**UNICESUMAR - CENTRO UNIVERSITÁRIO DE MARINGÁ**
CENTRO DE CIÊNCIAS EXATAS TECNOLÓGICAS E AGRÁRIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

# ANÁLISE DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DE EQUIPES DE UMA INSTITUIÇÃO DE ENSINO

**FELIPE CARNEIRO MAGRINELLI**

MARINGÁ – PR
2022

Felipe Carneiro Magrinelli

# ANÁLISE DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DE EQUIPES DE UMA INSTITUIÇÃO DE ENSINO

Artigo apresentado ao Curso de Graduação em Engenharia de Software da UNICESUMAR – Centro Universitário de Maringá como requisito parcial para a obtenção do título de Bacharel(a) em Engenharia de Software, sob a orientação do Prof. Dr. Nelson Tenório.

MARINGÁ – PR

2022

**FOLHA DE APROVAÇÃO**

FELIPE CARNEIRO MAGRINELLI

# ANÁLISE DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DE EQUIPES DE UMA INSTITUIÇÃO DE ENSINO

Artigo apresentado ao Curso de Graduação em Engenharia de Software da UNICESUMAR – Centro Universitário de Maringá como requisito parcial para a obtenção do título de Bacharel(a) em Engenharia de Software, sob a orientação do Prof. Dr. Nelson Tenório.

Aprovado em: ____ de _____ de _____.

BANCA EXAMINADORA

_____

Nome do professor – (Titulação, nome e Instituição)

_____

Nome do professor - (Titulação, nome e Instituição)

_____

Nome do professor - (Titulação, nome e Instituição)

# ANÁLISE DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DE EQUIPES DE UMA INSTITUIÇÃO DE ENSINO

Felipe Carneiro Magrinelli

**RESUMO**

Com o grande aumento na produção de dados nas últimas duas décadas, a extração de informação dessa massiva rede de dados aparentemente infinita tornou-se necessária para as organizações atuais. Pode-se dizer que em diversas áreas, incluindo a engenharia de software, é procurado na mineração de dados uma abordagem para auxiliar na resolução de problemas inerentes a elas. Sendo assim, o objetivo dessa pesquisa foi analisar o processo de desenvolvimento do software ágil por meio da técnica de agrupamentos (clustering) com funcionalidades provenientes de determinados dados históricos voltados a uma base de ensino do desenvolvimento de software. Por isso, foi apresentado ao longo desse trabalho uma análise de todo o processo do desenvolvimento de software utilizando técnicas de mineração de dados. Portanto foram analisados os dados referentes a matéria do Projeto Integrador da UniCesumar, que engloba projetos de desenvolvimento de software em equipes baseado na metodologia ágil. Ao final, foram levantadas três suposições acerca dos resultados encontrados. Estas suposições foram discutidas separadamente, considerando os aspectos da engenharia de software e da metodologia ágil. Assim, concluiu-se que o processo pode ser utilizado para a extração de informações voltadas aos projetos de desenvolvimento de software. Para trabalhos futuros, recomendou-se a utilização de outras técnicas de mineração de dados e a utilização de diferentes fontes de dados para avaliação de viabilidade.

**Palavras-chave**: Desenvolvimento Ágil. Engenharia de Software. Mineração de Dados.

# ANALYSIS OF THE SOFTWARE DEVELOPMENT PROCESS OF TEAMS IN AN EDUCATIONAL INSTITUTION


## ABSTRACT

With the large amount of data produced in the last two decades, extracting information from this massive network and seemingly infinite data has become necessary for organizations nowadays. It is possible to say that in diverse areas, including software engineering, It is searched in data mining to help solve problems inherent to them. In this way, the objective of this work was to analyze the agile software development process through grouping (clustering) technique functionalities from historical data of a software development teaching base. Thus, this work focus on the software development analysis process using data mining techniques. For this, data referring to the subject "Projeto Integrator" of UniCesumar, which comprises projects of software development teams based on the agile methodology, were analyzed. In the end, were raised three assumptions about the results found. These assumptions were discussed separately, considering the software engineering aspects and agile methodology. Thus, it was concluded that the process could be used to extract information from software development projects. For future works, it was recommended the use other data mining techniques, and to use of different data sources for feasibility.



**Keywords**: Agile Development. Data Mining. Software Engineering.

# 1 INTRODUÇÃO

Hoje em dia, é possível afirmar que vivemos na era da informação e da tecnologia. Devido à grande massificação do uso de tecnologias e a produção dos dados que estão em crescimento constante, (RAUTENBERG; CARMO, 2019), pode-se dizer que isso influencia as organizações a dependerem desse conhecimento para trilhar os planos de negócio e se manter competitivamente dentro do mercado de trabalho (SELAMAT et al., 2020). Desde o começo dos anos 90, os pesquisadores denotaram interesse pela mineração de dados e informações (FAYYAD et al., 1996), mas foi a partir dos anos 2000 que, com o aumento expressivo na produção de dados advindos das inovações tecnológicas, que os estudos de mineração de dados vêm sendo cada vez mais realizados, e isso com o objetivo de utilizar os dados na tomada de decisão e descoberta de padrões (CWEI et al., 2022). Determinadas áreas, como a medicina por exemplo, (ISLAM et al., 2018), a educação (ROMERO; VENTURA, 2020) e a agricultura (PADILHA et al., 2022) também se beneficiam da mineração de dados pelo caráter analítico e metodológico desse processo, que estão relacionados aos problemas intrínsecos que ocorrem nessas áreas.

Ocasionalmente, o estudo da mineração de dados pode ser aplicada na engenharia de software, principalmente relacionado aos problemas recorrentes em projetos de desenvolvimento, como em detecção de falhas (WANG, 2021), predição de defeitos e escaneamento de vulnerabilidade (LUZGIN; KHOLOD, 2020). Segundo Kiyak (2020), o encontro de defeitos, bugs, vulnerabilidades e falta de coesão nos códigos dos primeiros estágios de desenvolvimento são cruciais para garantir a qualidade final do produto, sendo justamente essa interdisciplinaridade que se busca resolver nessas ocasiões.

Neste sentido, a implementação da gestão do conhecimento nas organizações pode ser fundamentada a partir da utilização das informações descobertas através do estudo da mineração de dados (SELAMAT et al., 2020). Para os times de tecnologia em específico, segundo Apenko e Romanenko (2019), a relevância da gestão do conhecimento se dá principalmente por causa da flexibilidade que ocorre em projetos que seguem a metodologia ágil, vinculada à ideia de ter os integrantes da equipe como mantenedores de tal conhecimento. Dessa maneira, o compartilhamento de informações, aprendizados e estágios do processo de desenvolvimento do projeto entre os integrantes são formas de manter o projeto sustentável e aumentar as chances de sucesso (APENKO; ROMANENKO, 2019).

A motivação para a realização dessa pesquisa surgiu com a possibilidade de efetuar e contribuir com um processo de análise e verificação dos processos do desenvolvimento de software a partir da base de dados com as informações dos projetos ágeis.

Nesse sentido, a problematização dessa pesquisa nesse estudo é “Como avaliar o processo do desenvolvimento de software ágil, partindo da base histórica do ensino para equipes em desenvolvimento?”. Consequentemente, ao longo dessa pesquisa, foi verificado primeiramente os conceitos introdutórios sobre os estudos de mineração de dados. Em seguida, na segunda seção, são apresentados os objetivos gerais e os objetivos específicos necessários para alcançar o objetivo final proposto no trabalho. Na terceira seção, é apresentado um aprofundamento teórico sobre a mineração de dados, suas aplicações e os pontos relevantes para o seu entendimento. Na quarta seção, são apresentados os métodos utilizados neste trabalho. Na quinta e sexta seção apresentam-se, respectivamente, os resultados encontrados e as discussões finais. Por último, na sétima seção, as considerações finais englobam os principais pontos deste trabalho e sugere-se futuras pesquisas a partir do que foi encontrado.

## 1.1 OBJETIVO GERAL

Essa pesquisa tem por objetivo geral analisar o processo do desenvolvimento de software ágil por meio da técnica de agrupamentos (i.e., clustering) e funcionalidades provenientes dos dados históricos de uma base para o ensino e desenvolvimento do software.

## 1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo final proposto nessa pesquisa, fez-se necessário os seguintes objetivos específicos:

(a) Estudar os processos de mineração de dados na literatura
(b) Efetuar o processamento e tratamento dos dados disponibilizados
(c) Aplicar métodos de mineração de dados
(d) Apresentar os resultados encontrados
(e) Sugerir hipóteses após a análise

# 2 MINERAÇÃO DE DADOS

Segundo Cwei et al. (2022), a mineração dos dados é um processo analítico de exploração que busca padrões consistentes ou relação entre variáveis relacionadas a uma base de dados. A mineração de dados voltadas ao campo de estudo, nasceu de uma ramificação da área de Machine Learning (ML) e seus esforços se concentraram na utilização de algoritmos de ML para a análise de aplicações e resolução de problemas do negócio (FAWCETT; PROVOST, 2018). Por esse motivo, as técnicas de mineração de dados são amplamente utilizadas atualmente por organizações que buscam o crescimento e a expansão (OSMAN, 2019). Entretanto, é importante compreender que a utilização de diferentes técnicas culminará em alcançar diferentes resultados. Isso demonstra que, apesar da flexibilidade que essas técnicas propõem, a identificação e o entendimento do problema a ser resolvido é uma etapa necessária para atingir o resultado final (OSMAN, 2019).

Nesta primeira seção, serão apresentados os pontos considerados importantes para o entendimento do funcionamento e das diferentes aplicações da mineração dos dados atuais.

## 2.1 MINERAÇÃO DE DADOS NA ENGENHARIA DE SOFTWARE

A seguir, serão apontados e descritos as pesquisas e trabalhos relacionados com a mineração de dados no âmbito da engenharia de software encontrados nos últimos três anos.

