} = 2 * \frac{Precisão * Revocação}{Precisão + Revocação} \text{ (7)}$$

Considerando o teste hipotético de previsão de e-mails *Spam*, e as avaliações de precisão e *recall* calculadas através da matriz de confusão, chegou-se ao valor de 97% na métrica *F1-Score*, o que representa um excelente equilíbrio entre os valores de precisão e *recall*. Utilizando a fórmula acima para o cálculo de *F1-Score* chegou-se ao resultado apresentado conforme exposto abaixo:

$$F1_{Score\_Spam} = 2 * \frac{98*96}{98+96} \cong 97\% \text{ (8)}$$

2.6 – Pré-processamento de dados

Antes de iniciar a criação dos modelos preditivos há a necessidade de se realizar o tratamento da base de dados logo após a coleta. Essa etapa é conhecida como Pré-processamento ou *Feature Engineering*, em inglês, sendo uma das partes mais dispendiosas do processo para obtenção de resultados preditivos com certa qualidade. Engloba as fases de preparação, organização e estruturação da base de dados.

Nesse processo pode-se encontrar dados nulos (*missing data*), valores errados, dados inconsistentes, redundância de registros, tipos de dados errados, valores discrepantes (*outliers*). As etapas de pré-processamento podem conter limpeza de dados, integração com outras bases de dados, redução e transformação (HAN; KAMBER; PEI, 2012).

Destacam-se algumas técnicas de pré-processamento:

- Análise dos valores nulos (DATAGEEKS, 2019): para um bom desempenho dos algoritmos é necessário que a base de dados não possua dados ausentes, tanto em linhas e quanto em colunas. As formas de tratamento dependem muito da avaliação do negócio e podem envolver remoção de registros com atributos nulos, aplicação de média ou mediana com os valores de um atributo, preenchimento com valores mais presentes na base de dados;
- Análise de valores únicos: para avaliar a possibilidade de redução de *features* com objetivo de otimizar o processamento do modelo de *Machine Learning* a ser executado;
- Análise dos tipos de variáveis: processo que visa auxiliar na identificação das possíveis variáveis a serem utilizadas no modelo e corrigir as eventuais divergências de tipos que atrapalhem o processamento;
- Eliminação de *features* desnecessárias para o modelo: colunas com valores únicos em todos os registros não impactarão no modelo e, a exclusão dessas, podem otimizar o desempenho dos algoritmos;
- Transformação de dados: etapa em que pode ser definida a criação de novos atributos a partir dos registros atuais da base de dados (GOMES, 2019). Nesse processo pode ser definida a própria variável alvo. Além disso, pode ser utilizada a técnica para transformar variáveis categóricas em numéricas, conhecida como *Label Encoder*.

2.7 – Sobreajuste e Subajuste

Durante a execução de um modelo de predição, pode haver um alto grau de assertividade com dados de treino, mas que, na prática, não ocorre a generalização ao ser processado com os dados de teste, ou seja, o modelo desempenha um bom resultado no treino, mas não traz resultados satisfatórios no teste. A esse problema dá-se o nome de sobreajuste ou *overfitting*, em inglês (Figura 2.8) (GRUS, 2016).

Quando o modelo não apresenta um bom resultado já na fase de treino, diz-se que o modelo está subajustado ou *underfitting* (Figura 2.8), em inglês. Dessa forma, não há uma relação entre os dados que seja satisfatória para desenvolver um alto grau de assertividade e, por isso, não se torna necessário prosseguir com a etapa utilizando os dados de teste (GRUS, 2016).

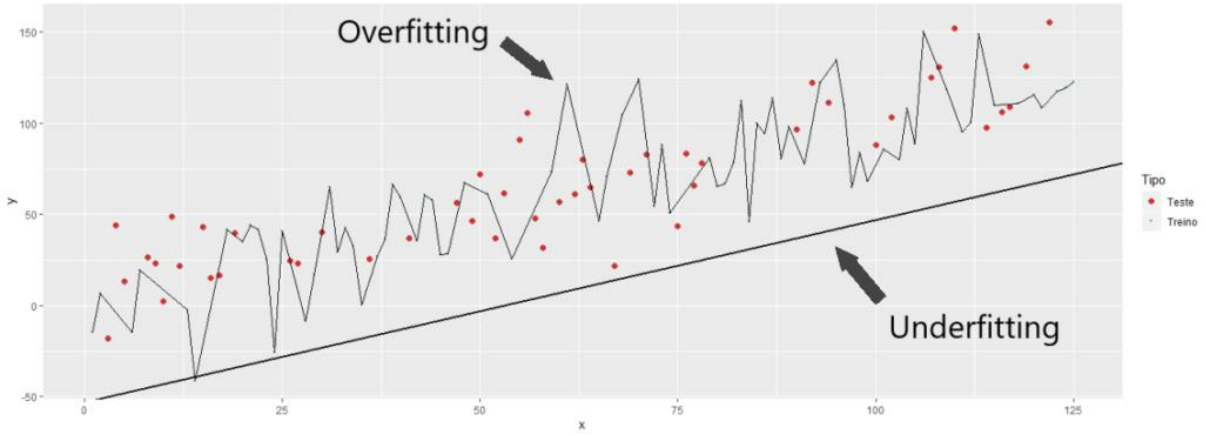


Figura 2.8 – Underfitting e Overfitting
Fonte: Didática Tech (2022)

2.8 – Modelos de Classificação

Para o estudo de caso apresentado utilizou-se algoritmos para resolver problemas de classificação. Aplicou-se o uso da biblioteca *Scikit-Learn* do *Python* e alguns de seus pacotes de modelos preditivos. Para o uso das tecnologias optou-se na utilização de uma aplicação de código aberto conhecida Jupyter Notebook.

A seguir serão apresentados os modelos a serem utilizados nesse trabalho.

2.8.1 – Máquina de Vetor de Suporte

Máquina de Vetor de Suporte ou *Support Vector Machine* (*SVM*), em inglês, é um algoritmo de aprendizado supervisionado muito utilizado para classificação (SCIKIT-LEARN, 2023). A Máquina de Vetor de Suporte divide linearmente os dados de treino (Figura 2.9), de tal maneira que possam ser definidos os vetores de suporte (pontos circulados), as margens (linhas tracejadas) e o limite de margem máximo (linha contínua).

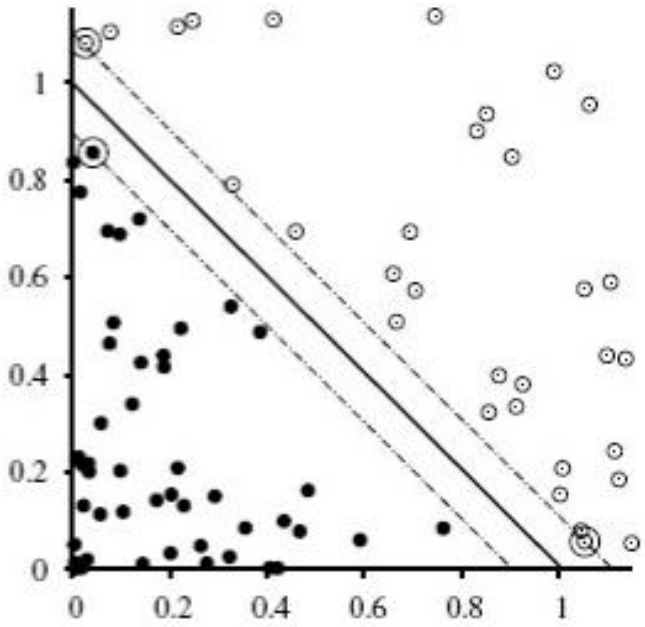


Figura 2.9 – Hiperplano com dados bidimensionais
Fonte: Russell e Norvig (2013)

Há situações em que os dados não são separados de modo linear e, nesses casos, o recurso utilizado é a transformação dos dados para uma dimensão acima do hiperplano atual (RUSSELL; NORVIG, 2013).

A Figura 2.10 mostra a representação gráfica de um conjunto de dados no plano bidimensional (a) e os mesmos dados visualizados em um gráfico tridimensional (b). Em (a), os dados não são linearmente distribuídos, ou seja, não é possível traçar uma reta demarcando os pontos de vetores de suporte. Já em (b), com a aplicação da técnica de transformação de dimensão do plano, é possível identificar separadores lineares (RUSSELL; NORVIG, 2013).

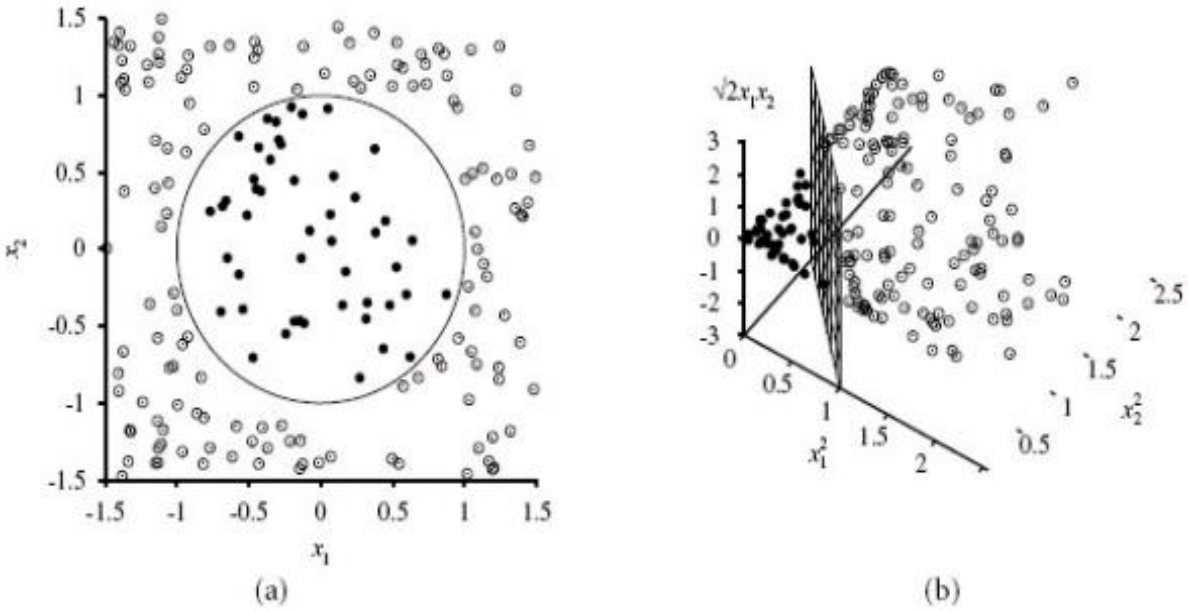


Figura 2.10 – Dados em plano bidimensional e tridimensional
Fonte: Russell e Norvig (2013)

2.8.2 – Regressão Logística

A regressão logística é um algoritmo que analisa estatisticamente a relação entre a variável binária que se quer prever e as demais variáveis relacionadas na base de dados definidas como importante para a modelagem.

De acordo com Fernandes *et al.* (2020), a variável dependente é composta por somente duas classes, sendo padronizada como “1” para o evento pretendido e “0” quando o evento não acontece.

Esse modelo se utiliza da função logística ou função sigmoide, como também é conhecida, para classificar os resultados através da probabilidade de se obter a classificação para as classes binárias. A fórmula da função logística é a seguinte:

$$\mathrm{P}_{+}(x) = \frac{1}{1+e^{-f(x)}} \quad (9)$$

Onde:

P = probabilidade de um evento acontecer

f(x) = função linear de combinação das constantes das variáveis independentes

e = número de Euler

A partir dos resultados obtidos, chega-se na seguinte representação gráfica (figura 2.11) em formato de “S”:

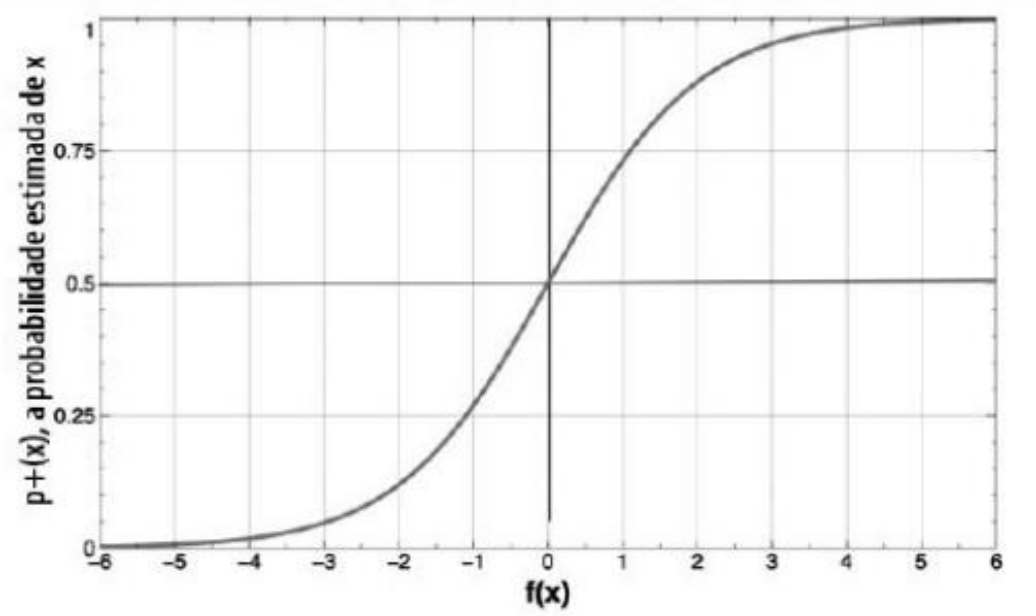


Figura 2.11 – Estimativa de probabilidade de classe de regressão logística
Fonte: Provost e Fawcett (2016)

2.8.3 – Árvore de Decisão

Árvore de Decisão é um método de aprendizado supervisionado que classifica uma variável alvo através de uma estrutura em formato de árvore, realizando perguntas através de nós, ramos e folhas para obter os resultados preditos com baixa complexidade (RUSSELL; NORVIG, 2013).

