UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM ENGENHERIA DE SOFTWARE E INOVAÇÃO

LEONARDO OLIVEIRA MELLO

Ciência de Dados Aplicada a Gestão de Projetos de Quality Assurance

Trabalho apresentado como requisito parcial para obtenção do grau de Especialista em Engenharia de Software e Inovação.

Orientador: Prof. Dr. Flávio Moreira de Oliveira

Porto Alegre

2021

RESUMO

Em Projetos de *Software* faz parte do processo de desenvolvimento, encontrar falhas durante a construção do software, e o quanto antes os mesmos forem encontrados, menor será o custo de correção e impacto no negócio. Essas falhas, dependendo do período em que são encontradas, podem acarretar perda financeira e de credibilidade para o cliente. Com um mercado cada vez mais volátil e competitivo, o quanto antes o time do projeto conseguir identificar a causa raiz dessas falhas, decisões estratégicas e até mesmo de mitigação dos riscos de ocorrer as falhas poderão ser tomadas, reduzindo o desperdício e retrabalho do time de projetos. Dentro desse contexto, foram realizadas as análises de um conjunto de dados contendo as falhas de projetos de desenvolvimento e utilizando Ciência de Dados e Aprendizado de Máquina, foi possível prever a causa raiz das falhas, levando em consideração as demais informações presentes no conjunto de dados. Como resultado do estudo, temos a confirmação da viabilidade de utilizar Ciência de Dados e Aprendizado de Máquina para auxiliar na tomada de decisão na gestão de projetos de desenvolvimento de *Software*.

Palavras-chave: Ciência de Dados. Aprendizado de Máquina. Gestão. Quality Assurance.

Data Science Applied to Quality Assurance Project Management

ABSTRACT

Software Projects, it is part of the development process, finding faults during the construction of the software, and the sooner they are found, the lower the cost of correction and affect the business will be. These failures, depending on the period in which they are found can lead to financial and credibility loss for the client. With an increasingly volatile and competitive market, the sooner the project team can identify the root cause of these failures, strategic decisions and even mitigation of the risks of failures can be taken, reducing waste and rework of the projects. Within this context, the analyzes of a dataset containing the failures of development projects were performed and using Data Science and Machine Learning techniques, it was possible to predict the root cause of the failures, taking into account the other information present in the dataset . Because of the study, we have confirmation of the feasibility of using Data Science and Machine Learning to assist in decision making in managing software development projects.

Palavras-chave: Data Science. Machine Learning. Management. Quality Assurance.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A tecnologia da informação desempenha uma função chave no dia a dia das empresas. Se é preciso expandir os negócios, oferece o insumo fundamental para planejar, executar e controlar as ações; se é necessário racionalizar os custos, tem as ferramentas vitais para superar as adversidades, em busca de uma gestão mais eficiente. O processo é global e dele o Brasil participa com dinamismo e inventividade. A indústria de TI (Tecnologia da Informação) tem mais de cinco décadas de crescimento e bons resultados; a TI, somada a comunicações, representa cerca de 6,8% do PIB brasileiro (BRASSCOM, 2021).

Com esse cenário, fica evidente que empresas que não se preocuparem com a qualidade e formas de realizar análises que possam ajudar de forma antecipada a tomada de decisões, terão dificuldades devido à velocidade de mudanças que as empresas hoje enfrentam e a necessidade constante de produzir projetos que realmente agreguem valor ao negócio.

A *DBServer* atua desde 1993 auxiliando empresas a realizar transformações digitais, otimização de legado e melhorias de processo de trabalho. Em um cenário de avanços tecnológicos rápidos e exponenciais, dispõe de serviços que auxiliam seus clientes a promover a adoção de novas tecnologias, melhoria contínua dos sistemas existentes, bem como implementação de processos de *Quality Assurance* (Garantia de Qualidade).

Em relação ao Serviço de *Quality Assurance*, alguns dos principais indicadores gerados no processo de trabalho são os problemas encontrados durante o processo de execução de testes de **integração**, de **sistema** e de **aceitação** dos usuários-chave. Esses dados geralmente são analisados sob a ótica de explicar o que ocorreu e não com o viés de base de conhecimento para decisões futuras.

Dentro desse contexto, o objetivo desse trabalho é aplicar conceitos e técnicas de *Data Science* (Ciência de Dados*)* e *Machine Learning* (Aprendizado de Máquina) em uma base de dados histórica de problemas encontrados em projetos de *software* da empresa. Acreditamos que o estudo trará informações relevantes e que auxiliarão uma análise preditiva para os próximos projetos de *Quality Assurance* que forem conduzidos, auxiliando no planejamento e estratégia de testes que deverá ser implantada.

Para o pesquisador, esse trabalho trará uma oportunidade de desenvolver e aplicar o conhecimento em *Data Science* e *Machine Learning*, analisando dados oriundos de projetos de *Quality Assurance*. Para a organização, será importante a aplicação desse trabalho, visto que, se as hipóteses e aplicação de algum modelo ao longo do trabalho produzir resultados palpáveis, esse modelo poderá ser adaptado para outros clientes ou projetos da empresa.

Para um melhor entendimento desse trabalho, a seção 1 Fundamentação Teórica, descreve os conceitos de Qualidade de *Software*, definições de erro, defeito, falha e níveis de testes, além de explicar brevemente sobre *Data Science* e *Machine Learning.* Já a seção 3 Objetivos, descreve o objetivo principal do trabalho proposto e as metas a serem alcançadas. A seção 4 Metodologia mostrará a forma como os dados foram coletados e analisados. Grifa-se a seção 5 Apresentação e Análise de Dados, onde serão apresentados como foi a coleta, pré-processamento, transformação, geração dos modelos e interpretação e validação dos resultados. Por fim a seção 6 Conclusão, trará a visão do autor em relação ao trabalho realizado e oportunidades futuras de trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo foi apresentada a evolução da qualidade de *software*, uma explicação do que é teste de *software* e uma introdução sobre ciência de dados aplicada em projetos de TI.

2.1 QUALIDADE DE SOFTWARE

A qualidade de *software* é uma área de conhecimento da Engenharia de *software*, que objetiva afiançar a conformidade do *software* através da descrição e normatização de processos de desenvolvimento. Embora os modelos aplicados na garantia da qualidade de *software* atuem principalmente no processo, o principal alvo é garantir um produto final que satisfaça às expectativas do cliente dentro daquilo que foi contratado inicialmente. (KOSCIANSKI, 2007).

Segundo a norma ISO 9000 (versão 2005), a qualidade é o grau em que um conjunto de características inerentes a um produto, processo ou sistema cumpre os requisitos inicialmente estipulados para estes.

Um dos erros mais comuns sobre qualidade é o modo como este é inserido dentro do processo de engenharia de *software*. É usual pensar o desenvolvimento de *software* como uma linha do tempo. Dessa forma, definem-se metodologias e processos de trabalho para produzir *software* e garantir que as etapas serão cumpridas. O erro é acreditar que dentro desse processo existe um período alocado especificamente para a realização dos testes; qualidade não é uma fase do ciclo de desenvolvimento de *software*, é parte de todas as fases (BARTIÉ, 2002).

Dessa forma pode-se verificar que a qualidade está relacionada a processos de condução, controles e melhorias das atividades, buscando sempre a melhoria da *performance* produtiva, não somente pela área de desenvolvimento, mas da organização como um todo.

A partir da década de 1980, observa-se uma mudança na forma de gerenciar os negócios e nas relações entre empresas e clientes. A produtividade, que era a palavra de ordem na primeira metade do século, abriu espaço para a qualidade. O importante não é produzir muito, mas sim produzir produtos com qualidade. E qualidade é avaliada pelo cliente, ou seja, é preciso produzir produtos que satisfaçam ao cliente.

2.2 TESTE DE SOFTWARE

Nos primórdios do desenvolvimento de *software*, a ação de testar era encarada como uma atividade de navegação pelo código e corrigir problemas já conhecidos. Essas ações eram realizadas pelos próprios desenvolvedores, não existindo profissionais dedicados a essa atividade (BASTOS et al, 2007).

No final da década de 50, o conceito do teste de *software* ampliou seus valores, tornando-se um processo de detecção de erros de *software*, mas com os testes sendo realizados no final do processo de desenvolvimento (BASTOS et al, 2007).

No início da década de 1970, o processo de desenvolvimento de *software* passou a ter uma abordagem mais profunda com a engenharia de *software* sendo adotada como modelo para as universidades e organizações, porém havia pouco consenso sobre o que viria a ser teste. Somente em 1972 é que haveria a primeira conferência formal sobre testes na Universidade da Carolina do Norte (BARTIÉ, 2002).