Em Kiyak (2020), é feita uma revisão dos estudos de mineração de dados considerando algumas tarefas da engenharia de software (estimativa de esforço, predição de defeito, análise de vulnerabilidade, refatoração e padrão de design), e posteriormente foram avaliados os algoritmos e conjuntos de dados utilizados. Para a predição de defeito, foi demonstrado que as técnicas de classificação são as mais utilizadas. Para a estimativa de esforço, foi concluído que redes neurais é a técnica mais utilizada nos estudos destas tarefas. Para a parte de análise de vulnerabilidade, verificou-se que, juntamente com as técnicas de classificação e mineração de texto, que as técnicas de clusterização são comumentemente utilizadas nestas situações. Para as tarefas de mineração padrões de design e refatoração, foram mostradas que as técnicas de classificação são as mais utilizadas para ambos. Ao final do estudo, concluiu-se que as técnicas de classificação são as técnicas de mineração de dados mais utilizadas neste âmbito da engenharia de software.

Em Wang (2021), foi apresentado um estudo de aplicação da mineração de dados na engenharia de software. O estudo cita três pontos gerais dos softwares: escaneamento de

vulnerabilidade, gestão de testes e detecção de falhas. Além disso, é proposto estratégias para resolvê-las. Para efetuar o escaneamento das vulnerabilidades de um software, o estudo sugere que seja considerado cada procedimento do código como um registro do dataset, e que seja utilizado o *fuzzy clustering* para efetuar as análises após uma prévia limpeza dos dados. O *fuzzy clustering*, neste sentido, permite que os pontos de dados analisados possam estar em mais de um cluster ao mesmo tempo, tornando mais fácil a sua análise e o encontro de vulnerabilidades. Para a gestão dos testes, o estudo sugere considerar cada requisito do software como um elemento no dataset, de modo que cada elemento é um vetor multidimensional. Dessa forma, extrai-se as *features* dos requisitos (existentes e novos), e aplica-se a clusterização sobre estes dados selecionados. Em seguida, pode ser feita a análise e estimativa do esforço e da quantidade do código para a nova demanda de teste. Para a detecção de falhas, o estudo sugere que seja feito todo o processo de coleta, transformação e limpeza dos dados, assim como a utilização de métodos de classificação com o intuito de predizer e classificar novas falhas. Sendo assim, com a entrada de novas informações, pode-se dizer que o algoritmo alertaria, com base na classificação, os possíveis pontos de falhas. Segundo o estudo, isso ajudaria no encontro de problemas que poderiam causar danos à segurança e a saúde do software.

Em Luzgin e Kholod (2020), foi realizado uma revisão sobre a MSR (Mineração de Repositórios de Software). O estudo defende que a qualidade e a segurança do software podem ser resolvidas analisando as informações não estruturadas dos artefatos do próprio software. Para isso, o artigo propõe algumas técnicas para realizar a MSR em alguns pontos de um software. O quadro 1 sumariza a proposta feita pelo artigo, abordando técnicas para diferentes complexidades e itens de uma mineração de dados em repositórios de software.

## 2.2 ALGORITMOS DE MACHINE LEARNING

*Machine Learning* (ML) é um subcampo de pesquisa que tem como base principal a Inteligência Artificial (IA) (FAWCETT; PROVOST, 2018). A ML utiliza teorias estatísticas para a criação de modelos matemáticos, e utiliza a ciência da computação para a aplicação destes modelos, tanto no treinamento, quanto para a execução de fato (ALPAYDIN, 2020). Mas ainda assim, pode-se dizer que o ponto principal dessa área é que, apesar de não ocorrer a identificação do processo analisado por completo, o aprendizado da máquina permite a construção de algo próximo ou que ajude a entender o problema. Diante disso, nos últimos anos é possível afirmar que esse subcampo tem crescido (FAWCETT; PROVOST, 2018), e é

utilizado em diversos setores, como nas áreas médicas, financeiras e comerciais (ALPAYDIN, 2020).

Algoritmos de ML são divididos principalmente em quatro categorias: aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço (SARKER, 2021). Essa divisão é feita baseada nas técnicas utilizadas pelos algoritmos. Entretanto, para a resolução de problemas dessa divisão, isso não é uma barreira, e a utilização de mais de uma categoria de algoritmo pode ser benéfica para o processo, uma vez que cada um pode atuar de forma expressiva, considerando suas determinadas capacidades (MARSLAND, 2015).

**Quadro 1**: Evolução das técnicas usadas em MSR a partir de seu nível de complexidade

| MSR | Técnicas | | |
|---|---|---|---|
| | Básico | Intermediário | Avançado |
| Reconhecimento de Entidades | Processamento de linguagem natural (PLN) e algoritmos baseados em regras heurísticas | Técnicas de aprendizado | Low-Shot (um tipo de aprendizado supervisionado) e técnicas de aprendizado semi-supervisionado. |
| Predição de Defeito | Utilização de modelos baseado em padrões de projeto e métricas de software | Análise de texto e PLN | Emsemble learning (conjunto de técnicas de aprendizado supervisionado) e técnicas de aprendizado não-supervisionado |
| Sumarização de Artefatos de Software | Utilização de modelos baseado em vetores | Técnicas de aprendizado supervisionado | Técnicas de aprendizado não-supervisionado |
| Generalização de Mensagens de *Commit* | Algoritmos baseados em regras heurísticas | Utilização de técnicas de PLN mais complexas | *Deep learning* |

Fonte: Adaptado de (LUZGIN; KHOLOD, 2020).

### 2.2.1 Estatísticas em *Machine Learning*

Na estatística, as variáveis definem as características dos dados, de modo que podemos agrupá-las em dois grupos: variáveis numéricas e variáveis categóricas.

As variáveis numéricas ou quantitativas podem ser discretas, as quais definem valores contáveis e finitos com uma lista de valores enumeráveis, como quantidade de linhas de um código, número de reclamações de um cliente etc. (ASSUNCAO, 2017). Além disso, as variáveis numéricas podem ser contínuas, podendo definir valores infinitos e assumir qualquer valor num intervalo de reta real, como comprimentos de equipamentos, data e hora de um pagamento, altura de uma pessoa etc. (ASSUNCAO, 2017).

As variáveis categóricas ou qualitativas possuem um número finito de categorias, indicando o conjunto de dados, e a forma que eles podem ser classificados (ASSUNCAO, 2017). Essas variáveis, portanto, incluem finitas possibilidades de rótulos nas quais não são submetidos a operações aritméticas (ASSUNCAO, 2017).

Um outro ponto importante no que diz respeito à estatística é o cálculo de distâncias. Para o cálculo de similaridade das classes e dos dados, que será explicado ao decorrer das próximas páginas, os algoritmos utilizam funções matemáticas para quantificar e atribuir uma distância numérica entre esses valores. Uma das formas de realizar esse cálculo é utilizando a Distância Euclidiana, que realiza o cálculo baseando no menor caminho possível entre dois pontos (uma linha reta) (MARSLAND, 2015). Quando um objeto é caracterizado por *n* variáveis, pode-se dizer, portanto, que, a equação geral para o cálculo da distância entre os pontos *p* e *q* pode ser definida como:

$$d(p,q) = \sqrt{\left(d_{1,p} - d_{1,q}\right)^2 + \left(d_{2,p} - d_{2,q}\right)^2 + \cdots + \left(d_{n,p} - d_{n,q}\right)^2} \quad (1)$$

na qual *p* e *q* representam os pontos (dados), e *n* representa a quantidade de atributos.

Outra forma de calcular as distâncias é utilizando a Distância de Manhattan. Diferente da Euclidiana, Manhattan calcula a distância absoluta entre as coordenadas dos pares de dados (KARKI; RANJITKAR, 2016). A equação geral da Distância de Manhattan pode ser definida como:

$$d(p,q) = \left|d_{1,p} - d_{1,q}\right| + \left|d_{2,p} - d_{2,q}\right| + \cdots + \left|d_{n,p} - d_{n,q}\right| \quad (2)$$

onde *p* e *q* representam os pontos (dados), e *n* representa a quantidade de atributos.

### 2.2.2 Aprendizado Supervisionado

Os algoritmos de Aprendizagem Supervisionada (AS) são baseados em um mapeamento de entrada e saídas. As saídas são produzidas a partir de inferências estatísticas baseada nos dados de entrada e, portanto, esses algoritmos necessitam de dados externos prévios para serem treinados (MAHESH, 2018). Os dataset dos treinos consistem em um grande volume de dados, geralmente agrupados em pares (E, R) contendo dados de entrada (E) que tem dados alvo de saída (R). Estes dados alvo de saída são os resultados que o algoritmo deve entender para a reprodução em cenários similares. Assim, após o treinamento do algoritmo, ou ele poderá deduzir certos resultados a partir dos dados de entrada que não estavam mapeados, ou pode-se afirmar que ele nunca teve o acesso durante o período do treinamento (MARSLAND, 2015).

#### 2.2.2.1 Técnicas de classificação

As técnicas de classificação consistem em atribuir vetores de entradas de dados (input) e verificar para qual das classes definidas durante o treinamento do algoritmo essas entradas se encaixam. Além de classificar as entradas, os algoritmos responsáveis pela resolução de problemas de classificação têm como objetivo definir qual é o limite entre cada classe, ou seja, o que estabelece com que a entrada *n* seja classificada como classe X e não como classe Y (MARSLAND, 2015).

Na Tabela 1, são citados e explicados brevemente os principais algoritmos utilizados na resolução dos problemas de classificação. Através disso, é possível verificar que eles não se limitam somente a isso (SARKER, 2021).