A estrutura hierárquica (Figura 2.12) é basicamente uma organização de perguntas no formato *se-então-senão*, que pode ser comparada a um fluxograma.

Figura 2.12 – Exemplo de Árvore de Decisão
Fonte: Didática Tech (2022)

O algoritmo de Árvore de Decisão trabalha com a divisão dos dados para chegar no resultado, colocando em prioridade para testes, os atributos mais significativos (RUSSELL; NORVIG, 2013).

Para a escolha dos atributos mais significativos, utiliza-se o método de divisão chamado de Ganho de Informação, que usa o cálculo de entropia, sendo essa uma referência de desordem dentro do espaço amostral. Essa desordem entende-se que, quando alta, as respostas estão muito divididas; quando baixa, as respostas estão mais ordenadas ou tendenciosas para uma determinada classe (Figura 2.13). Abaixo, a fórmula para calcular a entropia:

$$entropia = -\sum_i p_i \log_2 p_i \quad (10)$$

Onde:

p = probabilidade de cada variável

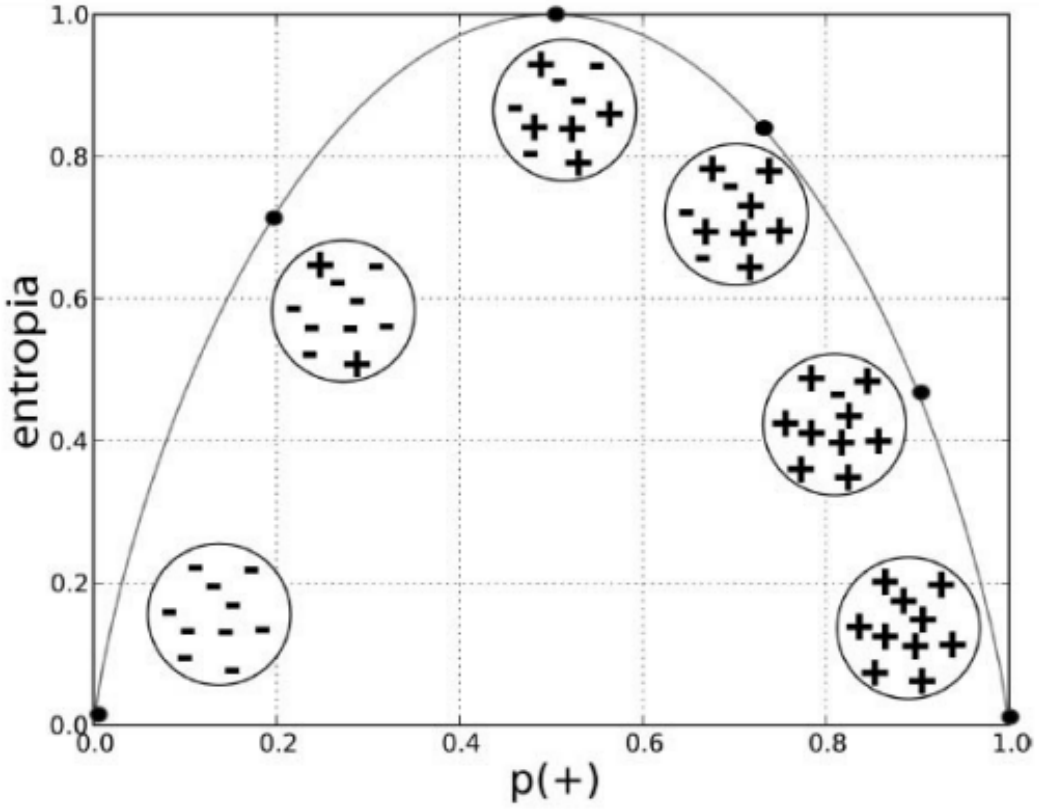


Figura 2.13 – Entropia baseada na probabilidade do evento
Fonte: Provost e Fawcett (2016)

Por fim, com o valor da entropia, aplica-se o resultado à fórmula do Ganho de Informação, que tem por objetivo estabelecer uma relação entre as variáveis alvo e as independentes. Com o resultado, o algoritmo utiliza o melhor valor para definir a variável que iniciará a árvore. Para entendimento, a fórmula do Ganho de Informação é a seguinte:

$$Ganho\ de\ Informação = Entropia_{Pai} - \sum Peso_{Filho} * Entropia_{Filho} \quad (11)$$

Onde:

$Peso_{Filho}$ = relação entre números de amostras do nó filho e do nó pai

2.8.4 – KNN

O *K-Nearest Neighbors (KNN)* ou K-Vizinhos Mais Próximos é um algoritmo supervisionado que pode ser utilizado para resolver problemas regressão e classificação aplicando-se uma técnica baseada em distâncias, comparando o ponto a ser predito com os

pontos dos dados usados para treinamento. Popularmente, essa distância é calculada através da distância Euclidiana.

Segundo Grus (2016), o algoritmo necessita somente da noção de distância e da premissa de que dados próximos um do outro possuem semelhanças.

Um ponto importante a ser observado é que um valor de *k* muito baixo, poderá acarretar em uma classificação equivocada, caso haja muitos ruídos nos dados. Por outro lado, valores muito altos para *k*, também poderá trazer problemas, pois será difícil de se definir as áreas de fronteiras de decisão. Para que esses problemas não aconteçam é necessário que seja definido um valor de *k* ideal conforme cada conjunto de dados. Na figura 2.14, é destacado um exemplo de aplicação do KNN onde o ponto a ser classificado pelo algoritmo está dentro do parâmetro de k = 7 vizinhos mais próximos, sendo a maior quantidade de amostras pertencentes ao rótulo A. Dessa forma, o algoritmo definirá o novo ponto como rótulo A, tendo em vista a distância e o parâmetro definido.

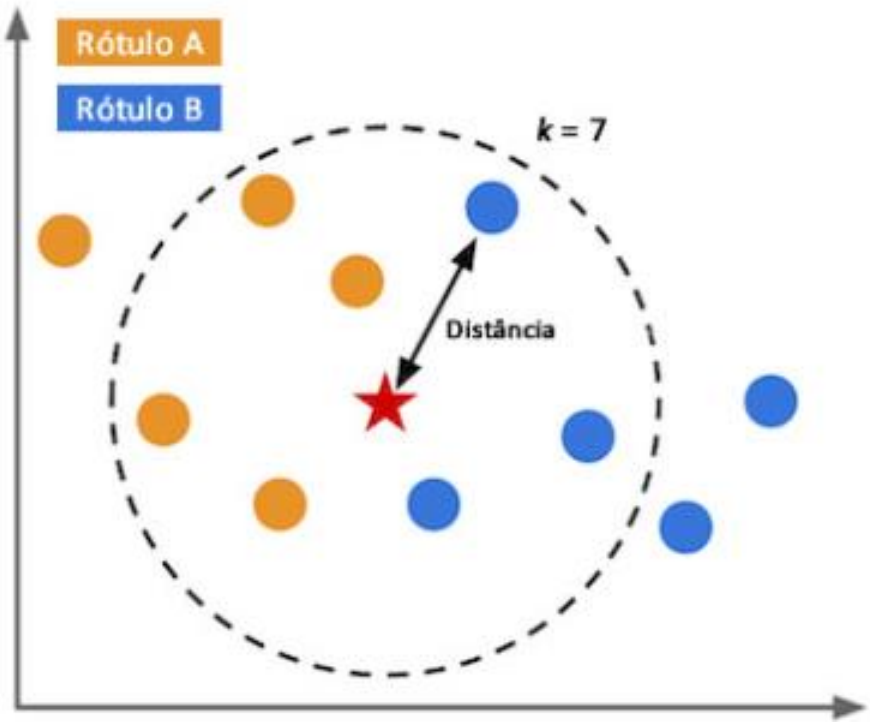


Figura 2.14 – Exemplo de KNN com k = 7
Fonte: Pacheco (2017)

2.8.5 – Random Forest – Florestas Aleatórias

O algoritmo de Florestas Aleatórias (*Random Forest*) é um aprendizado supervisionado utilizado para resolver problemas de classificação e de regressão, que combina várias árvores de decisão (Figura 2.15), selecionando, aleatoriamente, as *features* (características) em todos os nós para definição da profundidade das árvores (HAN; KAMBER; PEI, 2012).

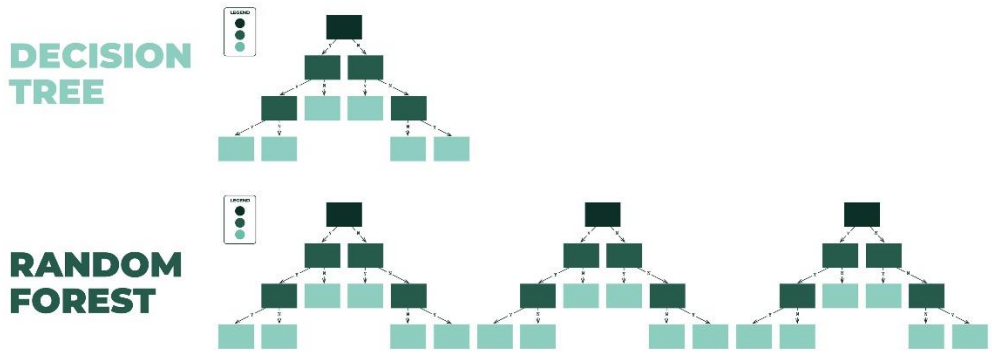


Figura 2.15 – Comparativo entre Árvores de Decisão e Florestas Aleatórias
Fonte: Datadrive (2022)

Ainda segundo Han, Kamber e Pei (2012, p. 383, tradução nossa), “Como as florestas aleatórias consideram muito menos atributos para cada divisão, elas são eficientes em banco de dados muito grandes”.

Esse algoritmo é, em comparação ao de Árvore de Decisão, mais robusto e menos propenso a sofrer *overfitting* (sobreajuste) já que não possui sensibilidade em relação à quantidade de características escolhidas no processo de desenho de cada árvore (HAN; KAMBER; PEI, 2012).

Capítulo 3

Propostas tecnológicas

Nesse capítulo serão apresentadas as propostas de aplicação de algoritmos de *Machine Learning* para resolver problemas de Gerenciamento de Serviços de Tecnologia da Informação (GSTI), voltados para equipamentos de controle de acesso instalados nos prédios da empresa de Telecomunicações OI S/A. A motivação para a realização desse trabalho foi a busca pela otimização do tempo aplicado pelos funcionários na cobrança das execuções da Ordens de Serviço pela prestadora de serviço, de tal modo que haja somente esforço da equipe nas demandas que o algoritmo prever como uma ordem de serviço fechada acima do prazo de 05 (cinco) dias corridos.

3.1 – Base de dados

A base de dados foi fornecida pela área de Segurança Empresarial da Oi, através da exportação das Ordens de Serviço no sistema interno Controle de Informações de Segurança (CIS). Após consulta direta no *software*, o arquivo base_OS_ctrlAc_24out17_27fev23.csv foi exportado para uso nos algoritmos expostos no capítulo dois.

3.2 – Linguagem Python e bibliotecas

Escolheu-se a linguagem *Python* para a elaboração desse trabalho por ser *Open Source* (Código Aberto), possuir códigos de fácil leitura e interpretação, além de contar com uma variedade de bibliotecas para análise de dados, em especial àquelas voltadas para o desenvolvimento de máquinas preditivas para resolver o problema de classificação que compõe o objeto desse estudo.

Optou-se em utilizar o Jupyter Notebook, da distribuição Anaconda, pois destaca-se por ser uma ferramenta organizada em campos de códigos e de textos, além de possuir diversas bibliotecas nativas, sem necessidade de instalação.