Foi Myers (1979 apud BARTIÉ, 2002), quem produziu um dos primeiros trabalhos mais completos sobre teste. Nesse trabalho, Myers já definia testes como um "processo de trabalho com a intenção de encontrar erros". Sua premissa era que, se o objetivo do teste fosse apenas provar a boa funcionalidade de um aplicativo, seriam encontrados poucos defeitos, uma vez que toda a energia do processo de teste seria direcionada apenas na comprovação do fato.

No início da década de 1980 surgiram os primeiros conceitos de qualidade de *software*. Nessa abordagem, desenvolvedores e testadores trabalham juntos desde o início do processo de desenvolvimento (BARTIÉ, 2002).

A definição do que é teste de *software* pode ser sintetizada através de uma visão intuitiva ou mesmo de maneira formal. Existem várias definições para esse conceito, mas de uma forma simples, testar um *software* significa verificar, através de uma execução controlada, a presença de discrepâncias entre o comportamento implementado e o especificado. O principal objetivo desta tarefa é encontrar o máximo de problemas com o mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos (BASTOS et al, 2007).

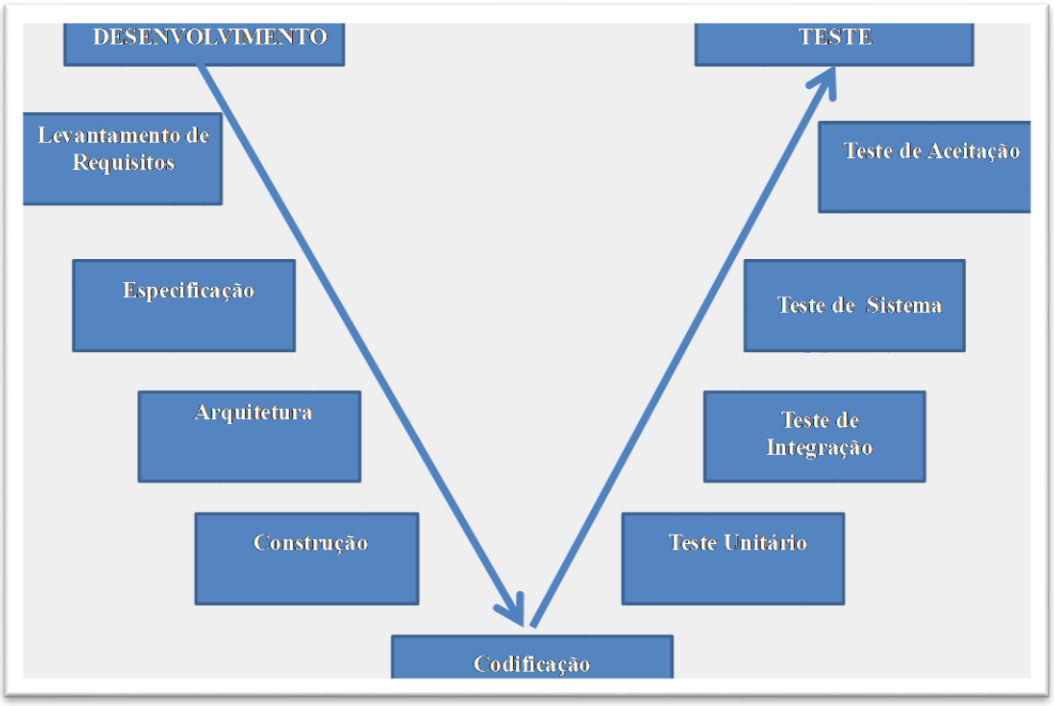
As atividades do teste devem estar inseridas dentro do processo de desenvolvimento do *software* desde o início de sua concepção e não somente após o produto de *software* estar implementado. Atualmente, existem diversos processos diferentes de desenvolvimento de *software* e nem sempre a cultura do teste está inserida nestes processos como deveriam estar. Visto isto, o processo de teste é criado paralelamente e deve acompanhar o ciclo de desenvolvimento, independente de qual seja (BASTOS et al, 2007).

Para que cada atividade no processo de teste de *software* seja corretamente realizada, os seguintes papéis da equipe são definidos: Auditor, Usuário-chave, Analista de Teste e Testador. O Auditor tem como principal papel no projeto e na organização realizar o controle e garantia da qualidade tanto do processo quanto do produto de *software* em desenvolvimento. É de sua responsabilidade definir processos e práticas a serem atribuídas a todos do time de teste de

forma que o projeto tenha a qualidade esperada pelo cliente. O Analista de Teste tem dentro de seu escopo de trabalho o entendimento do negócio do cliente para definição dos casos de testes. O Testador é responsável por executar os testes através dos casos de testes e relatar os defeitos encontrados para o time de desenvolvimento (BASTOS et al, 2007).

O principal modelo de interação do teste com o desenvolvimento, e mais difundido no mercado é o modelo-V. Este modelo define um conjunto de atividades de verificação e validação em cada etapa do ciclo de vida do projeto de forma que os defeitos sejam revelados o mais cedo possível, reduzindo custo de correção e retrabalho segundo Myers, (1979 apud BASTOS et al, 2007) conforme figura 1.

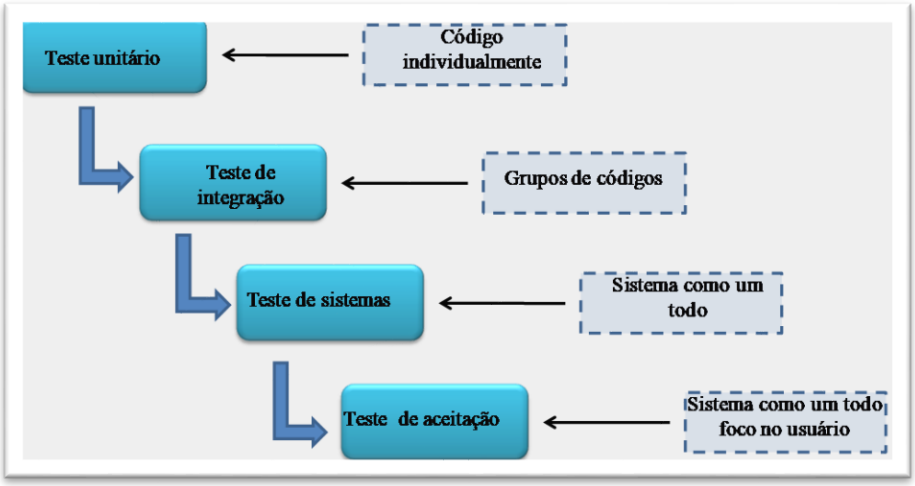
Figura 1 - O modelo-V.



Fonte: BASTOS et al (2007)

Os testes são divididos em quatro níveis: unitário, integração, sistema e aceitação. Cada nível representa um foco de abrangência. O primeiro nível é o teste unitário que se caracteriza por revelar defeitos no código. O teste de integração tem como objetivo avaliar se um ou mais componentes funcionam de forma interligada. O teste de sistema avalia funcionamento do *software* como um todo. Já o teste de aceitação avalia se as regras de negócios solicitadas pelos usuários chave foram desenvolvidas de forma correta em um ambiente pré-produtivo ou similar (BASTOS, et al, 2007).

Figura 2 - Níveis de testes.



Fonte: BASTOS et al. (2007)

A palavra "*Bug"* geralmente é utilizada para representar um erro, falha ou defeito, sendo que cada uma dessas palavras possuem um significado distinto. Segundo o padrão IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos):

- **Erro:** É a diferença entre um valor ou condição computada e o valor ou condição verdadeira especificada, produzido por um erro humano,
- **Defeito**: É uma imperfeição ou deficiência em um produto de trabalho que não atende a seus requisitos ou especificações e precisa ser consertado ou substituído,
- **Falha**: É um evento no qual um sistema ou componente do sistema não executa uma função necessária dentro dos limites especificados.

Para fim desse trabalho, vamos adotar a nomenclatura da IEEE, e usar o termo "falha" (IEEE Std 1044-2019).