**Tabela 1 -** Algoritmos utilizados em Técnicas de Classificação

| ALGORITMOS | COMO FUNCIONA |
|---|---|
| *Naive Bayes* | O classificador assume que as características de uma classe não são relacionadas entre si, e baseia-se no Teorema de Bayes para calcular a probabilidade de cada classe. A classe com maior probabilidade é a escolhida (MAHESH, 2019). |
| Késimo Vizinho Mais Próximo (KNN) | Utiliza-se a Distância Euclidiana efetuada nos dados de treino para calcular a similaridade entre novos pontos de dados e classificá-los (SARKER, 2021). |
| Máquina de Vetores de Suporte (SVM) | Performa-se uma classificação linear e constrói-se um hiperplano contendo a maior distância dos dados de treinamento mais próximos entre as classes, definindo |

| | |
|---|---|
| | uma lacuna ao meio e indicando a separação das classes (SARKER, 2021). |
| Árvore de Decisão | Utiliza-se uma árvore de grafos para representar os eventos e escolhas.<br>A instância da classe a ser classificada passa pela árvore a partir do nó raiz e vai em direção aos nós folhas. Em nós de decisão, são levados em consideração os valores dos atributos da instância para a classificação final (MAHESH, 2019). |

Fonte: Elaboração própria

2.2.2.1 Técnicas de regressão

As Técnicas de Regressão consistem em métodos de AS que tentam predizer o resultado de uma variável contínua Y, tendo como base uma ou mais variáveis. Assim, enquanto houver problemas de classificação, vai existir a preocupação em definir a classe de um determinado dado de entrada, de modo que os problemas de regressão possibilitem a predição das variáveis dessas classes (SARKER, 2021). Diante disso, define-se que os modelos de regressão englobam certos tópicos de predição de eventos ou valores, sendo utilizados em diferentes situações, como por exemplo, na previsão financeira de crescimento, na estimativa dos custos de uma empresa, e na identificação de transações fraudulentas (SARKER, 2021). Na Tabela 2, são citados e explicados brevemente os principais algoritmos utilizados na resolução dos problemas de regressão (SARKER, 2021). Através disso, é possível verificar também que essa resolução pode ser utilizada em outros problemas.

**Tabela 2 -** Algoritmos utilizados em Técnicas de Regressão

| ALGORITMOS | COMO FUNCIONA |
|---|---|
| Regressão linear Simples | O relacionamento entre a variável dependente P (contínua) e a variável independente Q (contínua ou discreta) é linear, na qual é definida pela equação: $p = a + bq + e$ .....(3), adaptado de (SARKER, 2021). |
| Regressão linear múltipla | O relacionamento entre as variáveis continua sendo linear, mas a regressão múltipla permite que duas ou mais variáveis independentes Q, realizem a predição da variável dependente P. É definida pela equação: $p = a + b_1q_1 + b_2q_2 + \cdots + b_nq_n + e$ .....(4), adaptado de (SARKER, 2021). |

| | |
|---|---|
| Regressões polinomiais | O relacionamento entre as variáveis independente P e dependentes Q não é linear, e sim definida pelo enésimo grau de Q. É definida pela equação: $p = b_0 + b_1 q^1 + b_2 q^2 + b_3 q^3 + \cdots + b_n q^n + e$.....(5), adaptado de (SARKER, 2021) |

Fonte: Elaboração própria

### 2.2.3 Aprendizado Não Supervisionado

Nos algoritmos de Aprendizagem Não Supervisionados (ANS) não há rótulos ou categorização. Esses algoritmos definem os padrões e as estruturas de um conjunto de dados sem nenhum treinamento prévio, de modo que são deixados por conta para descobrir e apresentar estruturas interessantes nos dados disponibilizados (MAHESH, 2019). Assim, a ANS tem o objetivo de encontrar um aglomerado de dados (MAHESH, 2019).

#### 2.2.2.1 Técnicas de clusterização

As técnicas de clusterização são métodos utilizados para identificar e agrupar os pontos de data relacionados a um grande dataset. Estes grupos são chamados de clusters. O agrupamento é efetuado com o intuito de reunir coleções de dados que, de alguma forma, são similares entre si considerando uma ou mais variáveis, de modo que se diferem dos outros dados contidos em outros clusters (SARKER, 2021). Essas técnicas são utilizadas em determinadas situações, como por exemplo, na gestão de itens de um inventário agrupando seu conteúdo por similaridade e análise de dados de saúde. Na Tabela 3, são citados e explicados brevemente os principais algoritmos utilizados na resolução dos problemas de clusterização.

**Tabela 3 -** Algoritmos utilizados em Técnicas de Clusterização

| ALGORITMOS | COMO FUNCIONA |
|---|---|
| *K-Means* | Identifica-se o número *k* centróides e, em seguida, atribui cada ponto de dados ao cluster mais próximo, mantendo os centroides o menor possível (SARKER, 2021). |
| Clusterização hierárquica | Procura-se construir uma hierarquia de clusters. Pode ser Aglomerativa, na qual cada par de clusters é combinado em um novo cluster, começando da base e indo ao topo, ou pode ser Divisiva, na qual o cluster inicial é o do topo e move-se para baixo na hierarquia dividindo-se em dois a cada nível (SARKER, 2021). |

| | |
|---|---|
| Análise do Componente Principal (PCA) | Converte-se num conjunto de observações de variáveis possivelmente relacionadas em um conjunto de valores de variáveis linearmente não relacionadas, chamadas de componentes principais (SARKER, 2021) |

Fonte: Elaboração própria

2.2.2.1 Técnicas de associação

As técnicas de associação consistem no descobrimento de padrões baseados no relacionamento entre as variáveis. Os algoritmos majoritariamente utilizam declarações ”Se-Então” para a criação dessas associações (SARKER, 2021). Além disso, os algoritmos não utilizam necessariamente uma ordem relacionada as variáveis que aparecem em um registro do dataset como parâmetro (SARKER, 2021). Esses métodos podem ser utilizados, por exemplo, para entender o padrão de consumo dos clientes, tendo como base os itens escolhidos nas compras anteriores (OSMAN, 2019).

### 2.2.4 Aprendizado Semi-Supervisionado

Os algoritmos que fazem parte da classe de Aprendizado Semi-Supervisionado (ASS), combinam as técnicas de AS e ANS para a resolução de determinados problemas. Geralmente, esses algoritmos utilizam um conjunto de dados misto, contendo dados rotulados e também não rotulados (MAHESH, 2019). Por conta disso, as técnicas e algoritmos utilizados em problemas de ASS podem conter ambos os tipos de aprendizados citados acima, de modo que isso pode ocasionar o aumento ou a diminuição da utilização deles, conforme a necessidade do problema, com o objetivo final de fornecer um melhor resultado, de acordo com os dados rotulados do modelo (SARKER, 2021). Dentre os tipos de aprendizagem o ASS, esse é o menos popular, uma vez que suas aplicações são muito específicas e difíceis de serem implementadas (SARKER, 2021).

### 2.2.5 Aprendizado por Reforço

As Técnicas de Aprendizado por Reforço (AR) é uma abordagem que permite que as máquinas de software (agentes) definam como positiva ou negativa a resposta do ambiente em

que o algoritmo está sendo executado (SARKER, 2021). Ao final, o agente necessita experimentar várias técnicas e verificar qual a melhor baseada na recompensa em que o ambiente retorna (MARSLAND, 2015). Diferentemente dos dados disponibilizados em pares para o treinamento de algoritmos supervisionados, o par de dados de treinamento tem o formato (E, R) no aprendizado por reforço.

Diante disso, o algoritmo pode optar pela busca de diferentes técnicas para coletar mais informações, ou optar pela exploração mais aprofundada dos retornos de maiores valores já atingidos (MARSLAND, 2015). Assim, diferentemente do AS, o AR apenas informa a qualidade da resposta, e não a forma de como o algoritmo pode melhorá-la, ficando por sua conta a exploração de novas estratégias. Para esse tipo de aprendizado, as utilizações podem variar, mas geralmente, elas são utilizadas para a automação de sistemas robóticos, tarefas de direção autônoma e nos processos de produção e logística (MARSLAND, 2015).

## 3 MÉTODO

Esse estudo que está sendo realizado é de uma abordagem mista e natureza exploratória. Para a execução da pesquisa, foram analisados dados referentes às participações das equipes da Escola de TI do Centro Universitário de Maringá (UniCesumar) dos anos de 2012 até 2021. A Escola de TI faz parte do Projeto Integrador, uma matéria interdisciplinar obrigatória dos cursos de Bacharel em Engenharia de Software e Análise e Desenvolvimento de Sistemas da UniCesumar. O objetivo do Projeto Integrador é possibilitar aos alunos a implementação de um produto de software, desde a concepção inicial da ideia até a entrega final do produto. Para o processo de desenvolvimento, as equipes basearam-se na metodologia ágil *Scrum*, com *Sprints* de entrega de 2 semanas. O gerenciamento do projeto é feito no software *Redmine*, assim como a criação das estórias, tarefas, apontamento das horas trabalhadas em cada tarefa, divisão das tarefas nas *Sprints* de entrega e na definição de prioridade nas tarefas. No dataset disponibilizado, cada linha do arquivo refere-se a uma tarefa de uma determinada equipe, contendo 28 colunas originais.

Para a revisão bibliográfica inicial, foram estudados diversos artigos científicos que descreveram processos e características da mineração de dados, assim como suas aplicações e algoritmos de ML utilizados. Além disso, foram estudados os artigos relacionados a área de mineração de dados em projetos *Scrum*. Para a pesquisa destes artigos, foram utilizadas as seguintes palavras chave " '*data mining' scrum*" , com recorte temporal de 2015 - 2022. Nessa

seção a seguir, serão apresentados os procedimentos utilizados para a análise do dataset em questão.