As seguintes bibliotecas foram utilizadas para o estudo:

- NumPy: pacote básico para trabalhar com arranjos, vetores e matrizes, contendo uma grande coleção de funções matemáticas;
- Pandas: biblioteca que fornece a estrutura para a manipulação de dados de forma rápida e flexível, através da leitura de dados em formato de *DataFrame*;
- Scikit-Learn: biblioteca contendo diversos algoritmos de *Machine Learning*;
- Matplotlib: biblioteca para visualização de dados;
- Seaborn: outra biblioteca para visualização de dados;
- Datetime: utilizada para manipulação de datas e horas.

3.3 – Pré-processamento

Inicialmente, os dados foram importados para o Jupyter Notebook para a criação de um *DataFrame* através da função *read_csv( )* do Pandas, utilizando-se o separador ponto e vírgula (;) e a codificação "raw_unicode_escape" para a correta leitura dos dados. Concluída a importação, iniciou-se a etapa de pré-processamento.

Utilizando-se a função *shape* do *Python* (figura 3.1) foi possível descobrir o tamanho do *DataFrame*, contendo 11.649 linhas e 23 colunas.

```
#Visualizar qtde de linhas e colunas
df_base_os.shape
```

```
(11649, 23)
```

Figura 3.1 – dimensão do *DataFrame*
Fonte: Autor (2023)

Através da função *head( )* do *Python* foi possível realizar uma análise visual dos dados em cada coluna do *DataFrame*.

Utilizou-se a função *info( )* do *Python* (figura 3.2) para identificar a quantidade de valores não nulos e, principalmente, observar que, somente a coluna 'Chamado' estava em formato numérico. Essa observação foi identificada como um problema para a análise e para os cálculos na elaboração do melhor *dataset* para os treinamentos e testes dos modelos de *Machine Learning*. Dessa forma, para algumas colunas no formato objeto, foi avaliada a necessidade de