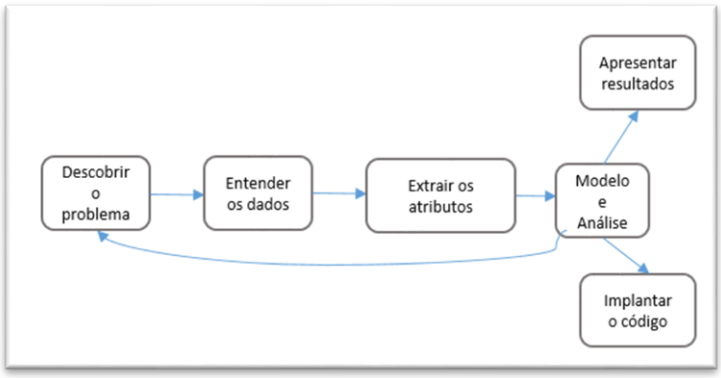
2.3 DATA SCIENCE

Data Science tem como objetivo trabalhar de forma analítica os dados, produzindo informações que servirão para um embasamento preditivo. Segundo Taurion (TAURION, 2013), as empresas não deveriam tomar decisões baseadas em palpites, pois com o aumento de volume de dados, os mesmos deveriam ser tratados e analisados para as decisões serem baseadas em dados concretos. *Data Science*, por si só, é uma grande disciplina que consolida um conjunto de habilidades especializadas, tais como: matemática, estatística, conhecimento

do negócio, programação e computação, além de teorias e técnicas como análise preditiva, *Machine Learning*, modelagem, engenharia e mineração de dados (TAURION, 2013).

Para a resolução de um problema de ciência de dados, existe uma sequência sugerida para a obtenção de resultados satisfatórios, descobrir o problema, entender os dados, extrair as funcionalidades, utilizar / criar modelos e análise, implantar o código e apresentar os resultados, conforme a imagem a seguir (CADY, 2017).

Figura 3: *Data Science Roadmap*



Fonte: Adaptado de Cady (2017)

2.3.1 Descobrir o Problema

O primeiro passo é descobrir o problema a ser analisado. Em projetos de desenvolvimento de software, existem alguns documentos que são utilizados para esclarecer o escopo a ser desenvolvido, tais como Termo de Abertura de Projeto, Especificação Funcional, *User Story*, *Story Mapping*, entre outros, que possuem o intuito de reduzir entendimentos equivocados e definir da forma mais clara possível o que precisa ser feito. Portanto, antes de investir esforço no desenvolvimento, é fundamental ter certeza que estão trabalhando no problema correto. Quando trazemos essa definição de construção de *software* para *Data Science*, onde os clientes serão máquinas, o problema geralmente é bem mais claro, mas pode haver ambiguidade sobre restrições de *software* (linguagem a utilizar, tempo de execução, previsões que serão realizadas, etc). DS resolve problemas de negócio, por isso é importante entender quais são as dores do cliente, compreender que tipo de questionamento ele possui, quais decisões que ele toma sem possuir embasamento de dados e principalmente o quanto o resultado de um projeto de DS poderia ajudá-lo na tomada de decisões, portanto, antes de mergulhar no projeto de DS é importante entender o que e como será construído para resolver o problema e qual será o critério para considerar o projeto de DS como pronto (CADY, 2017).

2.3.2 Entender os Dados

Depois de ter acesso aos dados, é importante realizar uma série de questionamentos padrões, buscando certificar que a base de dados é coerente e uma análise poderá ser feita de maneira adequada. É uma forma de identificar possíveis problemas de dados o mais cedo possível, evitando predições incorretas. Alguns questionamentos que precisam ser respondidos:

- Qual o tamanho do conjunto de dados?
- Este é o conjunto de dados inteiro?
- Esses dados são representativos o suficiente?
- Existem campos que são identificadores exclusivos?

A pergunta mais importante que precisa ser feita em relação aos dados disponíveis é se eles podem resolver o problema de negócio a ser resolvido. Caso não seja possível é importante procurar outras fontes de dados ou mudar o projeto a ser realizado (CADY, 2017).

De posse dos dados, é necessário organizá-los, tanto em atividades relacionadas a como serão obtidos esses dados, limpeza, filtros e equalização das informações. Depois de ter os dados entendidos é necessário realizar uma análise exploratória; basicamente é vasculhar os dados, visualizar de maneiras diferentes, tentando interpretá-los por diferentes perspectivas. A análise exploratória é uma das etapas mais criativas de DS, momento de calcular algumas correlações e similaridades (CADY, 2017).

2.3.3. Extrair os atributos

Essa etapa tem sobreposição com a análise exploratória e a preparação de dados. Um atributo é na verdade um número ou categoria extraída dos dados analisados. A maioria dos atributos que extraímos será usada para prever algo, contudo é necessário extrair o que se está prevendo, que também é chamada de variável de destino. Em termos práticos, extrair atributos significa pegar o conjunto de dados bruto, classificar em linhas e colunas, onde cada linha corresponde a alguma entidade do mundo real e cada coluna dá uma informação que descreve essa entidade. Extrair bons recursos é o mais importante para obter uma análise satisfatória (CADY, 2017).

2.3.4 Modelo e Análise

Uma vez que os atributos foram extraídos, a maioria dos projetos de DS envolve a utilização de algum tipo de modelo de *machine learning* – por exemplo, algum classificador que identifique se um cliente continua fiel, um modelo de regressão que prevê o preço de um aluguel ou um algoritmo para dividir os clientes em segmentos. Geralmente é a etapa mais

simples, onde o profissional de DS conecta os dados em alguns modelos para avaliar qual que obterá os melhores resultados (CADY, 2017).

2.3.5 Apresentar os Resultados

Essa é a etapa de descrever o trabalho realizado e os resultados obtido. Pelo fato do trabalho de DS ser altamente técnico é importante trazer uma linguagem mais lúdica e com correlações que facilitem o entendimento de um público variado e multidisciplinar.

2.3.6 Implementar o Código

É importante entender qual será a finalidade desse código, se fará parte de um outro *software*, como por exemplo um módulo analítico, escrito em linguagem de programação de alto desempenho e aderindo as melhores práticas de engenharia de *software*, produzir relatórios legíveis por humanos ou treinará modelo estatístico referenciado por outro código. A partir do entendimento da finalidade do código se define o detalhamento da documentação do código criado, como serão feitos os testes do código para garantir que ele tenha qualidade e obviamente a entrega do próprio código em algum repositório (CADY, 2017).

2.4 MACHINE LEARNING

Machine Learning consiste em dar aos computadores a habilidade de aprender a partir de informações repassadas, sem necessariamente ter sido programado para realizar uma determinada ação. Isso é possível através da disponibilidade de dados e fazendo com que haja um aprendizado dessas informações para a criação de modelos de decisões que dessa forma serão utilizados como ferramenta de predição; ou seja, a aprendizagem de máquina tem como objetivo aprender através da experiência (GERÓN, 2019).

Dessa forma, podemos dividir *Machine Learning* em supervisionado, que busca responder um *target*, ou seja, há uma variável explícita cujo valor busca-se predizer, e não supervisionado, em que se busca identificar grupos ou padrões a partir dos dados.

2.4.1 Aprendizagem Supervisionada

As técnicas de Machine Learning (ML) supervisionadas podem ser divididas em Classificação, que buscam exemplificar uma variável categórica, que possuam duas categorias ou mais, ou Regressão, que busca encontrar a evolução de uma variável, comparada as outras.

2.4.1.1 Classificação Supervisionada

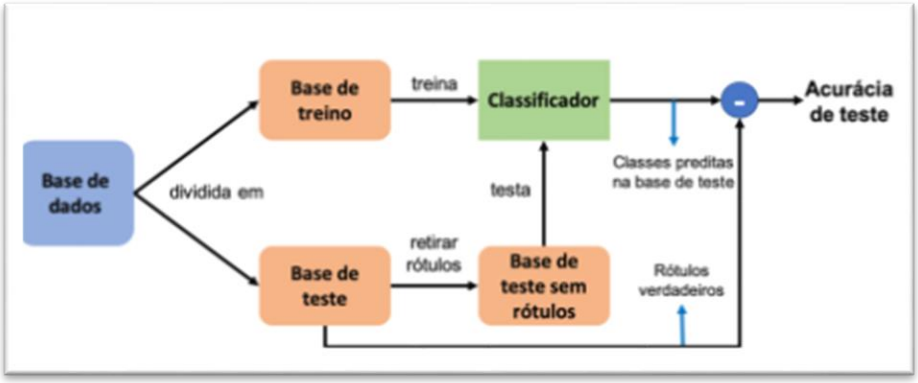
Em um problema de Classificação, é apresentado para o algoritmo um conjunto de atributos contendo as entradas e saídas corretas. O objetivo do algoritmo é aprender uma regra geral que mapeie as entradas nas saídas corretamente. Os dados de entrada são divididos em dois grupos: X, com os atributos a serem utilizados para determinar os dados de saída (atributos de predição) e Y, com o atributo de cujo valor desejamos fazer a predição do (atributo alvo ou target). Esses dados geram dois subconjuntos disjuntos (a base de treino e a base de teste), onde:

- A base de treino é submetida ao algoritmo para treinar o modelo, que pode ser calibrado e ajustado com base nos dados apresentado,
- E os dados são rodados na base de testes que como saída deverá realizar a predição.

