UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE SISTEMAS DE INFORMAÇÃO

VINÍCIUS ARALDI LUNARDI FARIAS

Um modelo de recomendação de métricas ágeis para organizações de software

Florianópolis
2024

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE SISTEMAS DE INFORMAÇÃO

VINÍCIUS ARALDI LUNARDI FARIAS

Um modelo de recomendação de métricas ágeis para organizações de software

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação como requisito para obtenção do grau de Bacharel em Sistemas de Informação, sob orientação do Prof. Dr. Jean Carlo Rossa Hauck.

Florianópolis
2024

RESUMO

A definição das métricas necessárias para auxiliar gerentes de projetos em contextos ágeis no gerenciamento dos seus projetos tende a ser uma tarefa árdua e muitas vezes secundária, uma vez que existem diversas métricas não catalogadas e descentralizadas sendo utilizadas de forma customizada por cada time de acordo com suas necessidades. A Inteligência Artificial tem sido utilizada no desenvolvimento de sistemas de recomendação para as mais diferentes aplicações, desde filmes e músicas em plataformas de streaming a novas compras em portais de vendas de produtos. Assim, o presente trabalho trata da documentação do desenvolvimento de um modelo de recomendação de métricas ágeis, utilizando um *dataset* de métricas criado através de pesquisas na área acadêmica, entrevistas e *surveys* com gerentes de projeto e pesquisadores. A recomendação de métricas se dá por meio um sistema web, em que o usuário responde a perguntas referentes ao contexto organizacional, recebendo como resultado da recomendação do modelo de *Machine Learning* métricas que se encaixam em suas necessidades. Os testes iniciais do sistema de recomendação, bem como da interface web indicam que o que foi desenvolvido atende os requisitos apresentados.

Palavras-chave: Machine Learning, Métricas, Métodos Ágeis, Gestão de Projetos, Gerência de Projetos, Sistema Web, Dataset, Sistemas de Recomendação

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

SUMÁRIO

1. INTRODUÇÃO

No contexto de gerenciamento de projetos, métricas de software são “características de um sistema de software, documentação de sistema ou processo de desenvolvimento que pode ser objetivamente medido”, que têm o objetivo de mensurar a qualidade do software e dos processos de desenvolvimento de software, a fim de aumentar a qualidade e confiabilidade nos processos de desenvolvimento (SOMMERVILLE, 2019). Além de métricas, o processo de Medição de Software consiste em catalogar dados “relativos aos produtos desenvolvidos e aos processos implementados na organização e em seus projetos”, com objetivo de auxiliar na tomada de decisão dos gerentes de projeto para atingir os objetivos organizacionais. Medições podem ser divididas em medidas básicas ou derivadas. Medidas básicas são medidas que podem ser vistas de maneira isolada e independentes de outras medidas com base em atributos únicos, enquanto medidas derivadas obtém informações de outros dados e podem ser combinações de duas ou mais medidas. As análises das medidas básicas e derivadas oferecem indicadores, que são estimativas e avaliações que servem de base para análise das métricas e tomada de decisão (SOFTEX; MPS.B, 2016).

Métodos ágeis surgiram como uma resposta da indústria de desenvolvimento de software para se adaptarem às constantes mudanças e oportunidades de mercado. Assim, softwares desenvolvidos utilizando métodos ágeis são entregues e incrementados constantemente, e não desenvolvidos de forma única (SOMMERVILLE, 2019). Um das principais características de softwares desenvolvidos utilizando métodos ágeis é o incremento constante de funcionalidades em diversas versões, de 2 a 4 semanas de duração em cada lançamento, com a participação dos *stakeholders* dando *feedbacks* rápidos sobre a evolução dos requisitos do sistema (SOMMERVILLE, 2019). Os principais métodos ágeis existentes hoje são SCRUM, Extreme Programming (XP), Kanban, dentre outros (CARDOZO, 2020).

Na área de gerência de projetos, as métricas geralmente são utilizadas de forma customizada para atender as necessidades de cada time, uma vez que as métricas e medições são fundamentais para o auxílio à tomada de decisão e alcance dos objetivos dos times e organizações (SOMMERVILLE, 2019). Sua intenção, entretanto, é feita de maneira descentralizada na maioria das vezes, com cada gerente de projeto buscando em lugares diferentes métricas que se encaixem em suas necessidades. Devido a essa descentralização, muitas destas métricas podem trazer resultados que não condizem com as necessidades do time, quando não observadas

atentamente. Isso leva o gerente de projeto a tomar decisões que não irão ajudar seu time a alcançar seus objetivos ou até mesmo a ignorar certas métricas, que não agregam valor algum à equipe.