transformação para os formatos numérico e data/hora, sendo esse processo demonstrado mais à frente nesse capítulo.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11649 entries, 0 to 11648
Data columns (total 23 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Chamado          11649 non-null  int64
 1   Cadastro         11649 non-null  object
 2   OS               11649 non-null  object
 3   SetorCGS         11649 non-null  object
 4   EmpEnv           11608 non-null  object
 5   Equipamento      11643 non-null  object
 6   Contratada       11649 non-null  object
 7   Tipo             11649 non-null  object
 8   TipoProblema     11649 non-null  object
 9   ResumoRelato     11649 non-null  object
 10  Status           11649 non-null  object
 11  Abertura         11649 non-null  object
 12  Fechamento       11546 non-null  object
 13  SLAFim           11575 non-null  object
 14  Horas            11575 non-null  object
 15  Estação          11643 non-null  object
 16  TipoPredio       11643 non-null  object
 17  UF               11643 non-null  object
 18  Endereço         11643 non-null  object
 19  LoginCadastro    11649 non-null  object
 20  LoginAbertura    11649 non-null  object
 21  LoginFechamento  11367 non-null  object
 22  StatusSimples    11649 non-null  object
dtypes: int64(1), object(22)
memory usage: 2.0+ MB
```

Figura 3.2 – Resultado da função .info( )
Fonte: Autor (2023)

Após a execução dos procedimentos acima, foi feita a verificação de variáveis com valores nulos e valores únicos, para avaliar a necessidade de exclusão ou de permanência de linhas e colunas que estivessem nessas condições. Com isso, identificou-se que havia valores nulos no campo ‘Fechamento’, sendo que esse representa data/hora da conclusão das OS, não sendo possível utilizar esses registros com valores nulos pois impactaria diretamente no treinamento do modelo preditivo.

Outro campo importante que poderia impactar nos resultados do modelo foi a coluna ‘Equipamento’ que apresentou resultados nulos e também foi feita a exclusão dos registros.

Nos valores nulos do campo ‘EmpEnv’ utilizou-se a técnica de preenchimento com o valor que mais se repetiu.

As colunas ‘SLAFim’ e ‘Horas’ foram desprezadas pois, apesar de terem relação com o tempo de fechamento que queremos prever, correspondem a valores antigos de contratos fora de vigência, que não se aplicam mais a realidade atual da empresa.

Considerou-se que as colunas 'SLAFim', 'Horas', 'SetorCGS', 'Contratada', 'Tipo', 'LoginCadastro', 'LoginAbertura' e 'LoginFechamento' não seriam necessárias para o modelo e também foram excluídas.

As colunas “Abertura” e “Fechamento” continham informações de data e hora no mesmo campo, além de possuírem o tipo da variável como objeto. Foi feita a conversão das colunas “Abertura” e “Fechamento” para tipo *datetime* e criadas as colunas ‘Data_Abertura’, ‘Data_Fechamento’, ‘Hora_Abertura’ e ‘Hora_Fechamento’, também sendo executada a conversão para tipo *datetime*.

Após os ajustes feitos no *DataFrame*, criou-se a coluna ‘Tempo_Conclusão’, mas como o tipo da variável era *timedelta*, precisou-se criar a variável ‘Tempo_Conclusão_dia’ para converter os valores para inteiro.

Com base no tempo de conclusão obtido, gerou-se a variável alvo denominada ‘DentroPrazo’, que passou a registrar a classificação em OS fechadas dentro do prazo, quando concluídas em até 05 (cinco) dias e, fora do prazo, quando concluídas acima de 05 (cinco) dias.

Por fim, como última etapa do pré-processamento, transformou-se as variáveis categóricas 'EmpEnv', 'Equipamento', 'TipoProblema', 'ResumoRelato', 'StatusSimples', 'Estação', 'TipoPredio' e 'UF' em variáveis numéricas, através da aplicação da função *Label Encoder* do pacote *preprocessing* do *Scikit-Learn*. Necessitou-se realizar essa transformação pois os algoritmos de *Machine Learning* não trabalham bem com dados tipo objeto.

3.4 – Análise dos dados

3.4.1 – Análise exploratória dos dados

Concluída a etapa de pré-processamento, iniciou-se a uma análise exploratória dos dados para conhecimento da base que está sendo utilizada. Executaram-se as funções “value_counts( )” e “*groupby”* para contabilizar os valores repetidos nas colunas ‘ResumoRelato’, ‘TipoProblema’, ‘EmpEnv’ e ‘StatusSimples’.

Em ‘ResumoRelato’, observou-se o agrupamento dos tipos de soluções para resolução do problema de cada OS.

Em ‘TipoProblema’, verificou-se a categorização do problema envolvendo determinado equipamento relacionado na OS.

Na coluna “EmpEnv”, observou-se a abertura das OS em nome de três empresas: Comodato V.tal, NovaOi e V.tal.

Verificou-se que a coluna, ‘StatusSimples’, após as etapas de pré-processamento, possuía somente as OS com status fechada. Para desenvolvimento do modelo preditivo somente poderiam ser empregadas as ordens de status fechada, pois como não existe a data de fechamento em outros status, esses dados não poderiam ser utilizados como referência. Esses registros foram eliminados na etapa de exclusão de valores nulos quando identificados 103 registros nulos na coluna ‘Fechamento’, que representa a data/hora de conclusão de OS.

3.4.2 – Análise preditiva dos dados

A análise preditiva está voltada para a cobrança da execução de OS pela OI, com o objetivo de priorizar os atendimentos previstos pelos algoritmos como resolução fora do prazo, otimizando assim a quantidade de funcionários previstos para esse tipo de atividade. Já para as placas de controle de acesso cujos modelos são *iSTAR Edge*, da fabricante *Tyco Security*, não foi possível estabelecer uma análise preditiva, pois o sistema que gerencia esses equipamentos, conhecido no mercado de controle de acesso como C-Cure 9000, não fornece dados suficientes para a elaboração de um modelo que preveja defeitos nesses equipamentos. Em consulta direta à prestadora de serviço, verificou-se também que não há dados suficientes nas OS para criação de um modelo preditivo que identifique um padrão nos defeitos das placas, inviabilizando assim uma substituição antecipada de equipamentos antes da apresentação de falhas e defeitos. A grande dificuldade é a parte cultural dos funcionários da terceirizada, que registram o detalhamento técnico das falhas e defeitos, que poderiam ser utilizados para o modelo de predição voltado diretamente ao funcionamento das placas de controle de acesso.

Ainda assim, com a aplicação da análise preditiva considerando somente o tempo de conclusão das OS, haverá um ganho operacional e financeiro para a OI através da redução da equipe de cobrança, já que os chamados não serão tratados em sua totalidade, mas sim através das OS definidas pelo algoritmo como fechamento fora do prazo estabelecido.

3.5 – Criação dos modelos preditivos