Comparando os dados apresentados com a base original é possível medir a acurácia desses dados, que é uma das métricas mais utilizadas. Existem também outras métricas que podem ser utilizadas, uma delas é a Precisão, utilizando como exemplo um problema de classificação que determina se um vídeo é adequado ou não para crianças. O objetivo é que o resultado desse algoritmo tenha alta precisão, ou seja, é importante reduzir ao máximo o número de falsos positivos, evitando que o algoritmo diga que um vídeo é para criança e na verdade não seja, é preferível que ele erre e diga que não é para criança embora seja, do que que diga que é para criança e o conteúdo ser impróprio. Outra métrica é a Revocação, utilizando como exemplo um problema de classificação que analisa se uma pessoa tem ou não uma doença grave. O objetivo é que o resultado desse algoritmo tenha o menor número possível de falsos negativos, evitando que o algoritmo diga que uma pessoa não tem a doença e na verdade ela tem. É preferível que o algoritmo erre e diga que a pessoa tem, e na contraprova se identifique o erro, do que não identificar que a pessoa tem e isso acarretar em um agravamento do quadro clinico ou até mesmo risco de morte.

Em problemas de classificação, o atributo alvo é sempre categórico KOSHIYAMA, ESCOVEDO (2020).

Figura 4: Resumo de um problema de classificação



Fonte: KOSHIYAMA, ESCOVEDO (2020)

2.4.1.1.1 Decision Trees (Árvores de Decisão)

A Árvore de Decisão é um dos modelos preditivos mais utilizados e que apresenta o resultado de uma forma visual que facilita o entendimento por um ser humano. Basicamente é um algoritmo supervisionado que se baseia na ideia de divisão dos dados em grupos homogêneos para a melhor classificação ou regressão. Um exemplo é a figura 5 que representa um *dataset* com os atributos que serão levados em conta para predizer se uma pessoa irá ou não sair para jogar golfe. No exemplo, com base nas colunas *Outlook* (panorama), *Temp* (temperatura), *Humidity* (humanidade), *Windy* (ventoso) e *Play Golf* (se irá jogar golfe) que é o atributo alvo (*target*) Santana (2020).

Figura 5: Exemplo de Árvore de Decisão Classificação

Santana (2020)

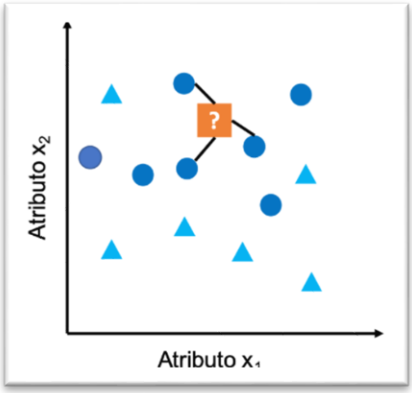
Árvores de Decisão possibilitam a seleção automática das variáveis, pois cada nó interno representa uma decisão sobre um atributo. Para classificar um novo exemplo, basta testar os

valores dos atributos na árvore e percorrê-la até chegar ao chamado nó folha (classe alvo a ser predita), que no exemplo apresentado é se a pessoa irá ou não sair para jogar golf.

2.4.1.1.2 KNN (k-Nearest Neighbours)

O algoritmo KNN (*k-Nearest Neighbours* ou, em português, k-Vizinhos Mais Próximos) é um algoritmo que funciona tanto para problemas de Classificação quanto para problemas de Regressão. Ele não assume premissas sobre a distribuição dos dados, sua ideia principal é considerar que os exemplos vizinhos são similares ao exemplo cuja informação se queira realizar a predição. O KNN considera que cada atributo representa uma dimensão dentro de um espaço n-dimensional, conforme a figura 6.

Figura 6: Exemplo de funcionamento do KNN



Fonte: KOSHIYAMA, ESCOVEDO (2020)

Inicialmente os dados são armazenados, quando um novo registro deve ser classificado ele é comparado a todos os demais dados já armazenados no conjunto de treinamento para identificar os “k” (parâmetros de entrada) vizinhos mais semelhantes de acordo com alguma medida de distância. Apesar de ser um algoritmo muito utilizado, o KNN tem algumas limitações: a performance de classificação pode ser lenta em *datasets* grandes, é sensível a caraterísticas irrelevantes, uma vez que todas as características contribuem para o cálculo da distância e consequentemente, para a predição e é necessário testar diferentes valores de k e a métrica de distância a utilizar KOSHIYAMA, ESCOVEDO (2020).

2.4.1.1.3 Naive Bayes (Bayes Ingênuo)

O *Naive Bayes*, ou Bayes Ingênuo, é um classificador genérico e de aprendizado dinâmico. É um dos métodos mais utilizados para classificação, principalmente pela baixa exigência computacional, poucos dados para treinamento e quando o problema possui um grande número de atributos (características). As primeiras aplicações comerciais do *Naive Bayes* foram na década de 90, com o objetivo de filtrar spam de *e-mails*. O método é bastante utilizado em *software* embarcados e em aplicações de *text mining* (mineração de texto), principalmente para classificação de sentimentos em função de um texto publicado nas redes sociais (por exemplo, identificar se o usuário está feliz ou triste ao publicar um texto). O método é chamado de ingênuo, porque desconsidera as dependências entre os atributos e possui a palavra *Bayes*, porque é baseado no teorema de *Bayes* relacionado com o cálculo de probabilidade condicional.

2.4.1.2 Regressão Supervisionada

A técnica de regressão é similar à técnica de classificação, que consiste em realizar o aprendizado por dados históricos. Além do tipo do resultado, a principal diferença entre os dois problemas está na avaliação de saída: na Regressão, em vez de se estimar a acurácia, estima-se a distância ou o erro entre o resultado do estimador (modelo) e a saída desejada. O resultado de um estimador é um valor numérico que deve ser o mais próximo possível do valor desejado, e a diferença entre esses valores fornece uma medida de erro de estimação do algoritmo que pode ser medido, por exemplo utilizando RMSE (*Root Mean Squared Error* ou Raiz do erro quadrático médio).

Exemplificando o uso de arvore de decisão para regressão, podemos considerar o mesmo *dataset* de pessoas que jogam golf, só que agora com um atributo numérico, que é a quantidade de horas jogadas.

Figura 7: Exemplo de Árvore de Decisão Regressão

Santana (2020)

Na aprendizagem supervisionada, ao mesmo tempo em que o modelo preditivo precisa ser suficientemente flexível para aproximar os dados de treinamento, o processo de treinamento deve evitar que o modelo absorva os ruídos da base KOSHIYAMA, ESCOVEDO (2020).

2.4.1.2.1 Neural network Models (Modelos de Rede Neural)

Em 1943 *MacCulloch* e *Pitts*, criaram o primeiro modelo de aprendizagem para os neurônios que pudessem ser utilizados nos computadores. Este trabalho inspirou *Rosemblatt* a criar o algoritmo chamado de *Perceptron* no final da década de 1950 modelo de neurônio artificial utilizado na maioria dos algoritmos até hoje. O *Perceptron* aprende conceitos, ele pode aprender a resposta com verdadeiro (1) ou falso (0) pelas entradas apresentadas a eles, "estudando" repetidamente os exemplos que lhe são apresentados. Em 1975, *Werbos* criou as camadas neurais, possibilitando que a aprendizagem de um neurônio pudesse ser repassada de um para o outro através do algoritmo *backpropagation*, dessa forma a estrutura poderia aumentar incrementalmente neurônios ou camadas, criando redes multicamadas para a resolução dos problemas. O MLP*Regressor* (*multi layer Perceptron Regressor* ou *Perceptron* Regressão multicamadas), é um algoritmo que pode ser utilizado para problemas de regressão, como por exemplo, diagnóstico de doenças, reconhecimento de rostos de foragidos, hábitos e classificação de perfil de consumo, entre outros (CARDON, MÜLLER 1994).