## 3.1 COLETA E TRANSFORMAÇÃO DOS DADOS

O dataset foi disponibilizado pelos professores responsáveis. Para a transformação dos arquivos descritos na sequência, foram utilizados os arquivos da biblioteca Pandas da linguagem Python. Inicialmente foram disponibilizados 53 arquivos no formato *csv* que, em seguida, foram ligados a um arquivo final contendo todas as tarefas, totalizando 14672 linhas, com cada linha representando uma tarefa de uma equipe em específico. Primeiramente, foi realizada a verificação das linhas duplicadas baseada em todas as colunas (i.e seriam removidas as linhas que continham todos os valores iguais), porém nenhuma duplicação foi encontrada.

Posteriormente, as seguintes colunas foram eliminadas do dataset: #, Assignee, Updated, Parent task, Author, Category, Target version, Start date, Due date, Estimated time, Total estimated time, Spent time, Total spent time, % Done, Created, Closed, Last updated by, Related issues, Files, Pontos, TipoNaoConformidade, Sprint e Private. Essas colunas não foram levadas em consideração na análise dos dados por não apresentarem informações significativas que colaborariam com o objetivo final dessa pesquisa. A coluna Project, que continha o nome das equipes no formato ”escoladeti + ano + time”(por exemplo, escoladeti2012time01) foi transformada em uma nova coluna de nome Equipe contendo apenas o ano e o time. Além disso, uma das equipes que tinha o nome ’2018time04 - Reset Arcade’ teve seu valor alterado apenas para ’2018time04’. Na sequência, as colunas remanescentes foram renomeadas para facilitar a manipulação e o entendimento do dataset. A Figura 1 ilustra o processo de transformação das colunas, mostrando a origem a partir do dataset original. Nesse sentido, as colunas que não estão contempladas na tabela do meio foram descartadas.

## 3.2 TRATAMENTO DOS DADOS DE TEXTO

Para o processo de mineração dos dados do dataset, utilizou-se o software Orange como ferramenta principal. Na figura 2 é demonstrado o fluxo de trabalho realizado, que são compostos pelo carregamento, pré-processamento, aplicação da clusterização e visualização dos resultados no gráfico.

Primeiramente, o dataset com 14672 registros e 5 variáveis foi adicionado ao software como um documento de texto, e a coluna de Funcionalidade foi escolhida para ser a variável utilizada na análise do texto, como pode ser visto na Figura 3. Em seguida, foi realizado o pré-processamento dos dados. O pré-processamento consiste em 3 etapas principais: Transformação, Tokenização e Filtragem. Na transformação, todas as palavras foram alteradas para letras minúsculas e quaisquer resquícios de código *html* ou links *url* foram removidos. No processo de Tokenização, ocorreu a separação do texto em componentes menores, dividindo o grupo palavras e as pequenas sentenças. Ao final, foi realizada uma filtragem para garantir a retirada de verbos, preposições, palavrões, números e pontuações do texto final.

Em seguida, a saída de texto do pré-processamento foi enviada para o Saco de Palavras (do inglês, *Bag of Words*). Essa ferramenta tem o objetivo de tratar cada documento (funcionalidade) como uma coleção de palavras particulares. Assim, ela trata cada palavra de uma frase com uma importância similar. Ainda, a *Bag of Words* realiza a contagem de frequência dos termos presentes no texto de entrada. O padrão recomendado pela ferramenta, e o que foi utilizado para essa pesquisa foi a frequência sublinear, que calcula o logaritmo da contagem do número de ocorrências dos termos no texto.

**Figura 1**: Transformação das colunas

| Colunas originais |
|---|
| # |
| Project |
| Tracker |
| Status |
| Priority |
| Subject |
| Assignee |
| Updated |
| Parent task |
| Author |
| Category |
| Target version |
| Start date |
| Due date |
| Estimated time |
| Total estimated time |
| Spent time |
| Total spent time |
| % Done |
| Created |
| Closed |
| Last updated by |
| Related issues |
| Files |
| Pontos |
| TipoNaoConformidade |
| Sprint |
| Private |

⇒

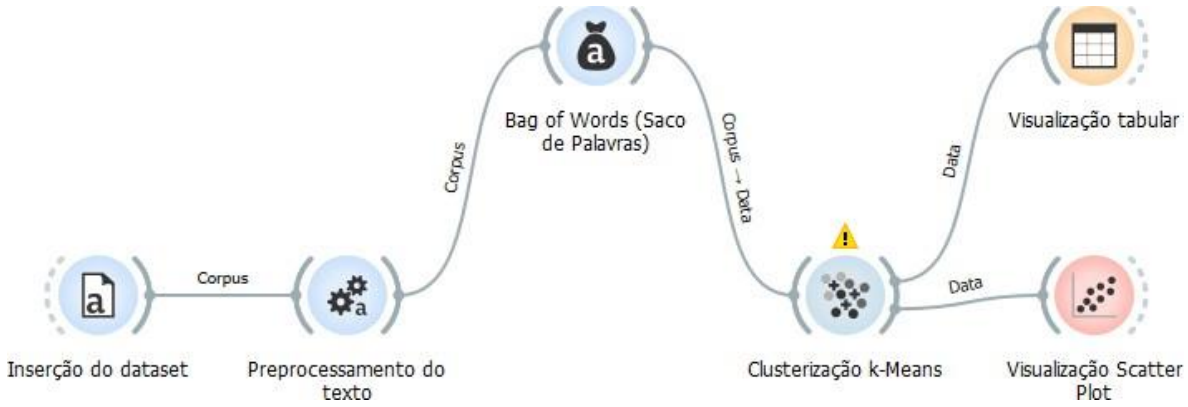
| Colunas após transformação |
|---|
| Tipo |
| Status |
| Prioridade |
| Equipe |
| Funcionalidade |

⇒

| Exemplo de registro do dataset |
|---|
| Task |
| Realizado |
| Normal |
| 2012time01 |
| Consulta de Item ao Acervo |

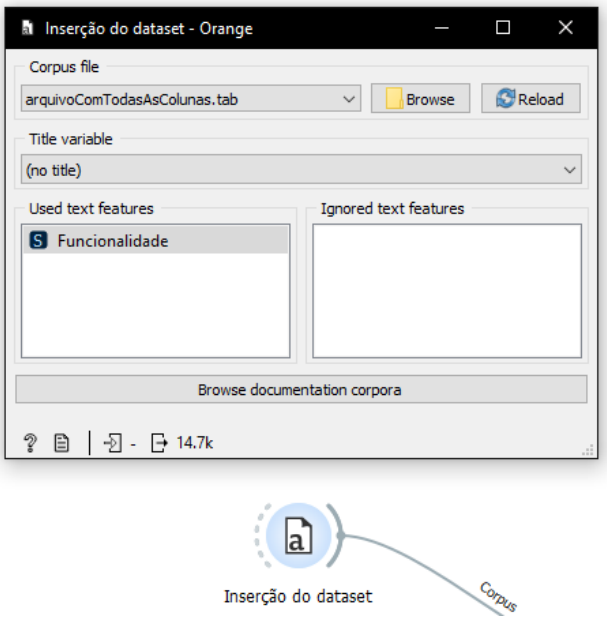
Fonte: Elaboração própria.

**Figura 2**: Fluxo de trabalho principal no software *Orange*




Fonte: Elaboração própria.

**Figura 3**: Inserção do dataset como documento de texto

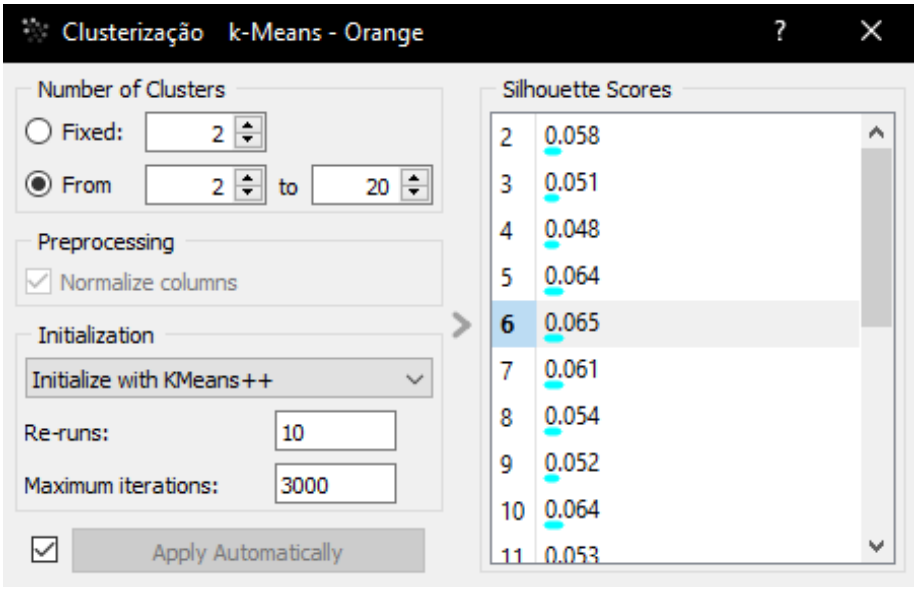



Fonte: Elaboração própria.

## 3.3 APLICAÇÃO DA CLUSTERIZAÇÃO

Nesta etapa, foi utilizado o algoritmo de clusterização k-Means. Para a seleção da quantidade de clusters utilizada pelo algoritmo, a ferramenta fez o cálculo baseado no coeficiente de Silhouette. Este coeficiente é calculado a partir da média da distância entre clusters, e pode ser calculado tanto pela Distância Euclidiana quanto pela Distância de Manhattan (GUPTA; PANDA, 2019). Na Figura 4 é possível verificar os valores de cada coeficiente com sua respectiva quantidade de clusters ao lado. Assim, na Figura 4 é indicado como é feita a inicialização do processo de clusterização. O primeiro centro do cluster é escolhido aleatoriamente, e os subsequentes são escolhidos entre os pontos restantes com probabilidade proporcional à distância quadrada do centro mais próximo. A quantidade de reexecuções do algoritmo foi mantida com o padrão de dez execuções a partir dos pontos iniciais, enquanto a quantidade máxima de iterações foi alterada do padrão de 300 para 3000. A quantidade de iterações foi aumentada com o intuito de estressar o algoritmo o suficiente para que ele atinja o resultado mais satisfatório baseado no coeficiente indicado. Ao final da clusterização, o número resultante ideal de clusters encontrados foram de seis com um coeficiente igual a 0.065.