A etapa de criação de modelos preditivos é a mais esperada, pois é o momento em que toda a parte teórica apresentada sobre os algoritmos de *Machine Learning* é aplicada, com objetivo principal de prever o resultado da variável alvo 'DentroPrazo'. Para atingir esse objetivo, a base de dados tratada foi dividida em dados de treino (70%) e de teste (30%). Após a divisão, foram criados os classificadores utilizando os algoritmos *SVM*, *LogisticRegression*, *DecisionTreeClassifier*, *KNeighborsClassifier* e *RandomForestClassifier*. E, por fim, foi feita a avaliação das máquinas preditivas criadas, através das métricas: acurácia, Precisão, Revocação e F1-Score.

O algoritmo *SVM* utiliza o conceito de separar os dados linearmente, para que haja uma divisão da classificação prevista. Acontece que, nem sempre, é possível conseguir uma distribuição linear em um mesmo hiperplano sendo necessária a utilização da função *kernel*, que transforma um conjunto de dados de treino não linear em uma equação linear através da criação de espaços multidimensionais. Um exemplo clássico na área de Ciência de Dados é o caso do *dataset* Íris que traz um conjunto de medidas de pétalas e sépalas de três espécies de flores: setosa, versicolor e virginica. A figura 3.3 representa a separação linear comparando o comprimento e largura das pétalas dos tipos de flores. Nessa separação linear é possível visualizar que há previsões com os dados de teste feitas incorretamente nos tipos de flores versicolor e virginica. Para minimizar esses erros, utiliza-se o recurso de *kernel*, obtendo-se uma representação mais suave no limite de decisão para um valor baixo no parâmetro *gamma* (Figura 3.4) ou um limite estreito de decisão ajustando-se o parâmetro *gamma* para um alto valor (Figura 3.5). Nesse ajuste, deve ser observada a otimização do parâmetro para que não haja a ocorrência de *overfitting*, pois um alto valor pode treinar de forma excelente e não generalizar com os dados de teste.

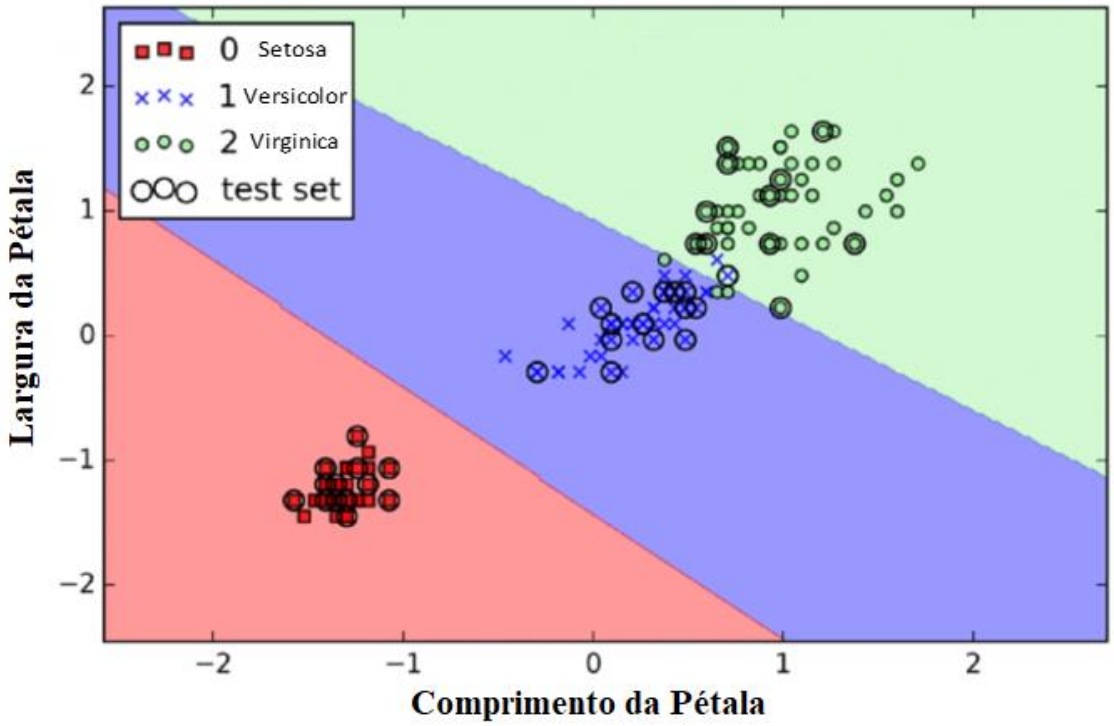


Figura 3.3 – Separação linear dos dados das pétalas
Fonte: Aprender Data Science (2023)

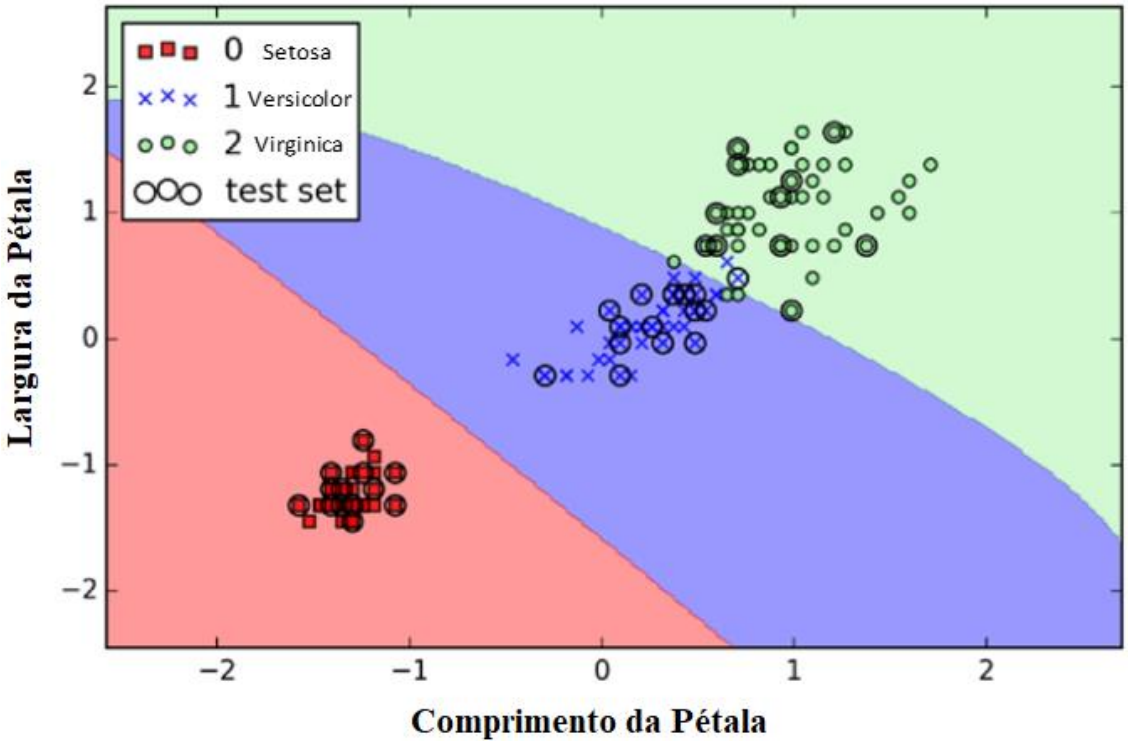


Figura 3.4 – Separação por kernel com gamma reduzido
Fonte: Aprender Data Science (2023)

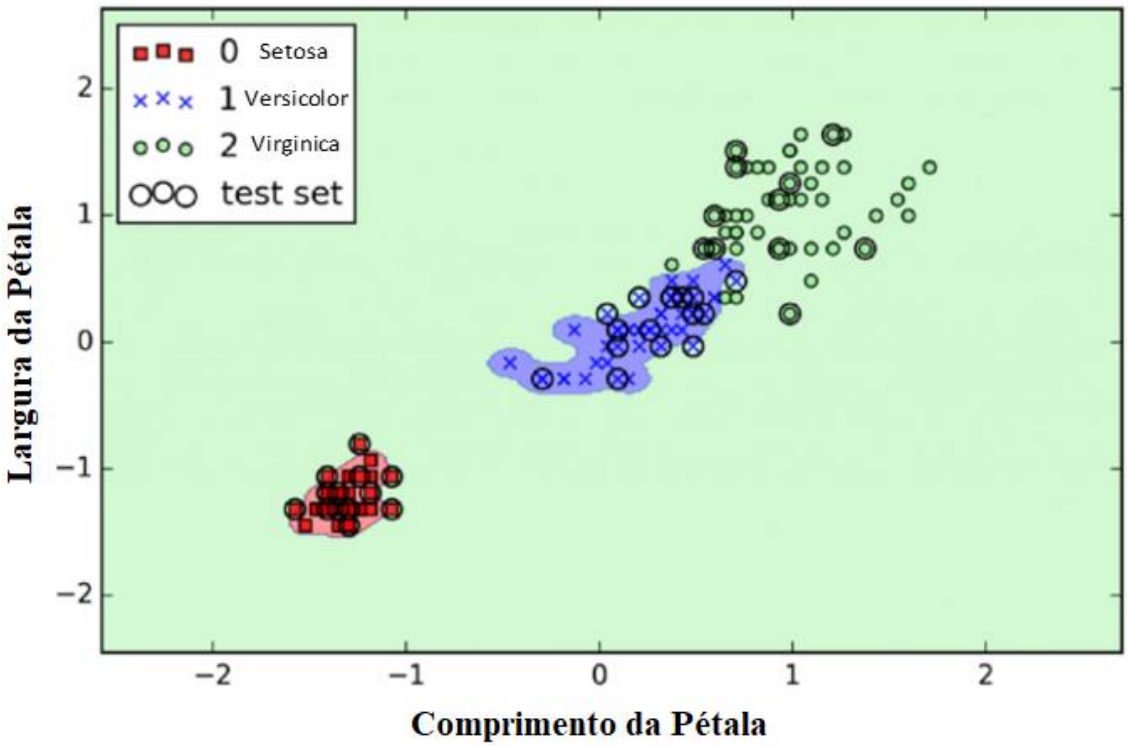


Figura 3.5 – Separação por kernel com gamma elevado
Fonte: Aprender Data Science (2023)

No algoritmo de Regressão Logística, o uso da função sigmoide apresenta graficamente a probabilidade da variável dependente pertencer a uma determinada classe. No mesmo *dataset* de flores Iris pode-se entender a aplicação desse algoritmo. A figura 3.6 representa a largura das pétalas de duas espécies, onde o eixo x recebe os valores do comprimento das pétalas e o eixo y recebe a probabilidade de um valor ser classificado como espécie setosa ou versicolor.

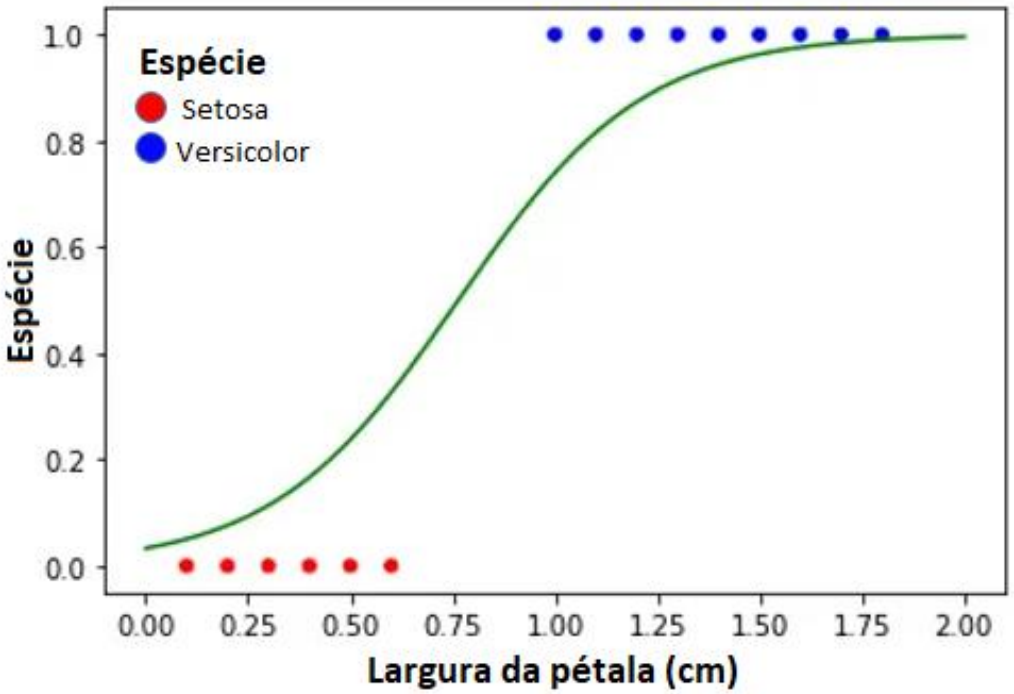


Figura 3.6 – Regressão logística para classificação de espécies de flores Iris
Fonte: Autor (2023)

Com a utilização do algoritmo de Árvore de Decisão é possível classificar a variável alvo através de perguntas em cada nó da estrutura, obtendo-se a folha como resposta final para a classificação de acordo com as demais características do *dataset*. Ainda considerando o exemplo do *dataset* Iris, para uma amostra com comprimento da pétala (*PetalLength*) igual a 4,2 e a largura da pétala (*PetalWidth*) igual a 1,2, é possível visualizar a árvore de decisão (Figura 3.7) formada pelo algoritmo para a classificação da espécie de flor Iris, onde as decisões são tomadas em cada nó até a obtenção do resultado como espécie versicolor.

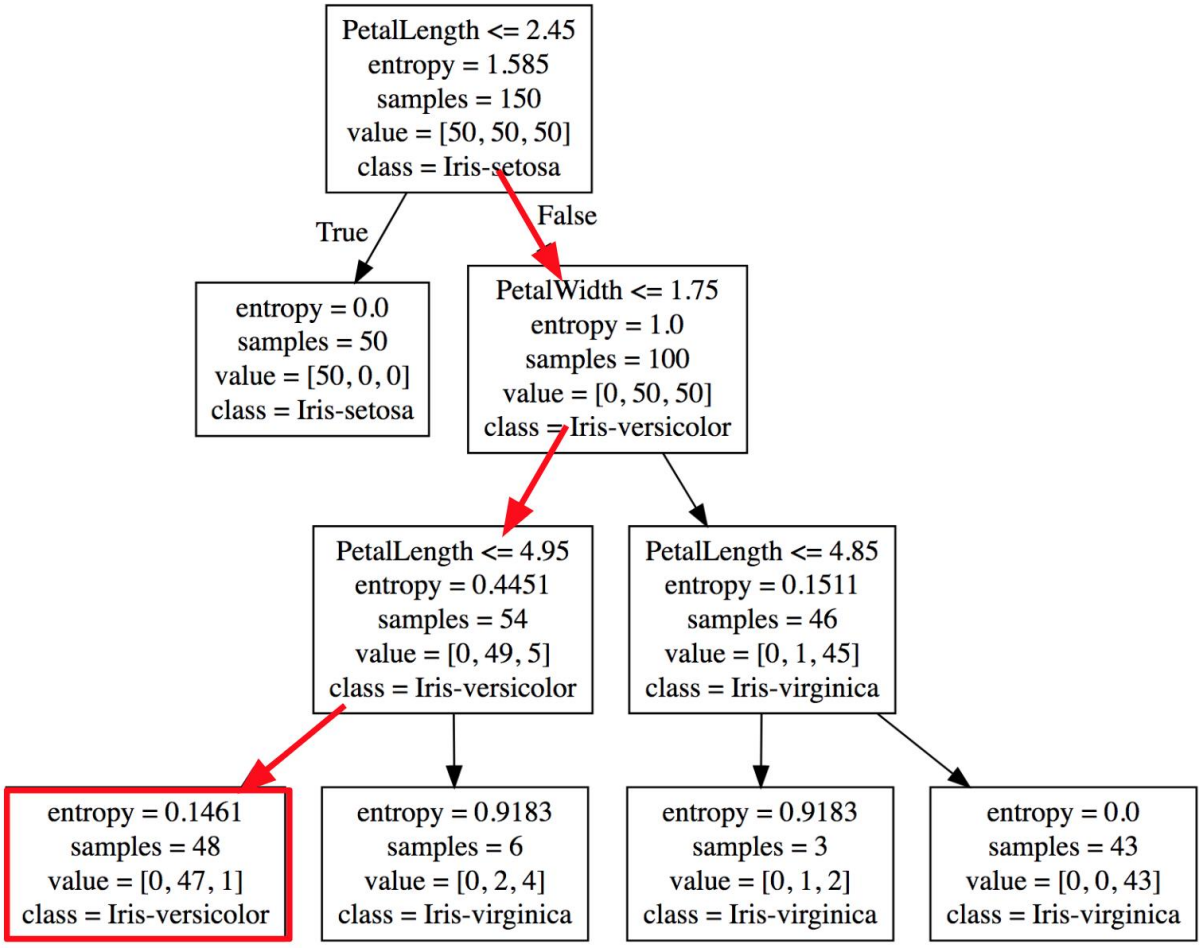


Figura 3.7 – Árvore de Decisão para classificação de espécie de flor Iris
Fonte: Sakurai (2023)

O algoritmo *K-Nearest Neighbors* utiliza a técnica de proximidade com os dados de treinamento para prever o resultado com os dados de teste. Na figura 3.8, pode-se verificar um exemplo gráfico de classificação utilizando *K-NN* com a utilização de *k* igual a 15. A utilização do *K-NN* é adequada com há uma dimensionalidade pequena das variáveis de entrada. Quando há uma grande quantidade de variáveis poderá ocorrer *overfitting*, prejudicando assim a performance do algoritmo.

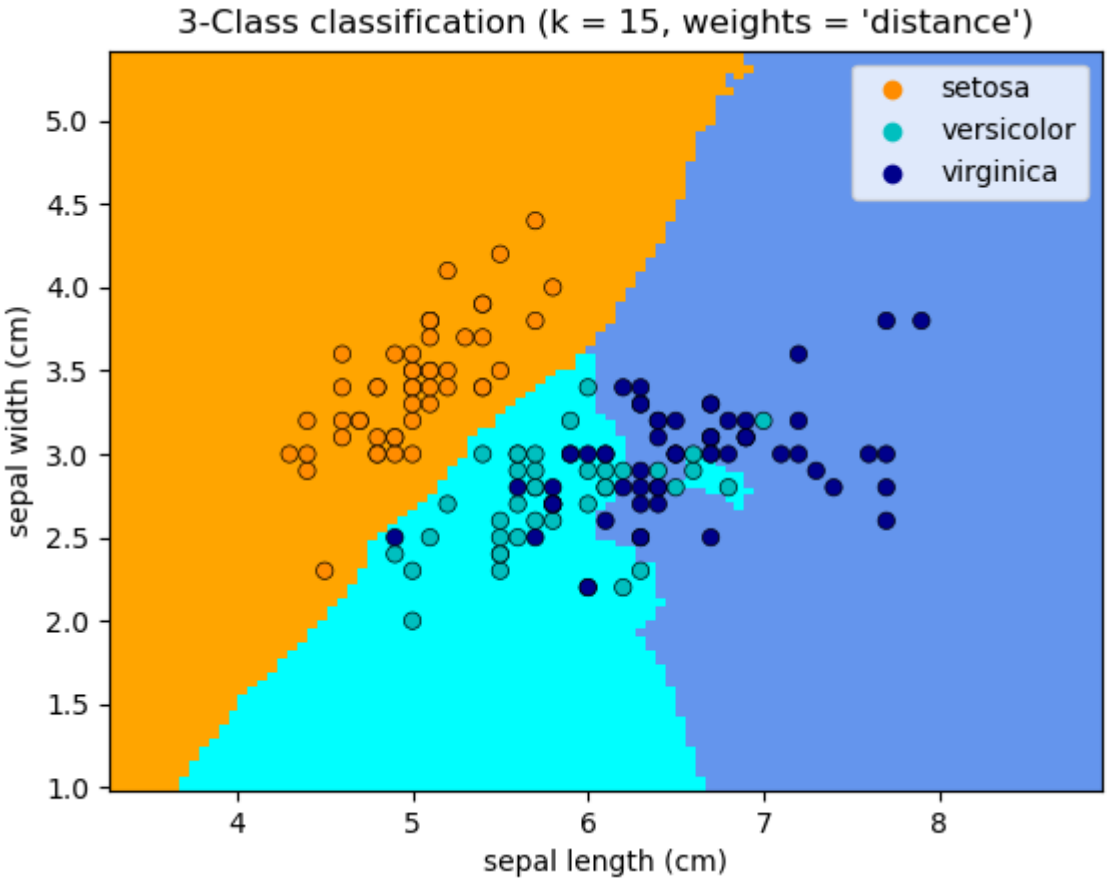


Figura 3.8 – K-NN para classificação de espécie de flor Iris
Fonte: Scikit-Learn (2023)

Por último, o algoritmo de *Random Forest* cria várias árvores de decisão de acordo com a quantidade de estimadores estabelecidos como parâmetro, fazendo uso de amostras aleatórias de linhas e colunas e não de todo o *dataset*. Isso pode produzir um resultado de uma árvore mais fraco em relação à árvore de decisão feita com todo o conjunto de dados, mas garante uma melhor performance no resultado final realizando a média entre todas as pequenas árvores. Quando se estabelece como parâmetro a quantidade de 300 estimadores, por exemplo, serão geradas 300 árvores de decisão com as amostras aleatoriamente escolhidas pelo algoritmo e, dessa forma, seria inviável uma representação gráfica para um conjunto de dados. A figura 3.9 exemplifica a representação gráfica do algoritmo *random forest*.

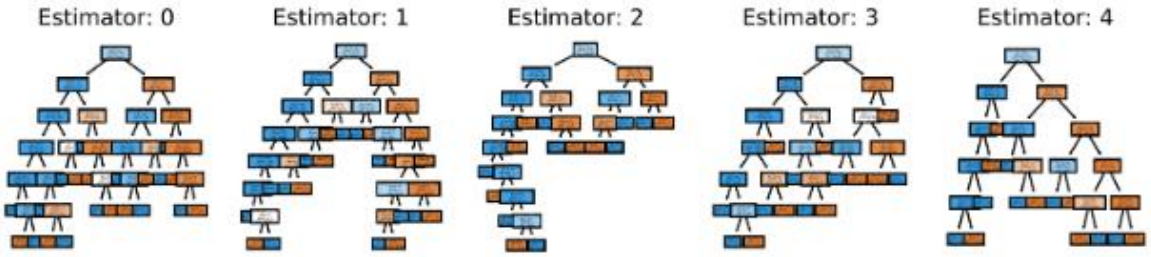


Figura 3.9 – Árvores de decisão criadas pelo Random Forest com estimador = 5
Fonte: Galarnyk (2023)

Capítulo 4

Resultados Obtidos

4.1 – Resultados da Análise Exploratória

Com a análise das variáveis categóricas, observaram-se os seguintes aspectos:

- Em ‘ResumoRelato’, consta o agrupamento de 69 tipos de soluções para resolução do problema de cada OS, destacando-se o tipo “EM BRANCO” pois representou 31,92% da base para essa classificação, o que pode demonstrar uma falha da operação em melhor categorizar a variável. Tal falha pode ter influenciado negativamente nos resultados;

- Em ‘TipoProblema’, verificou-se que 51,74% (coluna 8 da Figura 4.1) são categorizados como “Sem conexão – Equipamento off-line” e que 19,92% (coluna 7 da Figura 4.1) são atribuídos a “Outros”, podendo caracterizar uma generalização nas aberturas de OS que impactou negativamente no processo de aprendizagem dos algoritmos;

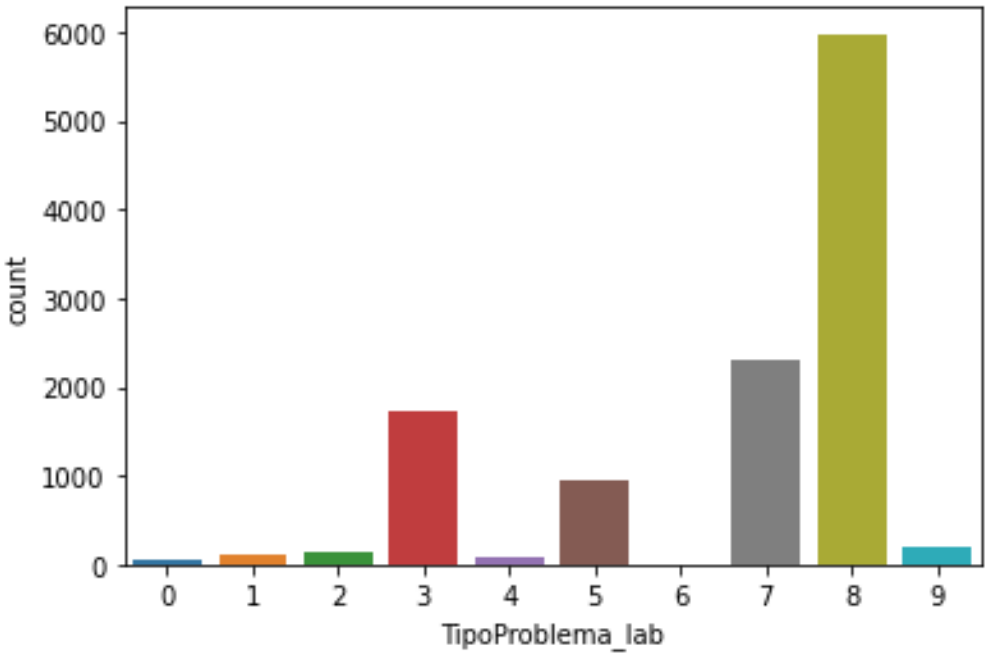


Figura 4.1 – Distribuição dos tipos de problema das OS
Fonte: Autor (2023)

- Na coluna ‘EmpEnv’, observou-se a seguinte distribuição das OS por empresa: 66,40% - Comodato V.Tal, 19,30% - V.tal e 14,30% NovaOi (Figura 4.2);

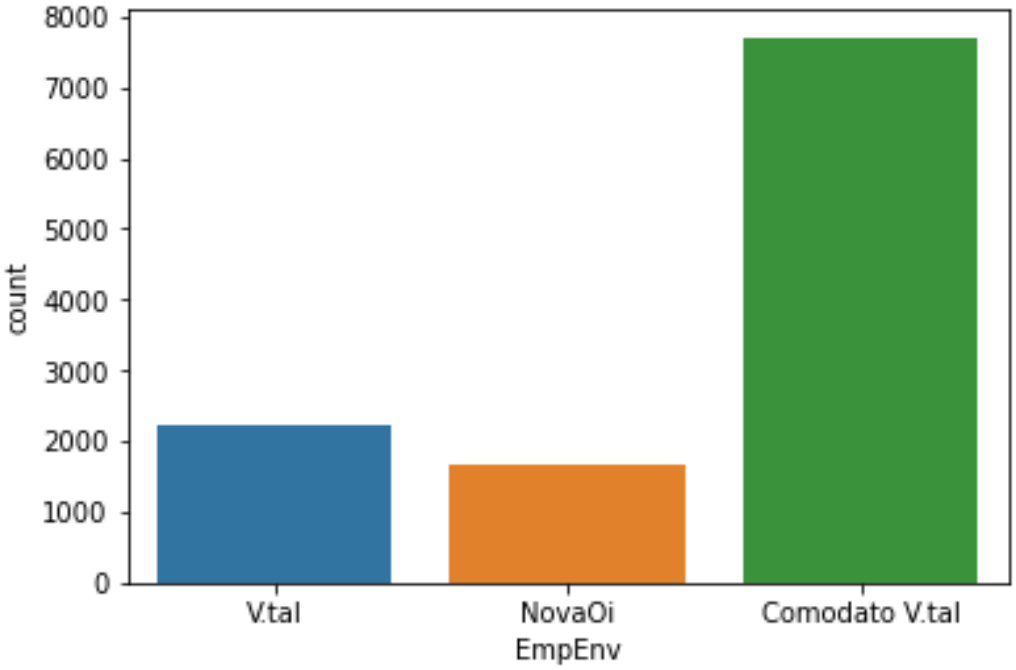


Figura 4.2 - OS por empresa
Fonte: Autor (2023)

- Com a criação da variável alvo ‘DentroPrazo’, inferiu-se que 56,67% das OS fecharam dentro dos 05 (cinco) dias e 43,33% foram fechadas fora deste prazo (Figura 4.3).

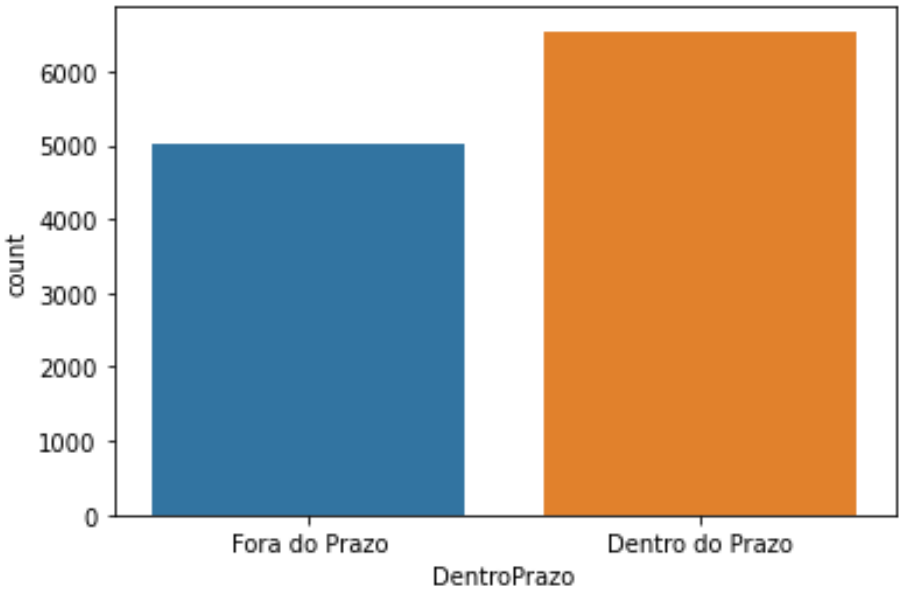