2.4.2 Aprendizagem não Supervisionada

Técnicas não supervisionados, são quando o conjunto de dados que será utilizado para o treino não possui rótulos com uma resposta correta, permitindo que o algoritmo tire suas

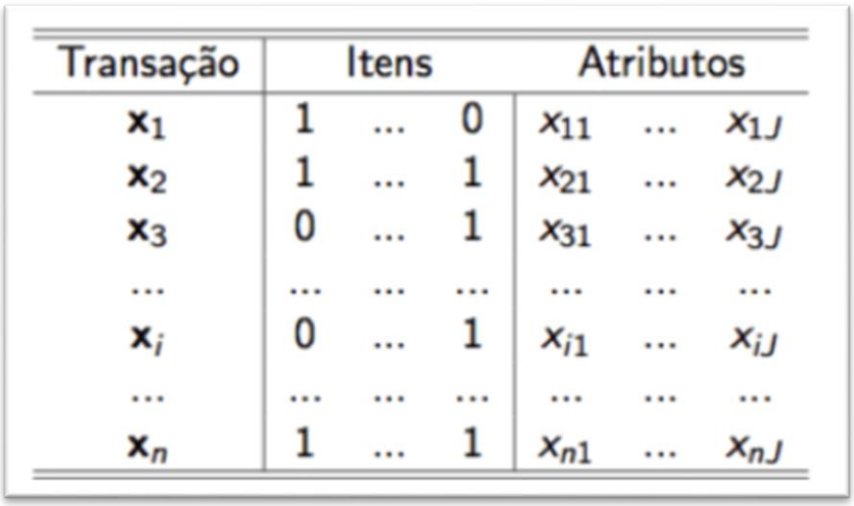
próprias conclusões. Não existe um atributo alvo específico e o objetivo é aprender algo sobre a estrutura ou distribuição do conjunto de dados, buscando padrões, perfis ou itens semelhantes (NORMAN, 2019).

A Aprendizagem não supervisionada tende a ser interessante sobre o aspecto de que o foco não é predições, mas sim em descobrir aspectos relevantes sobre os dados e as técnicas de aprendizagem não supervisionadas podem ser divididas em problemas de Associação ou Agrupamento.

2.4.2.1 Associação Não Supervisionada

Um problema de Associação, para exemplificar, pode ser a identificação de quais clientes com um determinado perfil compram produtos com determinadas características, considerando um *ecommerce* de produtos de beleza. A resposta será obtida considerando a análise de histórico, considerando um grupo de clientes que acessaram e/ou compraram aquele determinado produto. Um problema de associação pode ser definido com um conjunto de “n” transações, em que cada transação é composta por um conjunto de itens. Cada transação pode ou não conter um conjunto de atributos, e o objetivo é construir uma base de regras cujos acessos são associados. A figura 8 ilustra a regra.

Figura 8: Regra de Problema de Associação



Transação	Itens			Atributos		
$\mathbf{x}_1$	1	...	0	x_{11}	...	x_{1J}
$\mathbf{x}_2$	1	...	1	x_{21}	...	x_{2J}
$\mathbf{x}_3$	0	...	1	x_{31}	...	x_{3J}
...	...	...	...	...	...	...
$\mathbf{x}_i$	0	...	1	x_{i1}	...	x_{iJ}
...	...	...	...	...	...	...
$\mathbf{x}_n$	1	...	1	x_{n1}	...	x_{nJ}

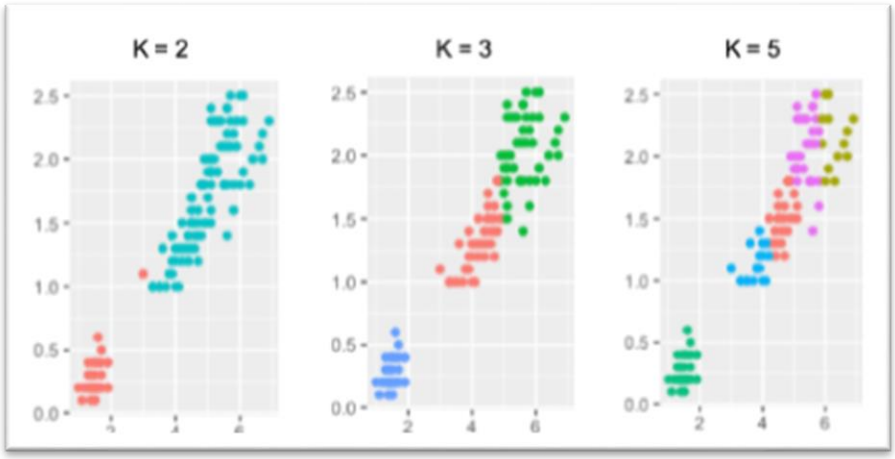
Fonte: KOSHIYAMA, ESCOVEDO (2020)

O Algoritmo de problema de associação mais conhecido é o *Apriori*, que possui como funcionalidade, extrair um conjunto de regras a partir da conjunção de itens mais frequentes na base de dados.

2.4.2.2 Agrupamento Não Supervisionada

Problemas de agrupamento ou clusterização, possuem como objetivo separar as informações de um conjunto de dados em subconjuntos em que os elementos desses novos subconjuntos possuam propriedades comuns que os diferencie dos elementos entre os clusters. Um dos algoritmos de agrupamento mais conhecido é o *K-means*, algoritmo baseado em distâncias. Em linhas gerais, ele calcula a distância existente de "n" pontos de dados x_1, x_2, x_n, tal que cada ponto pertença a um espaço "d" dimensional "R_d". Para separá-los em "k" grupos, deve-se encontrar pontos m_j em R_d, os chamados centroides, de tal forma que a distância entre cada ponto de dado e o centroide mais próximo seja minimizado. Para o correto funcionamento o K-means exige que seja informado previamente o parâmetro k (número de grupos). A figura 9 ilustra como exemplo os resultados do *K-means* considerando 3 diferentes valores de "k".

Figura 9: Resultados em uma base considerando 3 valores de K



Fonte: KOSHIYAMA, ESCOVEDO (2020)

Os projetos de *Quality Assurance* possuem informações que podem ser analisadas de forma estruturada para tomadas de decisões de forma antecipada. Exemplo disso são as falhas encontradas nos projetos. Esses dados podem ser analisados e correlacionados em relação a requisitos, tipos de *software*, criticidade etc., visando a possibilidade de trabalhar nessa base de dados para encontrar insumos focando em construir uma análise preditiva. Assim, este trabalho aplicou um modelo preditivo de *Machine Learning* para identificar a causa raiz das falhas em projetos de *Quality Assurance.*

2.5 PYTHON E JUPYTER NOTEBOOK

Python foi criado por Guido Van Rossum em 1991 e é uma linguagem de alto nível de programação, ou seja, foi criada para ser simples para os serem humanos escreverem e lerem e também é orientada a objetos (organizado em torno ou com base em objetos em vez de ações ou dados em vez de lógica). Segundo a definição de Thompson (2018), Python é muito simples e fácil de aprender, em comparação com outras linguagens de programação, porque requer uma sintaxe que enfatiza a facilidade de leitura e compreensão do código. A facilidade de desenvolver em Python, pode ser exemplificada com as telas de desenvolvimento da mensagem "Olá Mundo!!!, escrita em C, Java e Python nas figuras 1, 2 e 3 (FELTRIN, 2019).

Figura 10: Olá Mundo!!! Desenvolvido em C

Fonte: Fernando Feltrin (2019)

Figura 11: Olá Mundo!!! Desenvolvido em JAVA

Fonte: Fernando Feltrin (2019)

Figura 12: Olá Mundo!!! Desenvolvido em Python

Fonte: Fernando Feltrin (2019)

O desenvolvimento em Python, pode ser feito em diversas IDEs. Para esse trabalho foi utilizado o *Jupyter Notebook*, que é uma aplicação web de código aberto baseada na estrutura cliente-servidor que permite criar e manipular documentos. Ele foi criado a partir do *IPython* em 2014 e é umas das principais ferramentas utilizadas em ciência de dados, pois permite

adicionar componentes *html* de imagens e vídeo, não funcionando apenas como uma IDE, mas também como uma apresentação ou ferramenta educacional.

3 OBJETIVOS

Nessa seção são apresentados o objetivo principal e os secundários para esse trabalho.

3.1. OBJETIVO PRINCIPAL

Foram usados dados de um cliente real da DBServer e o problema de negócio é conseguir prever as *Root Cause* (Causa Raiz) de problemas de um projeto antes deles acontecerem, permitindo que o gestor do projeto possa se planejar e atuar proativamente na inspeção e adequação de processos, regras de negócio, infraestrutura e layout do sistema ou aplicação, evitando que impactem no processo de desenvolvimento do *software*. Para a realização do trabalho, foi utilizado um conjunto de defeitos encontrados em projetos de desenvolvimento e manutenção de *software* e foram aplicadas metodologias de *Data Science* e *Machine Learning.*

3.2 OBJETIVOS SECUNDÁRIOS

- Aplicar o processo de *Data Science* em um *dataset* (conjunto de dados) de falhas obtido em uma empresa real, abrangendo o período de cinco anos;
- Buscar um modelo preditivo que possa ser aplicado em outros projetos.

4. METODOLOGIA

A metodologia representa o caminho que o trabalho seguirá para atingir os objetivos propostos. Esse trabalho de pesquisa utilizou as técnicas e processo de mineração de dados como fluxo de trabalho para a criação de um modelo preditivo.