**Figura 4**: Configuração dos parâmetros do método k-Means




Fonte: Elaboração própria.

## 3.4 VISUALIZAÇÃO DOS RESULTADOS

Após a etapa da clusterização, os resultados foram enviados para um Scatter Plot. O Scatter Plot fornece uma visualização do gráfico de dispersão bidimensional. Os dados são exibidos como uma coleção de pontos, cada um tendo o valor do atributo do eixo *x* determinando a posição no eixo horizontal, e o valor do atributo do eixo *y* determinando a posição no eixo vertical. No contexto desta pesquisa, o eixo *x* foi alimentado com os Clusters encontrados, enquanto o eixo *y* foi alimentado com os tipos das tarefas. Dessa forma, as diferentes cores visualizadas na dispersão da Figura 5 representam os status das tarefas, e cada ponto dispersado no gráfico refere-se à uma funcionalidade original do dataset.

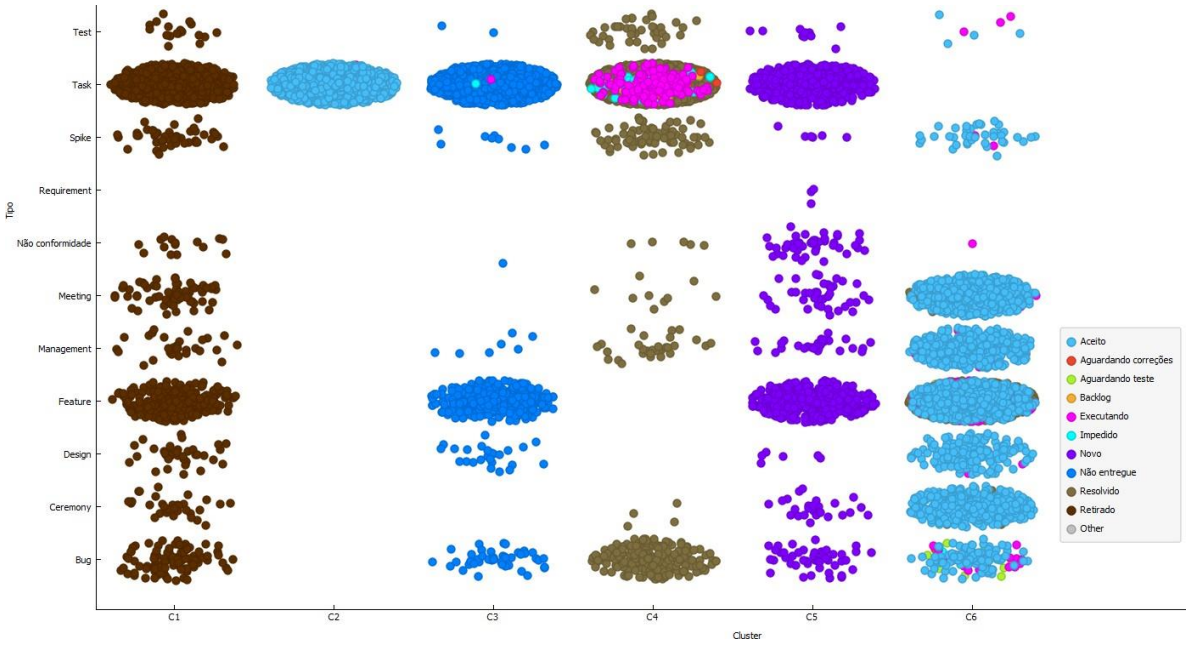
# 4 RESULTADOS

Nessa seção será apresentado os resultados após a aplicação da clusterização. Foram encontrados sete clusters, a distribuição das funcionalidades em relação aos clusters, e ao tipo das tarefas que foram representadas em um gráfico bidimensional de dispersão na Figura 5. Ainda, levando em consideração o Gráfico 1, é mostrado a distribuição da quantidade tarefas

por status. Os status “Testando”, “Code review” e “Rejeitado” tiveram suas quantidades anexadas e definidas com o status ”Other”.

Partindo do pressuposto de que, o processo do desenvolvimento de software da Escola de TI deva seguir os princípios da metodologia ágil, foram levantadas suposições baseadas nos resultados encontrados após a clusterização. Estas suposições serão apresentadas nesta seção e discutidas posteriormente.

**Figura 5**: Representação das funcionalidades encontradas relacionadas aos tipos de tarefas



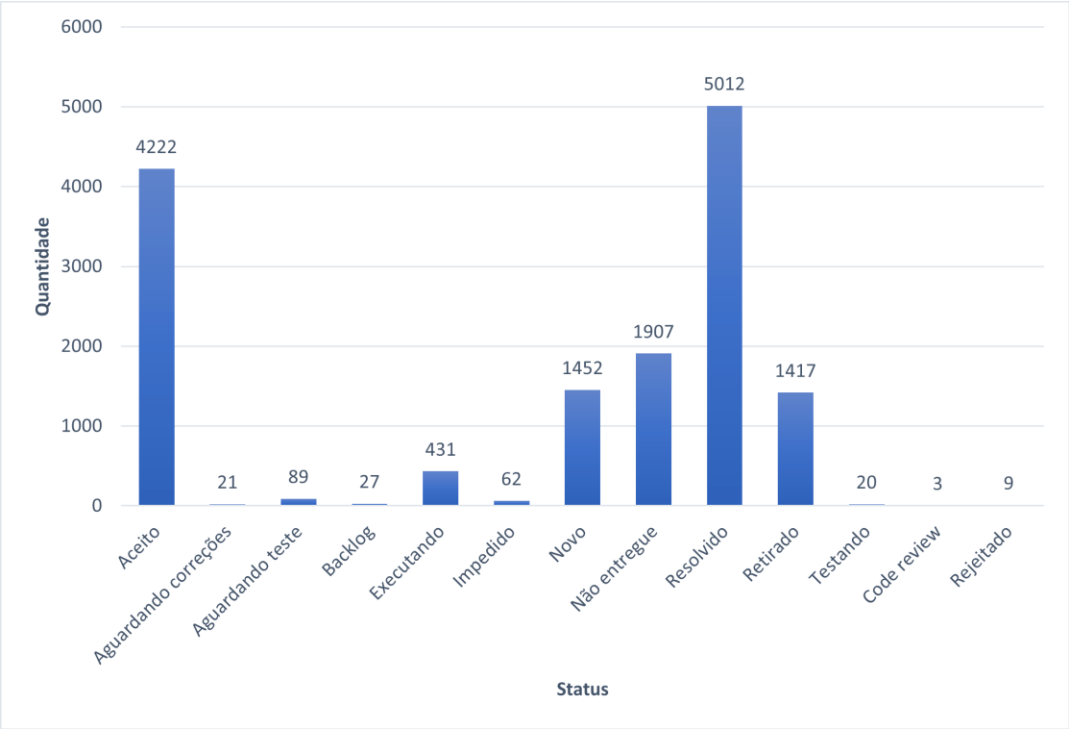

Fonte: Elaboração própria.

## 4.1 SUPOSIÇÃO #1: TAREFAS COM STATUS EXECUTANDO FORAM DESCONTINUADAS PELAS EQUIPES

A Figura 6 refere-se à uma ampliação da Figura 5, focando em um ponto principal: a quantidade de tarefas com status ’Executando’, juntamente com a disparidade dos tipos de tarefas. Essa disparidade refere-se aos tipos de tarefa como *Feature, Task, Bug, Design e Ceremony* para ainda obterem tarefas com esses status mesmo após a finalização do projeto. Na Figura 7, é evidenciado um foco maior no C6 e em tarefas do tipo *Feature*. A Figura 8 apresenta o mesmo exposto acima para o C4 e tarefas do tipo *Task*. Considerando o cluster com o tipo *Task* demonstrado na Figura 8, de um total de 3946 tarefas, 157 tem o status definido como executando, cerca de 3,98%. Da mesma forma, para as tarefas do tipo Feature, de um total de

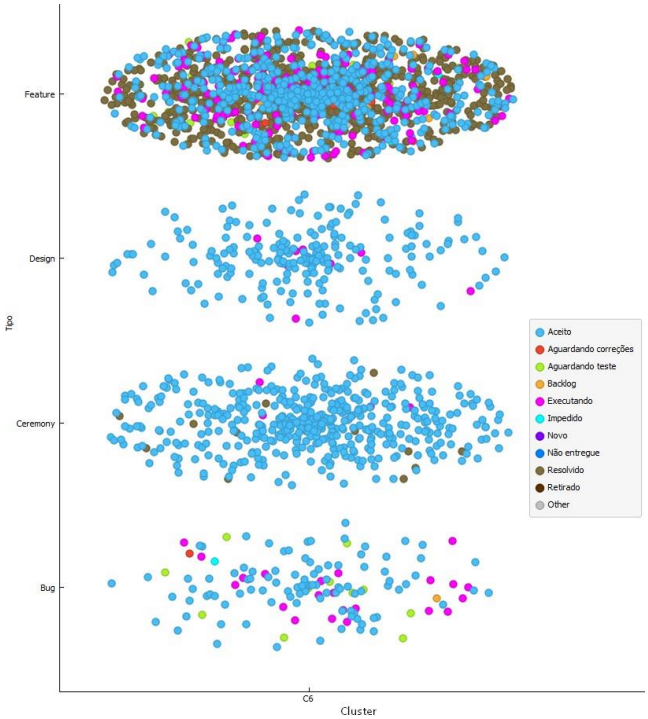
1722 tarefas, 184 apresentaram o status executando, cerca de 10,7% do total. Do total de 14672 tarefas do projeto, 431 estão com este status, correspondendo a 2,94%.