Figura 4.3 – Fechamento OS no prazo
Fonte: Autor (2023)

4.2 – Avaliação dos algoritmos de Machine Learning

Nesse trabalho, os algoritmos abordados no capítulo 2 – Embasamento Teórico, foram executados com o suporte da linguagem *Python* e, principalmente, da biblioteca *Scikit-Learn* onde utilizou-se a divisão da base de dados em treino e teste, de maneira que as predições pudessem trazer um resultado satisfatório.

A máquina preditiva criada com o algoritmo *SVM* obteve uma acurácia de 57% e uma pontuação média em F1-score de 43% (Tabela 4.1), o que demonstra uma assertividade baixa com a utilização desse método de classificação.

Tabela 4.1 – Avaliação do algoritmo SVM

Acurácia = 0.57	Precisão	Recall	F1-score	Suporte
Dentro do Prazo	0.57	0.95	0.72	1957
Fora do Prazo	0.56	0.08	0.14	1506
Média Macro	0.52	0.57	0.43	3463

Fonte: Autor (2023)

A matriz de confusão (Figura 4.4) representou a divisão dos resultados previstos pelo algoritmo SVM.

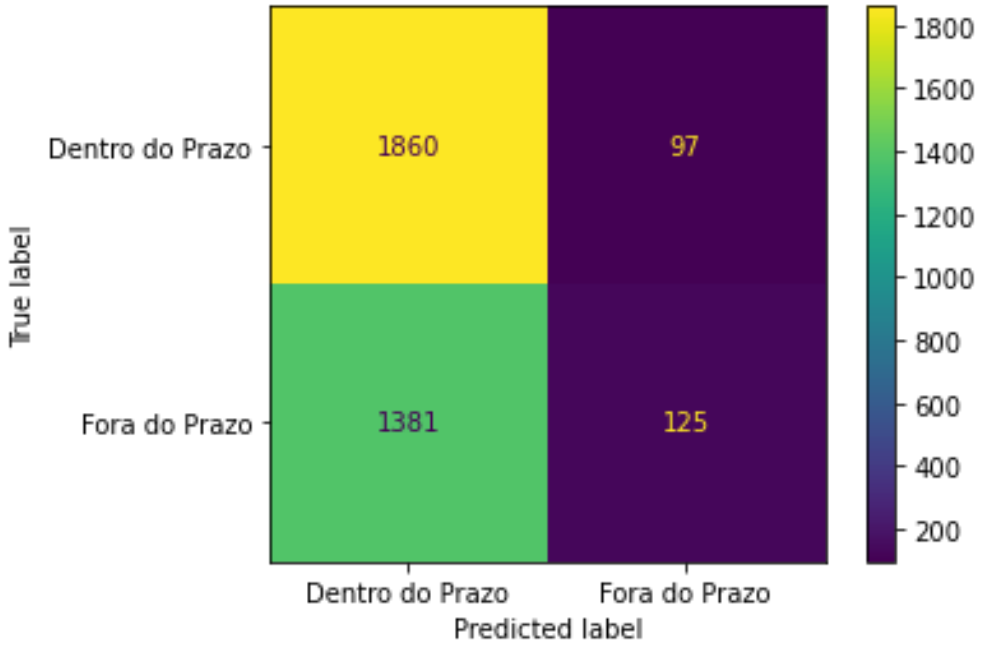


Figura 4.4 – Matriz de Confusão - SVM
Fonte: Autor (2023)

Com o algoritmo de Regressão Logística, conquistou-se um resultado com melhor precisão, em relação ao SVM, porém, ainda uma avaliação insatisfatória, haja vista as métricas de acurácia em 64% e de pontuação média em F1-score em 54% (Tabela 4.2).

Tabela 4.2 – Avaliação do algoritmo Regressão Logística

Acurácia = 0.64	Precisão	Recall	F1-score	Suporte
Dentro do Prazo	0.61	0.98	0.76	1957
Fora do Prazo	0.90	0.19	0.32	1506
Média	0.76	0.59	0.54	3463

Fonte: Autor (2023)

Na matriz de confusão desse algoritmo (Figura 4.5) é possível perceber a leve mudança das quantidades previstas corretamente nas classes “Dentro do Prazo” e “Fora do Prazo”, ainda assim 1214 ocorrências não seriam priorizadas na cobrança de resolução, já que foram previstas como “Dentro do Prazo” e, na realidade, pertencem a classe “Fora do Prazo”.

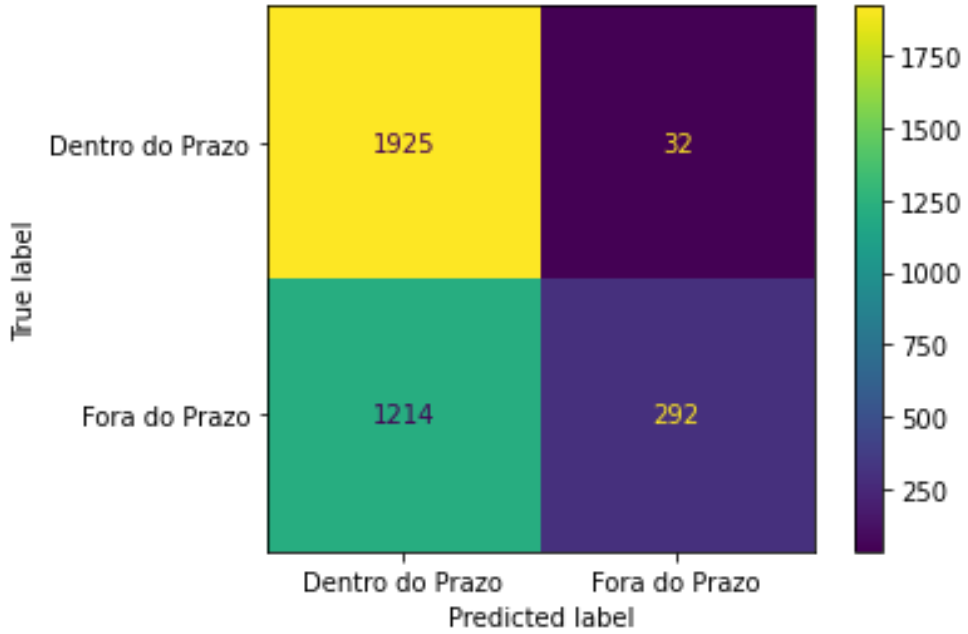


Figura 4.5 – Matriz de Confusão - Regressão Logística
Fonte: Autor (2023)

Executando-se o algoritmo de Árvore de Decisão, chegou-se ao resultado de acurácia de 63% e de pontuação média de F1-score de 62% (Tabela 4.3), demonstrando assim uma melhora em relação aos dois algoritmos anteriormente testados (SVM e Regressão Logística).

Tabela 4.3 – Avaliação do algoritmo Árvore de Decisão

Acurácia = 0.63	Precisão	Recall	F1-score	Suporte
Dentro do Prazo	0.66	0.73	0.69	1957
Fora do Prazo	0.59	0.51	0.55	1506
Média	0.63	0.62	0.62	3463

Fonte: Autor (2023)

Através da matriz de confusão do algoritmo Árvore de Decisão (Figura 4.6) percebe-se a mudança do comportamento dos dados previstos, principalmente a redução das quantidades da classe predita como “Dentro do Prazo” que são da classe real “Fora do Prazo”.

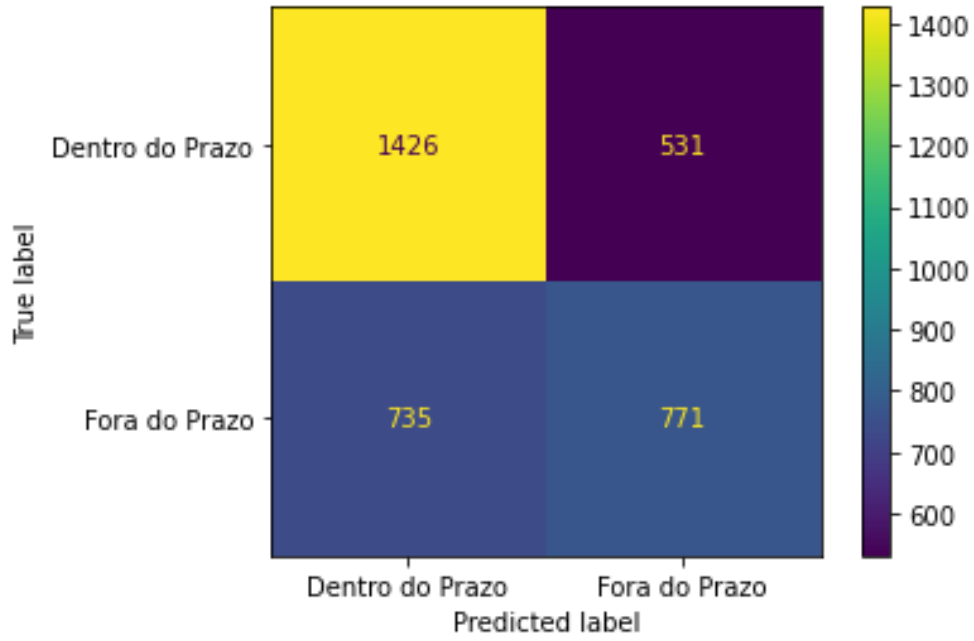


Figura 4.6 – Matriz de Confusão - Árvore de Decisão
Fonte: Autor (2023)

Outro importante algoritmo de classificação utilizado foi o KNN que obteve resultados piores em relação à Árvore de Decisão, tendo a acurácia de 61% e a média de F1-score de 59% (Tabela 4.4).

Tabela 4.4 – Avaliação do algoritmo - KNN

Acurácia = 0.61	Precisão	Recall	F1-score	Suporte
Dentro do Prazo	0.63	0.72	0.68	1957
Fora do Prazo	0.56	0.46	0.50	1506
Média	0.60	0.59	0.59	3463

Fonte: Autor (2023)

Verificando-se a matriz de confusão obtida com os dados de teste para o algoritmo KNN (Figura 4.7), observou-se uma distribuição minimamente inferior em relação aos resultados obtidos através da Árvore de Decisão.

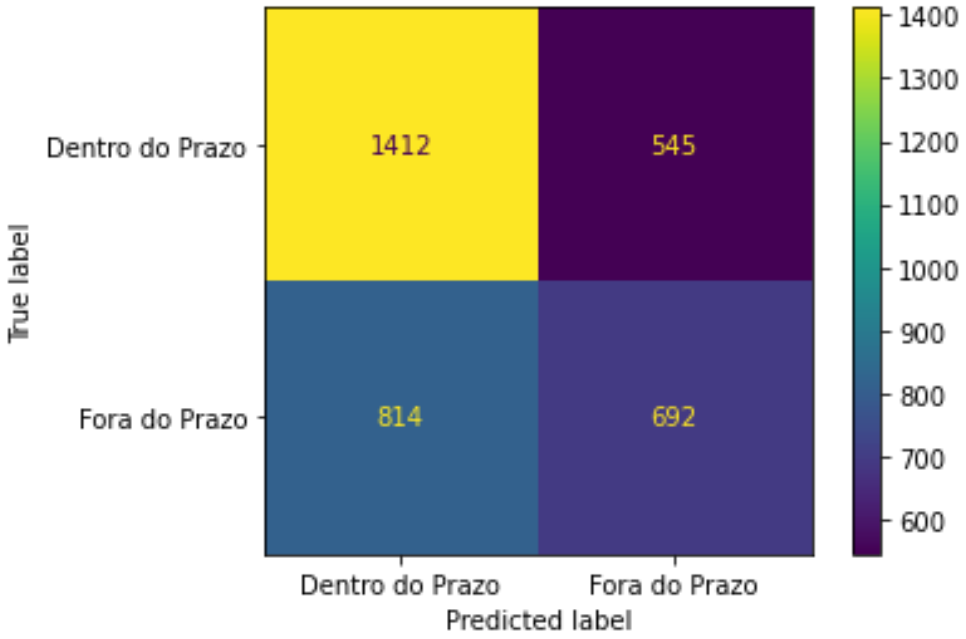


Figura 4.7 – Matriz de Confusão – KNN
Fonte: Autor (2023)

O último algoritmo utilizado foi o *Random Forest* que apresentou o melhor resultado em comparação aos demais algoritmos utilizados nesse trabalho. Obteve-se a acurácia de 65% e a média de F1-score de 64% (Tabela 4.5).

Tabela 4.5 – Avaliação do algoritmo – *Random Forest*

Acurácia = 0.65	Precisão	Recall	F1-score	Suporte
Dentro do Prazo	0.68	0.73	0.70	1957
Fora do Prazo	0.61	0.55	0.58	1506
Média	0.64	0.64	0.64	3463

Fonte: Autor (2023)

Ao plotar o gráfico da matriz de confusão para o algoritmo *Random Forest* (Figura 4.8) percebe-se a melhor distribuição dos dados nas classes previstas em relação às classes reais. Sendo assim, o algoritmo *Random Forest* demonstrou ser a melhor opção para prever a classificação de tempo de encerramento de ordens de serviço da base de dados utilizada.

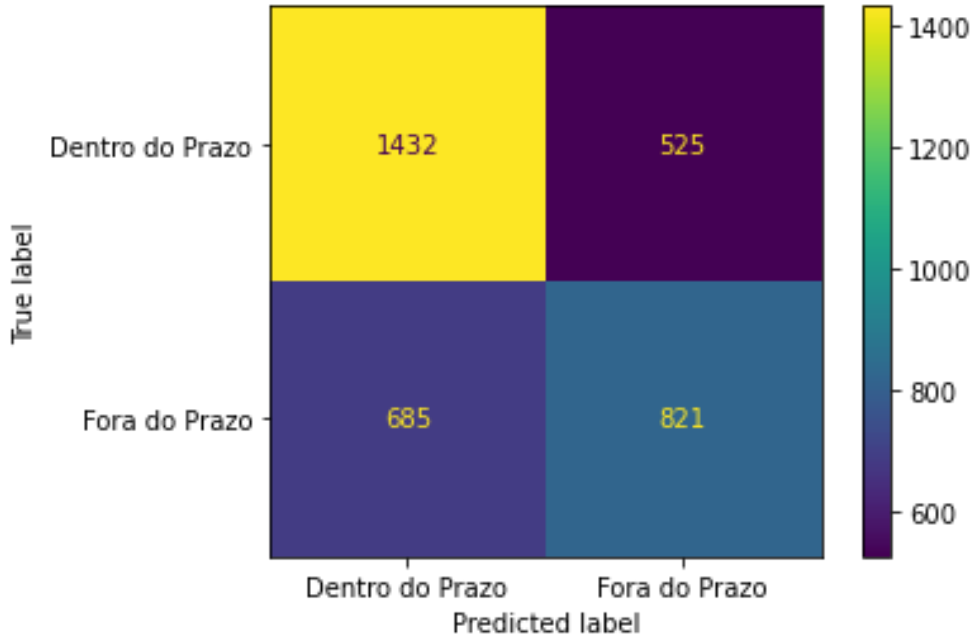


Figura 4.8 – Matriz de Confusão – *Random Forest*
Fonte: Autor (2023)

Um resumo das avaliações de cada algoritmo, contendo o percentual da acurácia e da média de F1-score, pode ser observado na tabela 4.6 abaixo:

Tabela 4.6 – Resumo das avaliações dos algoritmos

	Acurácia	**F1-score**
SVM	0.57	0.43
Regressão Logística	0.64	0.54
Árvore de Decisão	0.63	0.62
KNN	0.61	0.59
Random Forest	0.65	0.64

Fonte: Autor (2023)

O SVM teve o pior resultado devido à falta de padronização dos dados, ponto fundamental para o sucesso deste algoritmo. O resultado do algoritmo de Regressão Logística teve a acurácia próxima ao *Random Forest*, mas F1-Score menos eficaz, o que pode ser ocasionado pela dificuldade do próprio algoritmo em lidar com dados não lineares. O *K-NN* necessita de uma excelente padronização nos dados, podendo ter sido o motivo da avaliação de desempenho inferior ao algoritmo melhor classificado. O algoritmo de Árvore de Decisão demonstrou ser o segundo melhor desempenho pois exige menor preparação dos dados e por ser menos sensível a ruídos. Como se espera, o algoritmo *Random Forest* apresenta o melhor resultado pela própria fundamentação de criar várias árvores de decisão extraídas de amostras da base de dados de maneira aleatória e com um resultado médio melhor em relação a uma única árvore utilizando todos os dados, além do benefício de ter um bom desempenho com dados complexos e não lineares e ser menos propenso a *overfitting*.

Capítulo 5

Conclusão e Trabalhos Futuros

5.1 – Conclusão

Esse trabalho apresentou a utilização de algoritmos de classificação para prever as classes das ordens de serviço de manutenção de equipamentos de controle de acesso da empresa de telecomunicações OI S/A. O primeiro objetivo era identificar processos de melhoria no tratamento feito a cada ordem de serviço, cumprindo um dos princípios orientadores da biblioteca ITIL 4, cujo teor concentra-se na otimização e na automatização de processos. A aplicação dos algoritmos de *Machine Learning* nesse estudo cumpre exatamente as recomendações contidas na documentação.