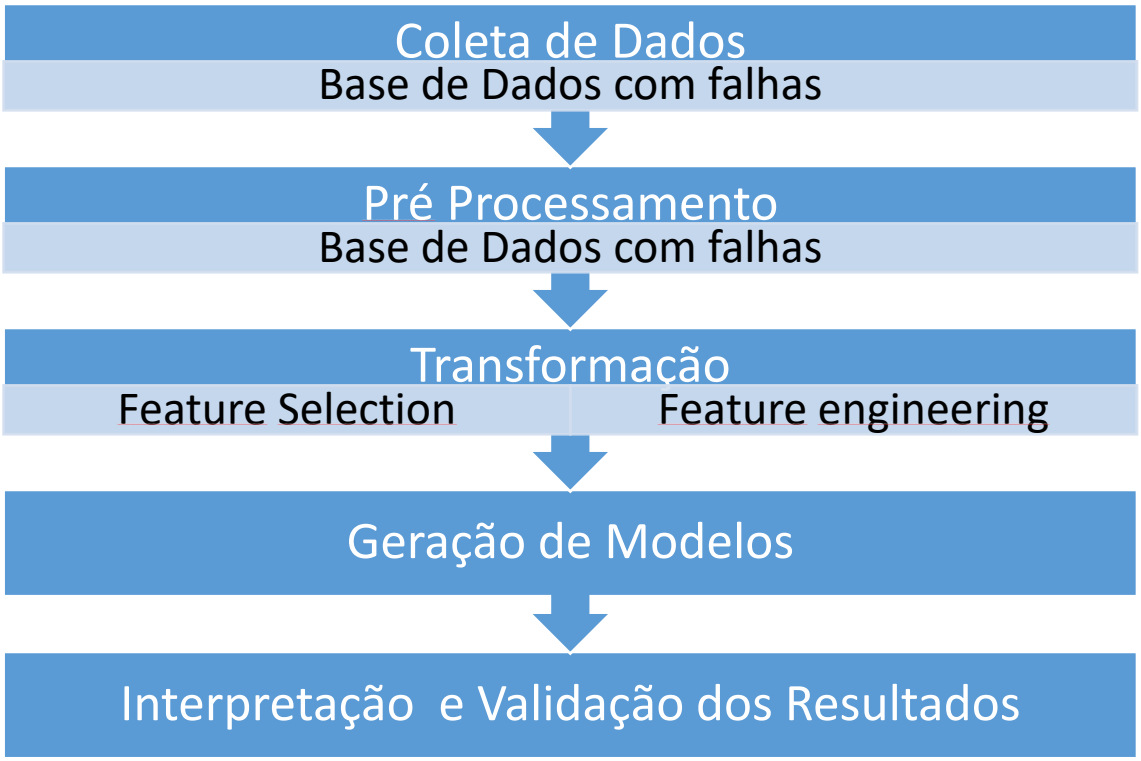
Quanto ao método, podemos dizer que esse trabalho possui uma natureza experimental. Para SEVERINO (2017) a pesquisa experimental tem como propósito submeter as informações pesquisadas em condições técnicas de observação e manipulação experimental, onde são criadas condições adequadas para sua análise. Para tanto o pesquisador deverá selecionar determinadas variáveis e testar as relações funcionais para validar as hipóteses levantadas (SEVERINO, 2017).

4.1 COLETA E ANÁLISE DE DADOS

A interferência do pesquisador foi em cinco momentos:

- O primeiro na separação da base de dados que foi utilizada para a análise das informações,
- O segundo no carregamento desses dados no *Jupyter Notebook* onde foram apresentados de forma preliminar os dados com falhas ainda,
- O terceiro no momento de escolha das variáveis e itens que foram avaliados,
- O quarto na aplicação de um modelo
- O quinto na interpretação e validação dos resultados. A figura 13 apresenta a ilustração das fases do trabalho:

Figura 13: Análise e coleta de dado



Fonte: Elaborado pelo autor (2021)

A seguir a explicação de cada uma das etapas:

a) Coleta de Dados: Foi coletado um *dataset* de informações históricas de projetos relacionados a *Quality Assurance*. Esse *dataset*, por questões de confidencialidade, será anonimizado e trará informações relacionadas à classificação dos erros por severidade, tipo de sistema, data de abertura e causa raiz.

b) Pré-Processamento: Nessa etapa os dados foram analisados, normatizados e ajustados. Essa etapa é de suma importância, pois determinará a qualidade final dos dados que serão analisados.

c) Transformação: Nessa etapa foi realizado o processo de *Feature Selection* (seleção de atributos), que tem como objetivo selecionar as variáveis e de subconjuntos relevantes para a construção do modelo, e *Feature Enginnering* (engenharia de atributos) objetivando melhorar a performance dos algoritmos de *machine learning.*

d) Geração de modelos: Foram aplicados os algoritmos de mineração de dados para geração de modelos com base nos objetivos e metas do trabalho de pesquisa.

e) Interpretação e Validação dos Resultados: Foram analisados os resultados obtidos após rodar os modelos.

5 DETALHAMENTO DAS ATIVIDADES REALIZADAS

Neste capítulo apresenta-se como foi conduzido o trabalho e avaliações das informações do *dataset* disponibilizado para a validação ou não da hipótese da utilização de um modelo preditivo que auxilie na gestão de projetos de *Quality Assurance*. Na seção 5.1 será apresentado como foi realizado a etapa de Coleta, na seção 5.2 será apresentado como foi realizado a etapa de Pré-Processamento, na seção 5.3 será apresentada a Transformação dos dados, elencando as descobertas e tomadas de decisões ao longo dessas etapas. Na seção 5.4 relata a percepção do autor na utilização do modelo e dificuldades encontradas. Veremos na seção 5.5 a interpretação e validação dos resultados obtidos ao rodar o modelo.

5.1 COLETA DOS DADOS

Com a definição da base de dados, a primeira atividade realizada foi o carregamento da Base de Falhas no *Jupyter Notebook* para a partir desse momento avaliar os dados do *dataset* que contém 5.812 falhas de todos os projetos de um dos clientes da DBServer em um intervalo de 8 anos.

Figura 14: Importação de tabela para o *Jupyte*

```
#importar Bibliotecas

import pandas as pd
from tqdm import tqdm


tabela = pd.read_excel('BaseFalhasClienteIndustria.xlsx')
```

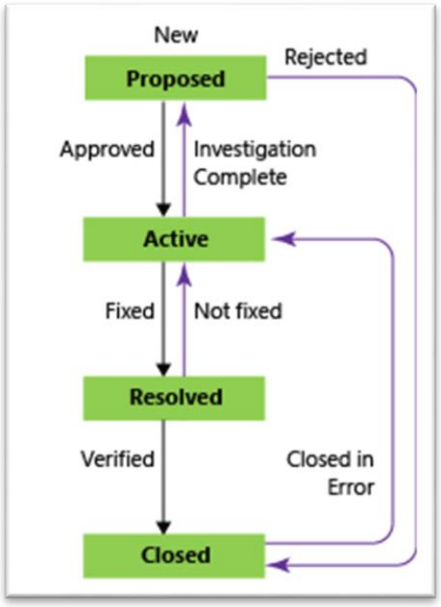
Fonte: O autor (2021)

Analisando as colunas, o *dataset* fornece as seguintes informações:

- ***Work item type*:** Tipo do item (*Bug*, *User story*, *task*) e que nesse *dataset* refere-se a *bugs (Falhas).*
- ***ID*:** Número referente a identificação do item no momento do cadastro no sistema.
- ***Team Project*:** Agrupados sistêmico em relação ao projeto.
- ***Title*:** título da falha que foi cadastrado na ferramenta.
- ***Assigned To*:** Para quem a falha está designado.

- ***State*:** Status da falha. *Proposed* (aberto, mas ainda não analisado), *Active* (em correção), *Resolved* (corrigido, mas ainda não retestado), *Closed* (retestado e fechado).
- ***Severity*:** Severidade das falhas encontrados, *low* (impacto praticamente nulo para o negócio / Sistema, *medium* (impacto baixo para o sistema), *high* (impacto grade para a fidelização do cliente ou impacto financeiro) ou *critical* (sistema não funciona).
- ***Created Date***: Data de criação do item.
- ***Closed Date**:* Data que o item foi fechado.
- ***Reason*:** Razão da falha, utilizado para esclarecer o motivo do estado da falha após uma mudança de *status*, *Approved*, *Investigation Complete*, *Rejected*, *Fixed*, *Not fixed*, *Verified ou Closed in error*, conforme imagem:

Figura 15: *Workflow* de mudanças de *status* e motivos das falhas

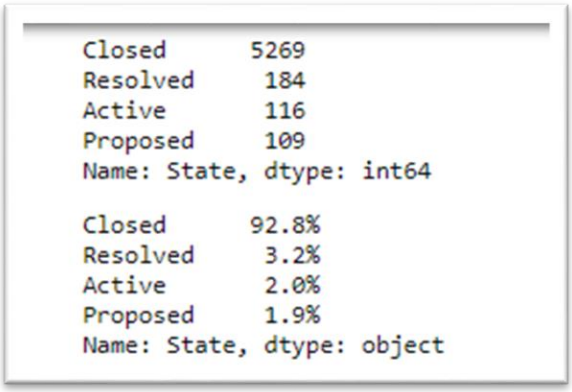


Fonte: Microsoft (2021)

- ***Root Cause***: Causa raiz da falha, *Coding Error* (erro de código), *Specification Error* (erro de especificação), *Design Error* (erro de projeto), *Communication Error* (erro de comunicação / entendimento) ou *Unknown* (origem da falha desconhecido).
- ***Iteration Path***: Etapa onde a falha foi encontrado, Teste Integrado, Homologação ou Produção.