**Gráfico 1 -** Gráfico representando a distribuição das quantidades de tarefas por status



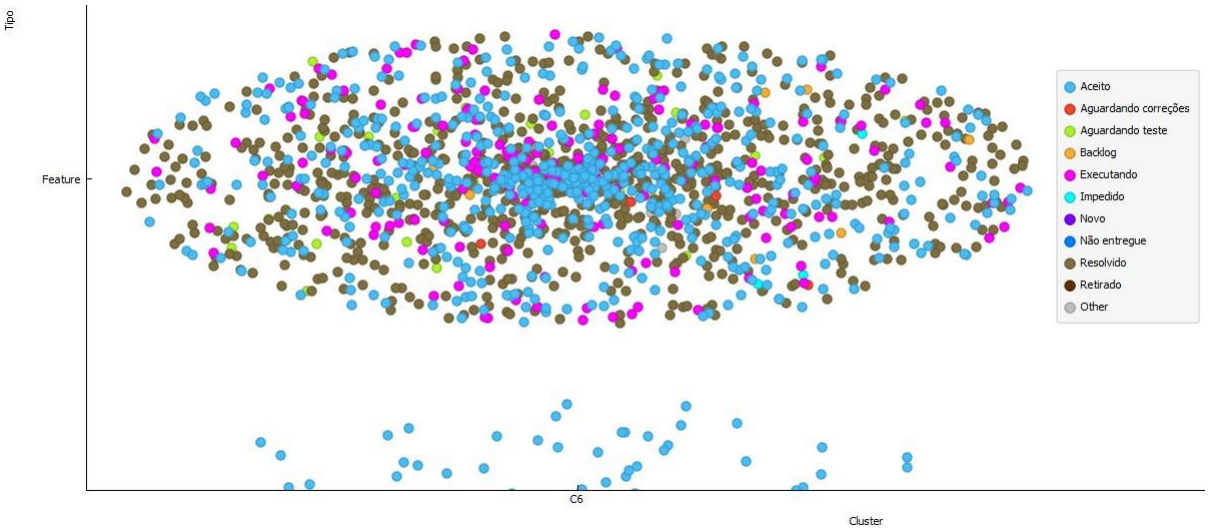

Fonte: Elaboração própria.

**Figura 6**: Representação das funcionalidades encontradas relacionadas aos tipos de tarefas



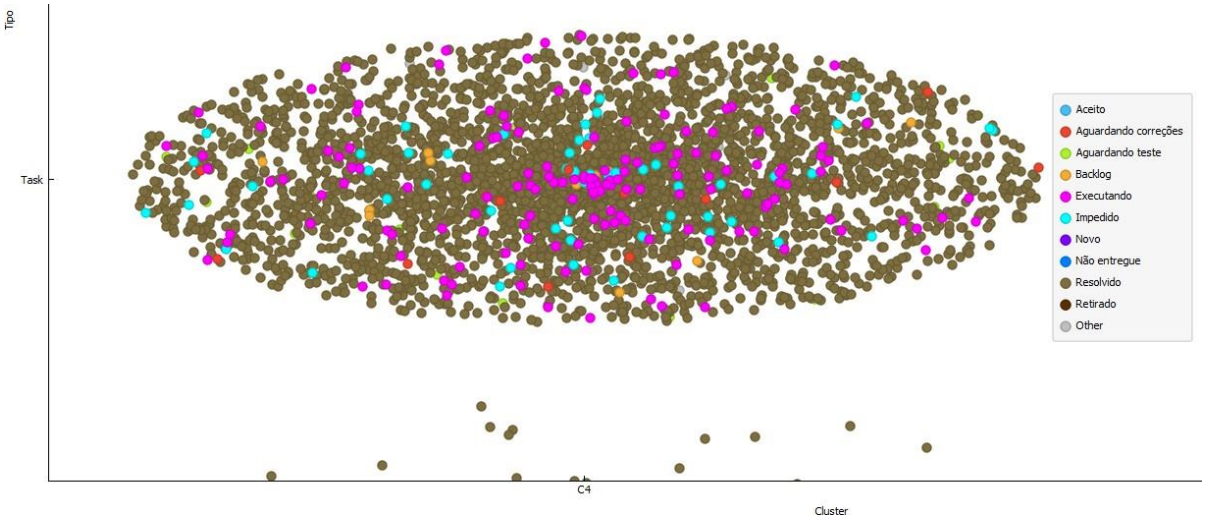

Fonte: Elaboração própria.

**Figura 8**: Combinação de análise do cluster C6 com tarefas do tipo *Feature*




Fonte: Elaboração própria.

**Figura 9**: Combinação de análise do cluster C4 com tarefas do tipo *Task*




Fonte: Elaboração própria.

## 4.2 SUPOSIÇÃO #2: O LEVANTAMENTO DE REQUISITOS REALIZADO PELAS EQUIPES OCASIONA UMA ALTA GRANULARIDADE

Na Figura 5, é possível perceber que muitas tarefas foram retiradas no Cluster C1. Na Figura 10 são exemplificadas algumas tarefas que não foram executadas e que pertencem ao cluster C1. Das 14672 tarefas analisadas, a quantidade de tarefas retiradas foram de 1417, correspondendo cerca de 9% do total.

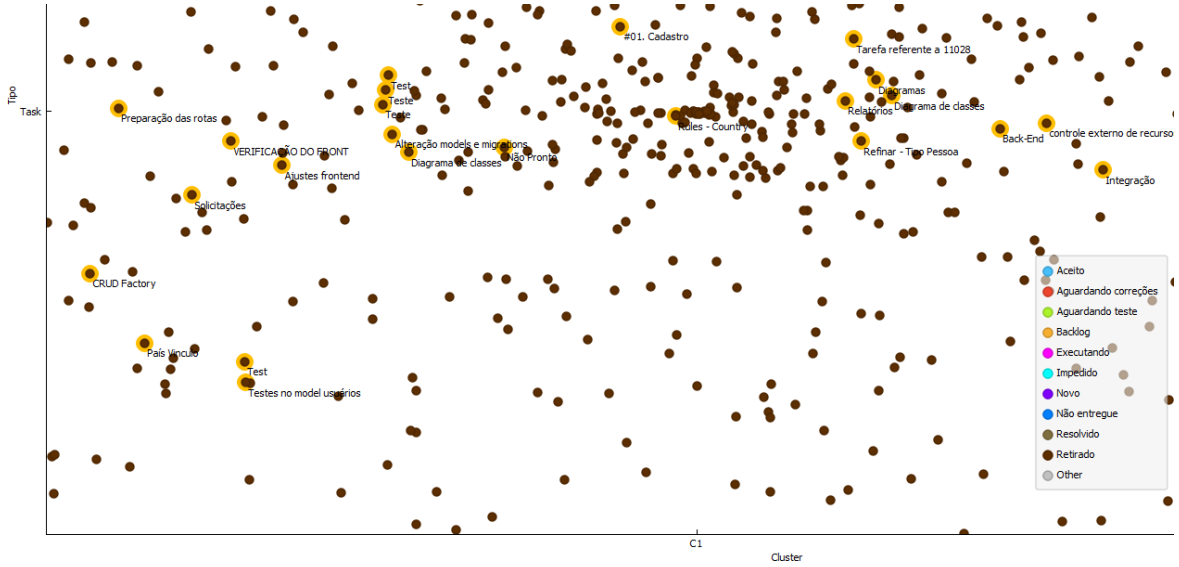
Além disso, ao analisar este mesmo cluster C1, verificou-se que a nomenclatura utilizada na criação das tarefas não seguia um padrão, sendo a maioria delas assertivas. A Figura 11 demonstra alguns exemplos de tarefas não assertivas encontradas.

**Figura 10**: Exemplos de tarefas retiradas

| | Funcionalidade |
|---|---|
| 1 | Criar Classe "Tipo" no pacote pessoa |
| 2 | Criar Controller currículo |
| 3 | Criar Controller empresas |
| 4 | Criar Models da currículo |
| 5 | Criar Models da empresas |
| 6 | Criar abas de situação e advertência no cadastro de aprendiz (front) |
| 7 | Criar ação de aumentar velocidade do rolo |
| 8 | Criar ação de diminuir velocidade do rolo conforme a quantidade de grãos adicionados. |
| 9 | Desenvolvimento de Rotas |
| 10 | Desenvolvimento feed |
| 11 | Mobile - Ajuste visual - Escalação |
| 12 | Mobile - Ajuste visual - Jogador pro |
| 13 | Mobile - Ajuste visual - Outros usuários |
| 14 | Mobile - Ajuste visual - Perfil |
| 15 | Mobile - Ajuste visual Dashboard |
| 16 | PADRONIZAÇÃO DE FORMULÁRIOS E BOTÕES |
| 17 | Refatoração Família |
| 18 | Refatoração Pessoa |
| 19 | Solicitar serviço ou produto |
| 20 | Solicitações |
| 21 | Tela Cadastro de Usuário. |
| 22 | Tela Criar nova conta de usuário |
| 23 | Tela de Cadastro [Em PSD] |
| 24 | Tela de Despesas |
| 25 | Tela de Landing Page |
| 26 | Tela de Login |
| 27 | Tela de chegada |
| 28 | Tela de enviar despesa |
| 29 | Tela de notificação |
| 30 | Tela de proposta |
| 31 | Tela visualizar anuncio |
| 32 | Teste manter perfil de acesso |
| 33 | Teste manter perfil de usuario |
| 34 | Teste manter usuário |
| 35 | Validação com Jquery |
| 36 | Validação de cnpj |
| 37 | Validação de login e token. |
| 38 | Validação dos campos CPF/CNPJ via JQuery |

Fonte: Elaboração própria.