O segundo objetivo foi aplicar técnicas de pré-processamento para a preparação da base de dados antes do envio do *DataFrame* para os modelos de predição. Estas foram as técnicas utilizadas: eliminação de linhas com valores nulos, preenchimento de valores nulos, exclusão de colunas desnecessárias ao modelo, alteração de tipos de dados envolvendo data/hora, criação de novas colunas para obtenção da variável alvo e aplicação do *Label Encoder* para transformação de variáveis categóricas em numéricas. Entende-se que esse objetivo foi alcançado para uma melhor execução dos algoritmos preditivos, mas observou-se uma deficiência na base de dados pois algumas variáveis continham dados mal categorizados. Nos valores da variável 'ResumoRelato', um desses foi classificado como "em branco" representando um total de 31,92% da base e, na variável 'TipoProblema', mais de 70% dos registros concentram-se nos valores "Sem conexão – Equipamento off-line" e "Outros". Essas observações podem ter impactado diretamente no resultado dos algoritmos.

O terceiro objetivo foi cumprido quando se obteve o melhor algoritmo para a predição através verificação métricas de acurácia, precisão, *recall* e *F1-score*. Dos algoritmos avaliados, o *Random Forest* foi o escolhido por obter 65% de acurácia e 64% na média de *F1-score*.

Conclui-se que, apesar de não se obter um valor alto nos resultados de avaliação do algoritmo *Random Forest*, o *recall* apresenta-se com 55% para a classe real "Fora do Prazo", ou seja, em 55% das vezes, o algoritmo previu corretamente que a ordem de serviço fecharia

fora do prazo. Sendo assim, torna-se possível que a área de Segurança Empresarial da OI defina a priorização de atendimentos conforme predição do algoritmo.

5.2 – Trabalhos Futuros

Recomenda-se uma melhor padronização da base de dados, principalmente a coluna ‘ResumoRelato’, onde os dados ficam abertos para digitação dos usuários e não para seleção dos principais relatos possíveis após o fechamento de uma OS. Uma lista prévia para escolha do usuário impedirá uma digitação incorreta, nomenclaturas erradas ou o uso de textos genéricos que não definam exatamente o que foi tratado. Essa padronização poderá melhorar os resultados obtidos pelos algoritmos.

Sugere-se também o estudo de melhoria dos possíveis valores na variável ‘TipoProblema’ pois há uma alta concentração de ocorrências em apenas dois valores, a saber: “Sem conexão – Equipamento off-line” e “Outros”. A melhoria nas entradas dessas variáveis pode trazer benefícios ao processo de avaliação dos algoritmos.

Para a aplicação prática desse trabalho, aconselha-se realizar a importação dos dados diretamente do banco de dados.

Por fim, propõe-se a utilização de outros algoritmos de classificação como *Gradient Boosting, Extreme Gradient Boosting (XGB), Light Gradient Boosting Machine (LGBM) e Multi Layer Perceptron Classifier (MLPC).*

Referências Bibliográficas

AXELOS. **ITIL Foundation**. 4. ed. Londres: The Stationery Office, 2019.

BON, Jan Van; VERHEIJEN, Tieneke. **Fundamentos do gerenciamento de serviços de TI baseado na ITIL.** Holanda: Wilco Printers, 2006.

CHIARI, Renê. **Vale a pena investir na certificação ITIL4® Foundation?**, 2019. Disponível em: https://www.itsmnapratica.com.br/itil-4-vale-a-pena/. Acesso em: 1 mar. 2023.

DATADRIVE. **Data Science Explained:** Random Forests, 2022. Disponível em: https://godatadrive.com/blog/random-forests. Acesso em: 31 mar. 2023.

DATAGEEKS. **Conheça as Etapas do Pré-Processamento de dados,** 2019. Disponível em: https://www.datageeks.com.br/pre-processamento-de-dados/ Acesso em: 27 mar. 2023.

DOCKET. **Novas funcionalidades da R.E.A.**, 2020. Disponível em: https://blog.docket.com.br/novas-funcionalidades-da-rea/ Acesso em: 27 mar. 2023.

FERNANDES, A. A. T. et al.. Leia este artigo se você quiser aprender regressão logística. **Revista de Sociologia e Política**, v. 28, n. 74, e006, 2020.

FREITAS, M. **Fundamentos do gerenciamento de serviços de TI:** preparatório para certificação ITIL Foundation. 2. ed. Rio de Janeiro: Brasport, 2013.

GALARNYK, Michael. **Visualizing Decision Trees with Python (Scikit-learn, Graphviz, Matplotlib)**, 2020. Disponível em: https://www.kdnuggets.com/2020/04/visualizing-decision-trees-python.html Acesso em: 24 abr. 2023.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 6. ed. São Paulo: Atlas, 2017.

GOOGLE CLOUD. **O que é inteligência artificial (IA)?**, 2023. Disponível em: https://cloud.google.com/learn/what-is-artificial-intelligence?hl=pt-br/ Acesso em: 20 abr. 2023.

GRUS, Joel. **Data Science do Zero.** Tradução Welington Nascimento. 1 ed. Rio de Janeiro: Alta Books, 2016.

HAN, Jiawei; KAMBER, Micheline; PEI, Jian. **Data mining:** concepts and techniques. 3 ed. Waltham - Estados Unidos da América: Elsevier, 2012.

HARDWARE. **Qual é a diferença entre Machine Learning e Deep Learning?**, 2022. Disponível em: https://www.hardware.com.br/artigos/qual-e-a-diferenca-entre-machine-learning-e-deep-learning/ Acesso em: 27 mar. 2023.

HURWITZ, Judith; KIRSCH, Daniel. **Machine Learning For Dummies®, IBM Limited Edition**. Hoboken: John Wiley & Sons, 2018.

KAUARK, Fabiana da Silva; MANHÃES, Fernanda Castro; MEDEIROS, Carlos Henrique. **Metodologia da pesquisa**: um guia prático. Itabuna: Via Litterarum, 2010.

KUNUMI. **Métricas de Avaliação em Machine Learning:** Classificação, 2020. Disponível em: https://medium.com/kunumi/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-em-machine-learning-classifica%C3%A7%C3%A3o-49340dcdb198/. Acesso em: 27 mar. 2023.

KURZWEIL, R. **The Age of Intelligent Machines**. MIT Press. 1990.

LANTZ, Brett. **Machine Learning with R:** Learn How to Use R to Apply Powerful Machine Learning Methods and Gain an Insight into Real-World Applications. Birmingham, UK: Packt Publishing, 2013.

MIGUEL, Tussevana. **Máquinas de Vetores de Suporte (SVM)**, 2023. Disponível em: https://aprenderdatascience.com/maquina-de-vetores-de-suporte-svm/ Acesso em: 24 abr. 2023.

PACHECO, André. **K vizinhos mais próximos – KNN**, 2017. Disponível em: http://computacaointeligente.com.br/algoritmos/k-vizinhos-mais-proximos/ Acesso em: 31 mar. 2023.

PROVOST, Foster; FAWCETT, Tom. **Data Science para negócios.** Rio de Janeiro: Alta Books, 2016.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência Artificial**. Tradução Regina Célia Simille. Rio de Janeiro: Elsevier, 2013.

SAKURAI, Rafael. **Decision Tree:** Aprendendo a classificar flores do tipo Iris, 2023. Disponível em: https://www.sakurai.dev.br/classificacao-iris/ Acesso em: 23 abr. 2023.

SCIKIT-LEARN. **Nearest Neighbors Classification**, 2023. Disponível em: https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html. Acesso em: 22 abr. 2023.

SCIKIT-LEARN. **Support Vector Machines**, 2023. Disponível em: https://scikit-learn.org/stable/modules/svm.html. Acesso em: 16 mar. 2023.

Apêndice 1

Código Python para utilização dos algoritmos de Machine Learning