Realizando uma análise inicial é possível concluir que 92.8% das falhas foram retestados e fechados e 7.2% ou não foram corrigidos (*active* e *proposed*) ou não foram retestados (*resolved*) em um *dataset* contendo 5.812 itens.

Figura 16: Status das falhas



Fonte: O Autor (2021)

5.2 PRÉ PROCESSAMENTO

A base de dados apresenta as falhas encontrados em diversos projetos realizados entre 2013 e 2020. Projetos esses que, para facilitar a análise, serão agrupados em relação a sua característica e objetivo da seguinte forma:

- **Sistemas de BI**: Sistemas relacionados a análise de empresarial, mineração de dados e visualização de dados para auxiliar na tomada de decisões.
- **Portal Informativo**: Portais criados para apresentar informações relacionadas a um tema ou processo.
- **Sistema ERP**: Sistemas de Gestão integrada empresarial, principais sistemas da companhia.
- **Sistema de CRM**: Sistemas de Gestão de relacionamento com os clientes.
- **Sistema de Gestão Materiais**: Sistemas de gerenciamento de matéria prima para a produção dos produtos.
- **Sistema de Gestão Vendas**: Sistemas de gestão das vendas dos produtos para os clientes.
- **Sistema Financeiro:** Sistema de controle e gestão financeira das matérias primas e produtos.

- **Sistema Fidelidade**: Sistemas de gamificação e gestão da fidelidade e bonificação dos clientes.
- **Sistema de Gestão de Contratos**: Sistemas Jurídicos de gestão de contratos e assuntos legais.
- **Sistema GED**: Sistema de catalogação e armazenagem de documentos.
- **Sistema Gestão Projetos**: Sistemas relacionados ao acompanhamento e controle dos projetos.
- **Sistema ITSM**: Sistema de gestão de incidentes e chamados da empresa.
- **Sistema Gestão Viagens**: Sistemas relacionados ao planejamento gestão e reembolso de viagens corporativas.
- **Sistema SAC**: Sistemas de Atendimento aos clientes.

5.3 TRANSFORMAÇÃO DOS DADOS

Das informações disponibilizadas conforme as colunas apresentadas, podemos fazer uma primeira análise, que considerando o objetivo do estudo, nem todas as colunas são necessárias, por esse motivo iremos retirar as colunas *ID, Work item type, Title, Assigned To, Created Date e Closed Date*, construindo um novo *dataset* com as colunas *Team Project, State, Severity, Reason, Root Cause e Iteration Path*.

Figura 17: Novo *dataset* falhas

	Team Project	State	Severity	Reason	Root Cause	Iteration Path
0	Omega	Closed	Medium	Verified	Coding Error	Teste Integrado
1	Iota	Closed	Medium	Verified	Coding Error	Teste Integrado
2	Iota	Closed	Medium	Verified	Coding Error	Teste Integrado
3	Iota	Closed	Medium	Verified	Coding Error	Teste Integrado
4	Iota	Closed	Medium	Verified	Coding Error	Teste Integrado
...	...	...	...	...	...	...
5807	Lambda	Active	High	Approved	Unknown	Teste Integrado
5808	Lambda	Proposed	High	Investigation Complete	Unknown	Teste Integrado
5809	Lambda	Proposed	Medium	Investigation Complete	Unknown	Teste Integrado
5810	Lambda	Active	Medium	Not fixed	Unknown	Teste Integrado
5811	Lambda	Closed	Medium	Verified	Coding Error	Teste Integrado

Fonte: O autor (2021)

Por questão de anonimizar os dados, antes de realizar a geração dos modelos, foram substituídos os nomes dos agrupadores no *Team Project* pelos codinomes: Alpha, Beta, Gamma, Delta, Epsilon, Zeta, Eta, Theta, Iota, Kappa, Lambda, Mu, Sigma e Omega.

5.4 GERAÇÃO DE MODELO

Para utilizar um modelo de predição, é importante primeiro entender a classificação do problema que estamos trabalhando e também se utilizaremos algum modelo supervisionado ou não supervisionado. Todas as variáveis escolhidas para a análise (*Team Project, State, Severity, Reason, Root Cause e Iteration Path),* são variáveis categóricas, sendo que a variável *Severity,* é categórica Ordinal, por que representa uma hierarquia entre as opções (*critical, high, medium e low)* e as demais variáveis categóricas nominais, pois representam apenas texto.

O objetivo desse trabalho é utilizar um modelo de *machine learning* que possa prever a classificação das falhas dos projetos, considerando os demais atributos apresentados no *dataset*, portanto trata-se de um problema de classificação e que o indicado é utilizar algoritmos supervisionados para a análise. *Root Cause* das falhas encontrados, é um dos atributos mais importantes, por que ajuda os gestores e times a identificar quais etapas do ciclo de vida da construção de um *software* que precisa de mais atenção por parte dos gestores para a redução de falhas em projetos futuros, dessa forma retiramos do *dataset* linhas com valores nulos, utilizando a função *.dropna* (*dataset* ficou com 5636 itens válidos) e o atributo *Root Cause*, através da função *.drop*, para utilizarmos como atributo de análise do modelo de predição (*target*).

Figura 18: Transformando o atributo *root cause* na variável “Y”

```
bugs_df = bugs_df.dropna()

X = bugs_df.drop('Root Cause', axis=1)
y = bugs_df['Root Cause']

for coluna in X.columns:
    X[coluna] = le.fit_transform(X[coluna])

y = le.fit_transform(y)

X
```

Fonte: O Autor (2021)

Foi utilizada a técnica *Label Encoder* (codificação de etiqueta / rótulo) da biblioteca *Sklearn*, que consiste em transformar cada uma das classes em número inteiro exclusivo com base na ordem alfabética (SCIKIT LEARN, 2021).

Figura 19: *Dataset* após aplicar a técnica *Label Encoder*

	Team Project	State	Severity	Reason	Iteration Path
0	10	1	3	5	2
1	6	1	3	5	2
2	6	1	3	5	2
3	6	1	3	5	2
4	6	1	3	5	2
...	...	...	...	...	...
5807	8	0	1	0	2
5808	8	2	1	2	2
5809	8	2	3	2	2
5810	8	0	3	3	2
5811	8	1	3	5	2

Fonte: O autor (2021)

Para a análise do objetivo do trabalho, foi utilizada o modelo *Decision Tree Classifier* (classificador de árvore de decisão) da biblioteca *Scikit-learn*, e separado o *dataset* (Conjunto de dados) em duas partes, uma para o modelo ser testado e o outro para o modelo ser aplicado, visando validar se é possível prever a *root cause* das falhas com base nos outros atributos.

Figura 20: Treinando o modelo *Decision Tree Classifier*

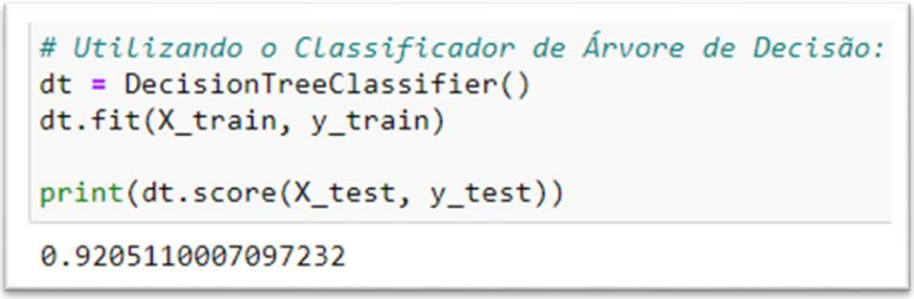
```
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=10)

dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)

print(dt.score(X_test, y_test))
```

Fonte: O autor (2021)

5.5 INTERPRETAÇÃO E VALIDAÇÃO DOS RESULTADOS

Com base no modelo *Decision Tree Classifier* é possível afirmar, com uma acurácia de 92,99% que o modelo conseguiu prever a *Root Cause* com base nos outros atributos do *dataset*.