**Figura 11**: Exemplos de tarefas retiradas não assertivas

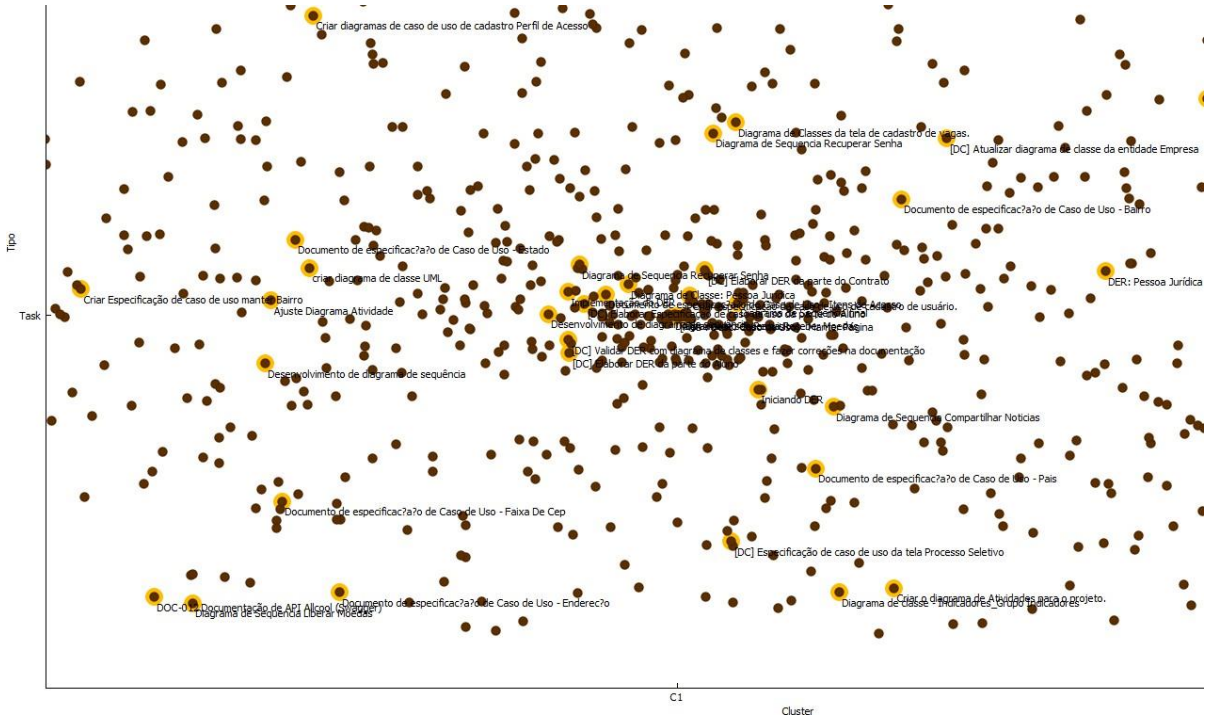



Fonte: Elaboração própria.

## 4.3 SUPOSIÇÃO #3: O LEVANTAMENTO DE REQUISITOS REALIZADO PELAS EQUIPES OCASIONA A RETIRADA, E A NÃO ENTREGA DE TAREFAS VOLTADAS AO PROCESSO DE DOCUMENTAÇÃO DO PROJETO

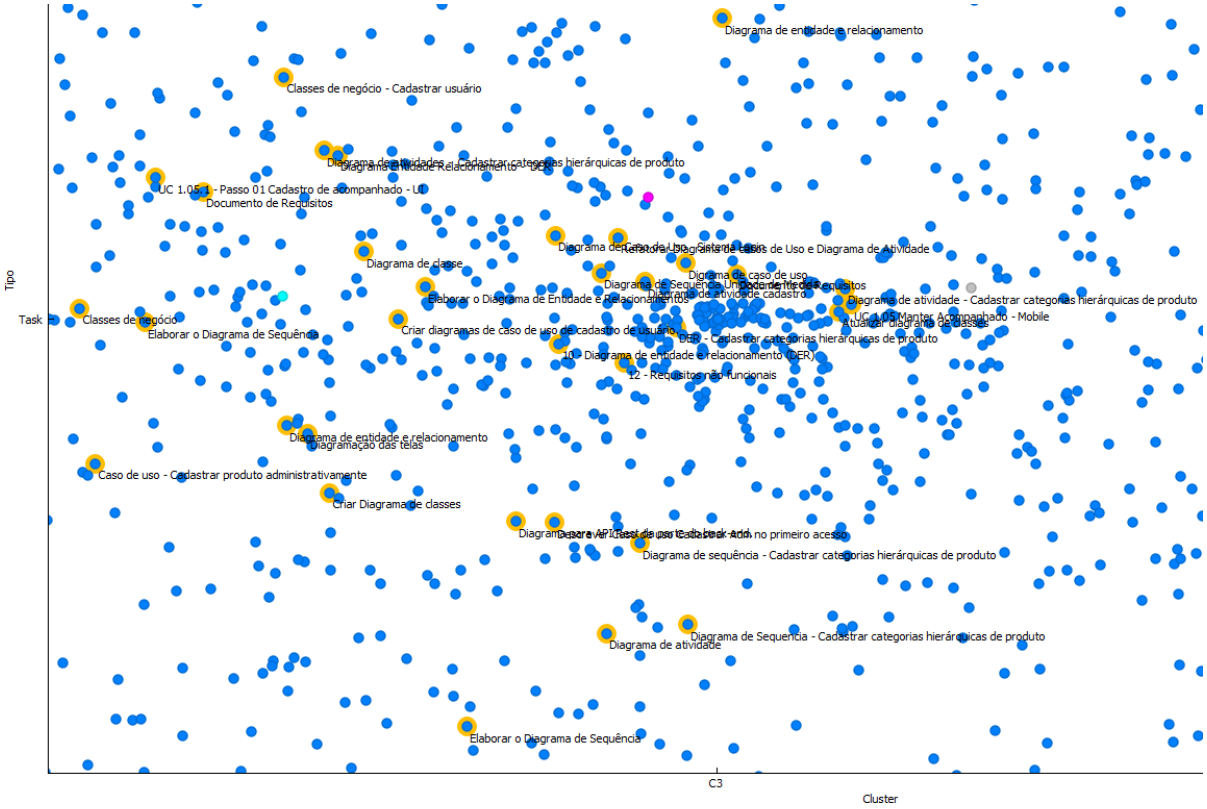
Seguindo o mesmo raciocínio indicado na suposição anterior, ao verificar a Figura 12, percebe- se que houve destaque nas tarefas voltadas a documentação do projeto. Além do mais, ao realizar uma filtragem com a expressão regular: ”diagrama|documentação|DER|der|MER|mer|doc|Diagrama|Mer|Der|Doc|DOC”, das 782 tarefas retiradas, e a do tipo *Task*, 98 eram relacionadas à documentação. Ademais, a Figura 13, referente ao cluster 3, também demonstrou exemplos de tarefas relativas à documentação que foram retiradas. Similarmente, a filtragem com a expressão regular revela que, das 1414 tarefas Não Entregues, e a do tipo *Task*, 156 eram relacionadas à documentação. Dessa forma, conclui-se que cerca de 11,5% das atividades retiradas ou não entregues para o tipo *Task*, são relacionadas com a documentação do projeto.

**Figura 12**: Exemplos de tarefas de documentação retiradas




Fonte: Elaboração própria.

**Figura 13**: Exemplos de tarefas e documentações não entregues




Fonte: Elaboração própria.

## 5 DISCUSSÕES

Após a análise dos resultados obtidos nas tarefas de status executando, verificou-se que do total de 14672 tarefas, quase 3% (431) não foram definidas como finalizadas pelas equipes. Por outro lado, percebe-se que cerca quase 11% das tarefas do tipo Feature ainda estão com este status, que é uma condição incomum para a atual situação (finalizada) do projeto. Através disso, levanta-se a suposição de que, na verdade, estas tarefas não foram finalizadas pelas equipes e não tiveram seu status alterado. Como é defendido em Almeida et al. (2019), a comunicação é um processo chave para as equipes de tecnologia, e a colaboração constante entre os participantes é essencial para o processo. Diante disso, levanta-se a hipótese de que houve uma falta de organização e comunicação entre alguns times, e que estas tarefas não foram totalmente finalizadas pelas equipes. Além disso, a existência de uma falha no processo de socialização do time na parte de facilitar a discussão formal e informal, no trabalho em grupo e no engajamento da equipe (PATEL et al., 2021) pode ter corroborado para a depreciação do processo de controle das tarefas.

Dessa forma, pode-se dizer que é papel da faculdade fazer as intervenções durante os desenvolvimentos dos projetos, como também auxiliar as equipes a melhorar seu autogerenciamento e fornecer *feedbacks* constantes sobre o andamento do projeto, levando em consideração os pontos aqui discutidos. Para as empresas, recomenda-se ferramentas de gestão do conhecimento para evitar este problema e também facilitar a socialização e a integração de novos integrantes nas equipes. Nesse cenário, essas ferramentas como o Storytelling, que tem o objetivo de compartilhar conhecimento e aproximar os integrantes com Vídeos Comunicativos e Webinars (YOUNG, 2022) são uma boa forma de garantir a boa intercomunicação dentro da equipe.