```
#!/usr/bin/env python
# coding: utf-8

# # Importação de pacotes

# In[1]:


#Manipulação e tratamento dos dados
import pandas as pd
import numpy as np
import datetime as dt
import time as tm
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib

#Remover limitação de exibição do DataFrame
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

#Remover aviso de SettingWithCopyWarning
pd.set_option('mode.chained_assignment', None)

#Importando as bibliotecas do Scikit Learn
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler #Utilizada para fazer a
padronização dos dados
from sklearn.preprocessing import LabelEncoder #tratamento de string para
número
from sklearn.model_selection import train_test_split #Utilizada para
separar dados para treino e teste
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier

from sklearn.metrics import accuracy_score #Utilizada para avaliar a
acurácia do modelo preditivo
from imblearn import under_sampling, over_sampling #Utilizada para fazer o
balanceamento de dados
```

```
from imblearn.over_sampling import SMOTE #Utilizada para fazer a
banlanceamento de dados


# # Coleta de Dados
# Serão importados os dados do arquivo .csv disponibilizado pelo Oi

# In[2]:


#Importando a base de dados
df_base_os = pd.read_csv('base_OS_ctrlAc_24out17_27fev23.csv', sep=";",
encoding='raw_unicode_escape')


# # Pré-Processamento

# ## Análise da base

# In[3]:


#Visualizar qtde de linhas e colunas
df_base_os.shape


# In[4]:


#Visualizar as primeiras linhas do dataframe
df_base_os.head()


# In[5]:


#Verificar os tipos de dados das variáveis
df_base_os.info()


# ## Eliminando linhas com valores nulos

# In[6]:


#Verificando a quantidade de nulos nas variáveis existentes
df_base_os.isnull().sum()


# In[7]:


#Filtrando a Coluna Fechamento somente onde há valores nulos
df_base_os.loc[df_base_os['Fechamento'].isnull()]


# In[8]:


#Verificando a quantidade de dados nulos em Fechamento
df_base_os['Fechamento'].isnull().sum()
```

```
# In[9]:


#Excluindo da análise os dados nulos em Fechamento
df_base_os = df_base_os.dropna(subset='Fechamento')


# In[10]:


#Verificando novamente a quantidade de dados nulos em Fechamento
df_base_os['Fechamento'].isnull().sum()


# In[11]:


#Identificando valores nulos em Equipamentos
df_base_os['Equipamento'].isnull().sum()


# In[12]:


# Excluir linhas com campos vazios na coluna Equipamentos
df_base_os = df_base_os.dropna(subset='Equipamento')


# In[13]:


#Identificando novamente valores nulos em Equipamentos
df_base_os['Equipamento'].isnull().sum()


# In[14]:


#Verificando a quantidade de dados nulos em Horas
df_base_os.loc[df_base_os['Horas'].isnull()]


# ## Preenchendo valores nulos

# In[15]:


#Preenchendo valores nulos com a moda em EmpEnv
df_base_os["EmpEnv"]=df_base_os["EmpEnv"].fillna(df_base_os["EmpEnv"].mode(
)[0])


# In[16]:


df_base_os["EmpEnv"].isnull().sum()


# In[17]:
```

```
#Verificando novamente a quantidade de nulos nas variáveis existentes
df_base_os.isnull().sum()


# ## Eliminando colunas desnecessárias para o modelo

# In[18]:


#Verificando Valores únicos
df_base_os.nunique()


# In[19]:


df_base_os.drop(['SLAFim','Horas','SetorCGS','Contratada','Tipo','LoginCada
stro','LoginAbertura','LoginFechamento'], axis = 1, inplace = True)


# In[20]:


df_base_os.shape


# In[21]:


df_base_os.info()


# ## Alterando o tipo de dados envolvendo data/hora

# In[22]:


#Consultando o valor da primeira linha da coluna Abertura
df_base_os['Abertura'][0]


# In[23]:


df_base_os.head()


# In[24]:


# Separando data/hora de abertura e de fechamento de OS
df_base_os['Abertura'] = pd.to_datetime(df_base_os['Abertura'])
df_base_os['Data_Abertura'] = df_base_os['Abertura'].dt.strftime('%d-%m-
%Y')
df_base_os['Hora_Abertura'] =
df_base_os['Abertura'].dt.strftime('%H:%M:%S')
df_base_os['Fechamento'] = pd.to_datetime(df_base_os['Fechamento'])
df_base_os['Data_Fechamento'] = df_base_os['Fechamento'].dt.strftime('%d-
%m-%Y')
```

```
df_base_os['Hora_Fechamento'] =
df_base_os['Fechamento'].dt.strftime('%H:%M:%S')


# In[25]:


df_base_os.info()


# In[26]:


df_base_os['Data_Abertura'] = pd.to_datetime(df_base_os['Data_Abertura'],
dayfirst=True)
df_base_os['Data_Fechamento'] =
pd.to_datetime(df_base_os['Data_Fechamento'], dayfirst=True)


# In[27]:


df_base_os.info()


# ## Criação da coluna Tempo de Conclusão tipo inteiro

# In[28]:


# Criando a coluna Tempo_Conclusão em dias
df_base_os['Tempo_Conclusão'] = (df_base_os['Data_Fechamento'] -
df_base_os['Data_Abertura'])


# In[29]:


df_base_os.info()


# In[30]:


#Convertendo Tempo_Conclusão em inteiro >>> nova coluna Tempo_Conclusão_dia
e ajustando valores negativos
df_base_os['Tempo_Conclusão_dia'] =
abs(df_base_os['Tempo_Conclusão'].dt.days.astype('int'))


# In[31]:


(df_base_os['Tempo_Conclusão_dia']<0) is True


# ## Criação da variável alvo

# In[32]:
```

```
#Criando a variável ALVO para predição
prazo = 5
df_base_os['DentroPrazo'] = np.where(df_base_os['Tempo_Conclusão_dia'] <=
prazo, 'Dentro do Prazo', 'Fora do Prazo')
# 1 = dentro do prazo / 0 = fora do prazo


# In[33]:


df_base_os["DentroPrazo"].value_counts()
graf = sns.countplot(x=df_base_os["DentroPrazo"])


# In[34]:


df_base_os["DentroPrazo"].value_counts()


# ## Preparação do dataframe para criação do modelo

# In[35]:


# Criando o encoder e aplicando LabelEncoder - transformar variáveis
categóricas em numéricas
lb = LabelEncoder()

for var in
['EmpEnv','Equipamento','TipoProblema','ResumoRelato','StatusSimples',
            'Estação','TipoPredio','UF']:
    df_base_os[var+'_lab'] = lb.fit_transform(df_base_os[var])


# In[36]:


df_base_os.shape


# In[37]:


df_base_os.head()


# In[38]:


df_base_os.info()


# # Análise Exploratória
# Analisando as variáveis categóricas e numéricas.

# In[39]:


#Visualizar qtde de linhas e colunas
df_base_os.shape
```

```
# ## Analisando as variáveis categóricas

# In[40]:


#Analisando a quantidade de repetições na variável ResumoRelato
df_base_os['ResumoRelato'].value_counts()


# In[41]:


df_base_os['ResumoRelato'].value_counts(normalize = True)*100


# In[42]:


import plotly.express as px


# In[43]:


#Analisando a quantidade de repetições na variável TipoProblema
df_base_os['TipoProblema'].value_counts()


# In[44]:


df_base_os['TipoProblema'].value_counts(normalize = True) * 100


# In[45]:


df_base_os.groupby(['EmpEnv']).size()


# In[46]:


graf = sns.countplot(x=df_base_os["EmpEnv"])


# In[47]:


df_base_os['EmpEnv'].value_counts(normalize = True)*100


# In[48]:


df_base_os.groupby('StatusSimples').size()


# In[49]:
```

```
df_base_os["DentroPrazo"].value_counts()
graf = sns.countplot(x=df_base_os["DentroPrazo"])


# In[50]:


df_base_os["DentroPrazo"].value_counts()


# In[51]:


df_base_os['DentroPrazo'].value_counts(normalize = True)*100


# In[52]:


df_base_os.select_dtypes(include='object').describe()


# In[53]:


graf = sns.countplot(x=df_base_os["TipoProblema_lab"])


# In[54]:


df_base_os.groupby(['TipoProblema']).size()


# # Criação dos modelos preditivos

# ## Separação dos dados de treino e teste

# In[55]:


# Separação dos dados de treino e de teste
# X são as variáveis de entrada do algoritmo
# Y é a variável alvo
X =
df_base_os[['EmpEnv_lab','Equipamento_lab','TipoProblema_lab','ResumoRelato
_lab','StatusSimples_lab',
            'Estação_lab','TipoPredio_lab','UF_lab']]
Y = df_base_os['DentroPrazo'].values.tolist()

#Separação de dados de teste e treino (30% para teste)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3,
random_state=42)


# ## Máquina Preditiva

# #### SVC
```

```
# In[56]:


#Criando o classificador com algoritmo a ser avaliado
# SVC
clf_svc = SVC()


# In[57]:


#Treinando o classificador usando dados de treino
clf_svc = clf_svc.fit(X_train,Y_train)


# #### LogisticRegression

# In[58]:


#Criando o classificador
clf_log_reg = LogisticRegression(max_iter=400)


# Tive que colocar máximo de iterações para o algoritmo convergir

# In[59]:


#Treinando o classificador usando dados de treino
clf_log_reg = clf_log_reg.fit(X_train,Y_train)


# #### Árvore de Decisão

# In[60]:


#Criando o classificador
clf_dec_tree = DecisionTreeClassifier()


# In[61]:


#Treinando o classificador usando dados de treino
clf_dec_tree = clf_dec_tree.fit(X_train,Y_train)


# #### KNN

# In[62]:


#Criando o classificador
clf_knn = KNeighborsClassifier()


# In[63]:
```

```
#Treinando o classificador usando dados de treino
clf_knn = clf_knn.fit(X_train,Y_train)


# #### Random Forest

# In[64]:


#Criando o classificador
clf_rf = RandomForestClassifier()


# In[65]:


#Treinando o classificador usando dados de treino
clf_rf = clf_rf.fit(X_train,Y_train)


# ## Avaliação da Máquina Preditiva

# #### SVC

# In[66]:


# Validando o classificador
accuracy = clf_svc.score(X_test, Y_test)
print('Accuracy: ' + str(accuracy))

# Criando a matriz de confusão
prediction = clf_svc.predict(X_test)

cm = confusion_matrix(Y_test,prediction)
cr = classification_report(Y_test,prediction)
print(cm)
print(cr)


# In[67]:


matrix = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Dentro do Prazo','Fora do Prazo'])

matrix.plot()
```

	Acurácia	F1
SVC	57%	43%

```
# #### Logistic Regression

# In[68]:


# Validando o classificador
accuracy = clf_log_reg.score(X_test, Y_test)
print('Accuracy: ' + str(accuracy))

# Criando a matriz de confusão
prediction = clf_log_reg.predict(X_test)
```

```
cm = confusion_matrix(Y_test, prediction)
cr = classification_report(Y_test, prediction)
print(cm)
print(cr)


# In[69]:


matrix = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Dentro do Prazo','Fora do Prazo'])

matrix.plot()


# #### Árvore de Decisão

# In[70]:


# Validando o classificador
accuracy = clf_dec_tree.score(X_test, Y_test)
print('Accuracy: ' + str(accuracy))

# Criando a matriz de confusão
prediction = clf_dec_tree.predict(X_test)

cm = confusion_matrix(Y_test,prediction)
cr = classification_report(Y_test,prediction)
print(cm)
print(cr)


# In[71]:


matrix = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Dentro do Prazo','Fora do Prazo'])

matrix.plot()


# #### KNN

# In[72]:


# Validando o classificador
accuracy = clf_knn.score(X_test, Y_test)
print('Accuracy: ' + str(accuracy))

# Criando a matriz de confusão
prediction = clf_knn.predict(X_test)

cm = confusion_matrix(Y_test,prediction)
cr = classification_report(Y_test,prediction)
print(cm)
print(cr)
```

```
# In[73]:


matrix = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Dentro do Prazo','Fora do Prazo'])

matrix.plot()


# #### Random Forest

# In[74]:


# Validando o classificador
accuracy = clf_rf.score(X_test, Y_test)
print('Accuracy: ' + str(accuracy))

# Criando a matriz de confusão
prediction = clf_rf.predict(X_test)

cm = confusion_matrix(Y_test,prediction)
cr = classification_report(Y_test,prediction)
print(cm)
print(cr)


# In[75]:


matrix = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['Dentro do Prazo','Fora do Prazo'])

matrix.plot()


# ### Resultado
                          Acurácia         F1
   SVC                       57%          43%
   Logistic Regression       64%          54%
   Decision Tree             63%          62%
   KNeighbors                61%          59%
   Random Forest             65%          64%
# Problema a ser resolvido: Identificar os padrões associadas aos
fechamentos de OS fora do prazo de 5 dias.

# In[ ]:
```