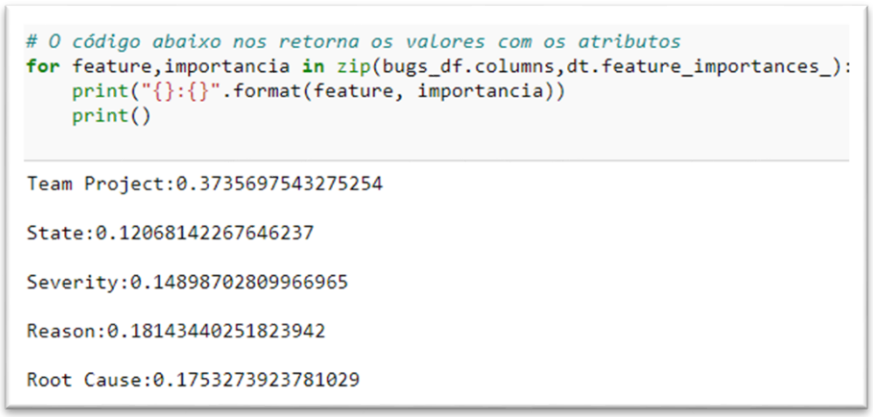
Figura 21: Utilizando o Classificador de Árvore de Decisão



Fonte: o autor (2021)

Em relação a importância dos atributos, o que apresentou mais relevância para a previsão da root cause foi o *Team Project*, ou seja existe uma relação muito forte entre o tipo de projeto e *root cause*, o que sugere um olhar detalhado nos tipos de projetos e entender dentro do ciclo de vida de cada um deles o que pode ser melhorado para a redução das falhas em projetos.

Figura 22: Porcentagem de importância dos atributos



Fonte: o Autor (2021)

Uma boa prática para validar as acurácias de um modelo é realizar 10 medições para identificar se ao rodar novamente e os dados mudarem, se ocorrerá uma grande variação da acurácia apresentada. Conforme a tabela 1 é possível identificar que a acurácia não apresentou muita variação, validando as informações geradas.

Tabela 1: Média das Acurácias

Medições	Acurácia
1	93.04%
2	93.47%
3	92.26%
4	93.12%
5	92.62%
6	93.54%
7	93.75%
8	93.19%
9	91.63%
10	93.26%
Média	**92.99%**

Fonte: O autor (2021)

Além da acurácia foi calculado a Precisão, Revocação e a Pontuação F1, que é uma média harmônica entre a Precisão e Revocação, conforme a figura 23 é possível confirmar acurácia de 92% e a distribuição da Precisão e Revocação conforme as variáveis de *Root Cause*, que por questões de confidencialidade manteremos como atributos de 0 a 4 sem apresentar os rótulos.

Figura 23: Métricas

```
#Utilização das Métricas para Calcular Precisão,
#Revocação e Pontuação F1
from sklearn import metrics
print(metrics.classification_report(y_test,resultado))
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1296
           1       0.43      0.19      0.26        16
           2       0.00      0.00      0.00        17
           3       0.50      0.02      0.03        57
           4       0.95      0.83      0.88        23

    accuracy                           0.93      1409
   macro avg       0.56      0.41      0.43      1409
weighted avg       0.90      0.93      0.91      1409
```

6 CONCLUSÃO

Ao longo desse trabalho foi possível estabelecer uma análise em relação ao conjunto de dados apresentado, onde apresentou de forma objetiva a aplicação de um modelo de *Machine Learning* para a predição da causa raiz de um conjunto de dados de falhas de vários projetos.

O objetivo geral do trabalho era "Analisar um conjunto de defeitos encontrados em projetos de desenvolvimento e manutenção de *software*, aplicando metodologias de *Data Science* e *Machine Learning"* o qual foi atingido com a aplicação de técnicas de DS e ML no *dataset* "basefalhasclienteIndustria.xlsx".

Entretanto, para o alcance do objetivo geral alguns objetivos específicos foram detalhados realizando a definição de qual atributo do *dataset* seria analisado. Foi definido que o atributo *Root Cause* é um fator importante dentro do âmbito das análises de falhas encontradas em um projeto, por esse motivo ficou definido a utilização desse atributo como "*target*". Ao longo da análise dos resultados, conseguimos evidenciar a correlação com o atributo *Team Project*, ou seja, o tipo de projeto possui uma alta contribuição para o modelo preditivo prever a causa raiz do *bug*.

Como parte integrante desta conclusão é impossível não citar os próximos passos que se aconselha para este trabalho futuro, de forma a viabilizar a aplicabilidade desse modelo em outros projetos e até mesmo outros problemas. Aconselha-se a aplicação desse *dataset* (ou outro com mais informações) com outros algoritmos de classificação, como por exemplo *Naive Bayes* (Bayes Ingênuo), que é um classificador muito utilizado quando existe um volume muito grande de dados ou KNN (*k-Nearest Neighbours* ou, k-Vizinhos Mais Próximos) que é um algoritmo simples de entender e que funciona muito bem na prática tanto para problemas de Classificação quanto para problemas de Regressão. Uma outra sugestão é identificar dentro da empresa quais outros problemas do dia a dia DS e ML poderiam auxiliar na tomada de decisão, como por exemplo a predição dos chamados em produção de um determinado sistema com base no histórico de chamado, o custo de um projeto com base no histórico de projetos similares ou até mesmo o perfil profissional mais adequado para atender determinado tipo de projeto.

Data Sciense e Machine Learning são temas que estão na vanguarda da TI e cada vez mais essas técnicas farão parte do dia a dia das empresas, quem não se interessar por conhecer e explorar os benefícios de DS e ML, corre o risco de ficar estagnado no mercado.

REFERÊNCIAS

ASSESPRO - **ASSOCIAÇÃO DAS EMPRESAS BRASILEIRAS DE TECNOLOGIA DA INFORMAÇÃO**. Disponível em: <http://www.assespro.org.br/version2010/index.asp>. Acessado em: 15 de fev. 2021

BARTIÉ, Alexandre. **Garantia de Qualidade de Software:As melhores práticas de Engenharia de Software aplicadas a sua empresa**. 6° Edição. São Paulo: Elsevier,2002.

BASTOS, Aderson, RIOS, Emerson, CRISTALLI, Ricardo, MOREIRA, Trayahú. **Base de Conhecimento em Teste de Software**. 2ª Edição. São Paulo: Martins Fontes, 2007.

BIBLIOTECA SCIKIT LEARN disponível em: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html acessado em 15/07/2021

BIBLIOTECA PANDAS disponível em: < https://pandas.pydata.org/pandas-docs/version/0.21/index.html> acessado em 10/02/2021

CADY, Field. **The data Science handbook**. New Jersey: John Wiley & Sons, 2017.

CARDON, André; MÜLLER, Daniel Nehme. **Introdução às Redes Neurais Artificiais**. Porto Alegre: UFRGS, 1994

GÉRON, Aurélien. **Mãos à Obra Aprendizado de Máquina com Scikt-Learn & TensorFLow: Conceitos, Ferramentas e Técnicas Para a Construção de Sistemas Inteligentes**. Traduzido por Rafael Contatori Rio de Janeiro: Alta Books, 2019.

KOSHIYAMA, Adriano.; ESCOVEDO, Tatiana. **Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise.** São Paulo: Casa do Código, 2020

NORMAN, Alan T. **Aprendizagem de máquina em ação: Uma Obra Para O Leigo, Guia Passo A Passo Para Novatos.**

ROMERO, Sonia Mara Thater; NASCIMENTO,Belmiro J.C Métodos de Pesquisa. In: FOSSATI, Nelson C.; LUCIANO, Edimara Mezzomo. **(Orgs) Prática Profissional em Administração: Ciência, Método e Técnicas**. 1ª Edição. Porto Alegre: Sulina, 2008, p. 51-64.

SEVERINO, Antônio Joaquim. **Metodologia do trabalho científico**. São Paulo: Cortez, 2017.

TAURION, Cezar. **Big Data**. Rio de Janeiro: Brasport, 2013

FELTRIN, Fernando. **Python do Zero a programação orientada a objetos**. Disponível em: < https://livrariapublica.com.br/python-do-zero-a-programacao-orientada-a-objetos-fernando-belome-feltrin/> acessado em 16/07/2021

SANTANA, Felipe. Disponível em https://minerandodados.com.br/arvores-de-decisao-conceitos-e-aplicacoes/ acessado em 25/07/2021