Para Pressman (2009), a engenharia de requisitos é uma parte fundamental na estruturação de um projeto de software. Sem ela, o projeto perde um dos principais pilares de sustentação, e tende a falhar. Sendo assim, ao remeter à suposição de dois e verificar que cerca de 9% das atividades estão sendo retiradas, é necessário ter um pouco de atenção para entender o real motivo para isso acontecer. Primeiramente, analisando a nomenclatura de algumas tarefas, é possível perceber que, a alta granularidade no momento de criação das tarefas faz com que grande parte delas ficassem contidas em tarefas maiores, mas que só foram percebidas no momento do desenvolvimento. Por exemplo, é citado as tarefas de índice 7 e 8 da Figura 10: “Criar ação de aumentar a velocidade do rolo” e “Criar ação de diminuir a velocidade do rolo conforme a quantidade de grãos adicionados.” respectivamente. Pode-se retirar desse exemplo que, na verdade, estas funcionalidades foram desenvolvidas em uma tarefa mais abrangente, porém também não foram descartadas totalmente do projeto.

Além disso, considerando a nomenclatura das tarefas, podemos dizer que outro ponto é levantado. Tarefas com nomenclaturas não assertivas podem ser encontradas na Figura 10. Por exemplo, podem ser citadas algumas tarefas como: ”#01. Cadastro”, ”Integração”, ”Test”, ”Solicitações” e ”País Vinculo” que, mesmo considerando o contexto do projeto, não são possíveis de serem avaliadas facilmente e muito menos entender o real motivo da sua implementação. Estes são apenas alguns exemplos de tarefas não assertivas encontradas, mas na totalidade, ainda existem uma quantidade sustentável para fazer tal hipótese.

Diante disso, argumenta-se que o levantamento dos requisitos realizado pelas equipes não conseguiu ser precisa em certas ocasiões, ou seja, de forma que não foi possível definir adequadamente a separação das tarefas, e muito menos da assertiva, uma vez que muitas tarefas possuem nomes inadequados, que não descrevem de forma sucinta o objetivo da tarefa, o que acaba dificultando o entendimento do projeto como um todo.

Sendo assim, a suposição três segue um raciocínio similar à suposição dois. Segundo Sommerville (2015), a documentação formal de um projeto que segue a metodologia ágil deve descrever o sistema, contendo seus limites e funcionalidades e manter-se atualizada conforme o andamento do projeto de software. Por isso, após analisar os resultados obtidos relacionados às tarefas de documentação, verifica-se que 11,5% de todas as tarefas retiradas ou não entregues do dataset estão relacionadas à documentação dos projetos. As Figuras 12 e 13 exemplificam as tarefas de documentação que foram retiradas ou não entregues, respectivamente. Ao visualizar alguns exemplos, nota-se que ações importantes de documentação, como a criação de diagramas de classe, diagrama de caso e de uso, e a especificação de caso e de uso, são consideradas retiradas ou não entregues. Dessa forma, podem ser levantados alguns questionamentos como: “Por que essas tarefas não foram feitas?”, “Estas tarefas foram contempladas em outras tarefas maiores?”, “Qual foi o impacto da retirada ou não entrega dessas tarefas para o projeto?”. Essa métrica conduz uma linha de argumentação que vai de encontro com a proposta feita pela metodologia ágil na engenharia de software acerca da documentação. Como é mostrado por Almeida et al. (2019), a falta de documentação é um dos maiores desafios encontrados na gestão do conhecimento, e essa suposição corrobora para essa afirmação. Por isso, pode-se argumentar que a especificação de requisitos realizada pelas equipes não foram o suficiente para suprirem as necessidades do projeto, uma vez que provavelmente houve refacção do trabalho na documentação. Para às empresas, a utilização de técnicas como Webinars e vídeos colaborativos, e esquemas de Mentor/Mentorado são apenas alguns dos métodos indicados pela Organização Asiática de Produtividade (APO) (YOUNG, 2022) que podem facilitar o processo de admissão e integração de novas pessoas.

## 6 CONSIDERAÇÕES FINAIS

Nessa pesquisa foi proposta uma outra forma de analisar o processo de desenvolvimento de software ágil por meio da técnica de agrupamentos das funcionalidades provenientes dos dados históricos de uma base de ensino de desenvolvimento de software. Os dados analisados compõem 9 anos (2012 - 2021) de registros referentes aos projetos de desenvolvimento ágil de equipes da matéria de Projeto Integrador da UniCesumar. O trabalho utilizou a clusterização para agrupar as funcionalidades similares do dataset, e analisar as informações encontradas. Ao final, foi concluído que o processo usado é útil para analisar os

dados de desenvolvimento de software, e além do mais, foi possível extrair informações relevantes sobre a base de dados analisada. Constatou-se três suposições finais que foram discutidas, considerando os conhecimentos bem fundamentados da engenharia de software.

Portanto, para futuras pesquisas, é possível utilizar este trabalho como ponto de partida para realizar um processo de verificação e analisar outros projetos de desenvolvimento de software, com o mesmo objetivo de encontrar determinadas informações relevantes. Logo, recomenda-se a utilização de outras técnicas de mineração de dados, assim como outras fontes de dados, para que o intuito seja verificar sua utilização em projetos ágeis e que ocorra a expansão de conhecimento nessa área de estudos.

**REFERÊNCIAS**

ALMEIDA, F.; MIRANDA, E.; FALCÃO, J. **Challenges and facilitators practices for knowledge management in large-scale scrum teams.** ***Journal of Information Technology Caseand Application Research***, Routledge, v. 21, p. 90–102, 2019. ISSN 23336897.

ALPAYDIN, E. ***Introduction to Machine Learning***. The MIT Press, 2020.

APENKO, S. N.; ROMANENKO, M. A. **Knowledge management in agile teams of flexible enterprise projects**. 2019.

ASSUNCAO, R. ***Fundamentos Estatísticos de Ciência dos Dados voltado para aplicações***.[S.l.]: Pu, 2017. ISBN 9788550803906.

CWEI, L. J.; SAHOO, N.; SRIVASTAVA, G.; DING, W**. Introduction to the special issue on pattern-driven mining, analytics, and prediction for decision making, part 1.** ***ACM Transactionson Management Information Systems***, v. 13, n. 1, p. 1–3, 2022.

FAWCETT, T.; PROVOST, F. ***Data Science para Negócios: O que você precisa saber sobre mineração de dados e pensamento analítico de dados***. Alta Books, 2018. ISBN 9788550803906.Disponível em: <https://books.google.com.br/books?id=1rZwDwAAQBAJ>.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. ***From Data Mining to Knowledge Discovery in Databases)*** *(© AAAI)*. 1996. Disponível em: <www.ffly.com/>.

GUPTA, T.; PANDA, S. P. **Clustering validation of clara and k-means using silhouette dunnmeasures on iris dataset**. In: *2019 International Conference on Machine Learning, Big Data,Cloud and Parallel Computing (COMITCon)*. 2019. p. 10–13.

ISLAM, M. S.; HASAN, M. M.; WANG, X.; GERMACK, H. D.; NOOR-E-ALAM, M. ***A systematic review on healthcare analytics: Application and theoretical perspective of data mining*****.** MDPI, 2018.

KARKI, S.; RANJITKAR, H. S. ***Comparison of A*, Euclidean and Manhattan distance usingInfluence Map in Ms***. ***Pac-Man***. 2016. Disponível em: <www.bth.se>.

KIYAK, E. O. **Data mining and machine learning for software engineering**. In: BIRANT, D. (Ed.). *Data Mining*. Rijeka: IntechOpen, 2020. cap. 8. Disponível em: <https: //doi.org/10.5772/intechopen.91448>.

LUZGIN, V. A.; KHOLOD, I. I. **Overview of mining software repositories**. In: *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering(EIConRus)*. 2020. p. 400–404.

MAHESH, B. **Machine learning algorithms -a review**. 01 2019.

MARSLAND, S. ***Machine Learning: An Algorithmic Perspective, Second Edition*****.** CRC Press, 2015. (Chapman & Hall).

OSMAN, A. S. ***Data Mining Techniques: Review***. 2019. Disponível em: <https: //www.educba.com/7-data->.

PADILHA, V. M.; ANTONIAZZI, R. L.; SCHUCH, R. R. ***MINERAÇÃO DE DADOS PARA PREVISÃO DE NUTRIENTES NO SOLO***. 2022.

PATEL, P.; RAMMAL, H. G.; FERREIRA, J. J.; PRIKSHAT, V. **Knowledge management, sharing and transfer in cross-national teams and the remote management of team members: the onsite-offshore phenomenon of service emnes**. *Journal of Global Mobility*, Emerald GroupHoldings Ltd., v. 9, p. 574–590, 11 2021. ISSN 20498802.

PRESSMAN, R. ***Engenharia de Software: Uma abordagem profissional.*** McGraw Hill Brasil, 2009. ISBN 9788580550443. Disponível em: <https://books.google.com.br/books?id=y0rH9wuXe68C>.

RAUTENBERG, S.; CARMO, P. **Big data e ciencia de dados.** *Brazilian Journal of InformationScience*, v. 13, p. 56–67, 03 2019.

ROMERO, C.; VENTURA, S. **Educational data mining and learning analytics: An updated survey.** *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*,Wiley-Blackwell, v. 10, 5 2020. ISSN 19424795.

SARKER, I. H. **Machine learning: Algorithms, real-world applications and research directions**.
*SN Computer Science*, Springer, v. 2, 5 2021. ISSN 26618907.

SELAMAT, S. A. M.; PRAKOONWIT, S.; KHAN, W. **A review of data mining in knowledgemanagement: Applications/findings for transportation of small and medium enterprises**. *SN*
*Applied Sciences*, v. 2, n. 5, 2020.

SOMMERVILLE, I. ***Software Engineering*. 10th. ed.** Pearson, 2015. ISBN 0133943038.

WANG, Y**. Automatic management analysis of computer software engineering project basedon data mining**. Association for Computing Machinery, 2021. p. 910–914. ISBN 9781450390422.

YOUNG, R. ***Knowledge management: Tools and techniques manual***. Asian Productivity Organization, Mar 2022. Disponível em: <https://www.apo-tokyo.org/publications/ knowledge-management-tools-and-techniques-manual/>.