A área de Inteligência Artificial (IA) é uma área da computação que atua com sistemas que executam tarefas e tomam decisões de forma adaptativa, baseando-se nos dados que foram coletados previamente Este conceito foi cunhado em 1956 por John McCarthy durante um evento realizado por ele no Dartmouth College, porém o tema já havia sido abordado antes por Allan Turing em 1950 em seu artigo que deu origem ao Teste de Turing (ROCHA, 2022). Aprendizado de Máquina (*Machine Learning,* ML) são modelos de Inteligência Artificial que se caracterizam pela análise de um conjunto de dados processados por algoritmos, que geram um modelo com capacidade de explicar ou representar os dados de acordo com os aspectos requeridos (Brunialti *et al*., 2015). Sistemas de Recomendação (SR) baseados em IA e *Machine Learning* segundo Mendes; Machiavelli; Gusmao (2019), são capazes de recomendar materiais que mais se aproximam das necessidades de seus usuários; de acordo com Brunialti *et al*. (2015) SRs tem como objetivo “sugerir itens de forma a satisfazer sob algum aspecto as necessidades de um usuário”. Hoje, empresas como Netflix e Youtube utilizam IA para recomendações de vídeos, filmes e séries para cada usuário, baseado em suas preferências previamente coletadas (ROCHA, 2022).

Desta forma, o objeto de estudo deste trabalho é a recomendação de métricas ágeis, catalogadas por meio de pesquisas bibliográficas e via entrevistas e *surveys* com gerentes de projetos e pesquisadores da área. Com essa recomendação, gerentes de projeto que ainda não utilizam métricas em seus projetos ou que não obtém os benefícios esperados poderão obter métricas que já foram validadas e catalogadas, além de servirem às suas necessidades, facilitando assim a análise de resultados dos seus projetos.

1.1. Objetivos

Nesta seção estão descritos os objetivos deste Trabalho de Conclusão de Curso.

1.2. Objetivos Gerais

Desenvolver um modelo de IA para recomendação de métricas ágeis, que auxilie gerentes de projeto a encontrarem métricas que serão utilizadas em seus times de acordo com suas necessidades. Para esse fim, é necessária a catalogação e classificação destas métricas em um dataset, criado por meio de pesquisas bibliográficas e em entrevistas e *surveys* com gerentes de projeto e pesquisadores da

área. A partir desse dataset, um modelo de recomendação de métricas é desenvolvido, assim como uma interface web para permitir a sua utilização.

1.3. Objetivos Específicos

- Analisar a literatura sobre métricas ágeis e sistemas de recomendação;
- Analisar o estado da arte sobre métricas ágeis e sistemas de recomendação;
- Buscar e estudar métricas de análise de resultado na literatura acadêmica, em artigos especializados, em entrevistas e *surveys* com gerentes de projetos e pesquisadores na área.
- Criar um dataset com as métricas encontradas.
- Desenvolver um modelo de IA para recomendação de métricas.
- Desenvolver um sistema web para o modelo de recomendação ser utilizado.
- Avaliar o resultado do modelo treinado em termos de precisão.

1.4. Método de Pesquisa

O presente projeto tem caráter exploratório e será dividido nas seguintes fases:

Fase 1 – Análise da fundamentação teórica – análise e compreensão dos conceitos básicos para o desenvolvimento do projeto.

Atividade 1.1 – Identificação e documentação dos principais conceitos de gerenciamento de projetos.

Atividade 1.2 – Identificação e documentação dos principais conceitos de medição

Atividade 1.3 – Identificação e documentação dos principais conceitos de machine learning e sistemas de recomendação

Fase 2 – Estado da arte – análise e compreensão dos trabalhos relacionados e artigos, para identificar o estado atual do tema na literatura.

Atividade 2.1 – Definição do protocolo de pesquisa

Atividade 2.2 – Seleção dos estudos

Atividade 2.3 – Extração e análise dos dados

Fase 3 – Realização dos *surveys* – etapa de pesquisa com gerentes de projetos e pesquisadores para obtenção de métricas.

Atividade 3.1 – Definição do questionário

Atividade 3.2 – Realização dos surveys com pesquisadores e gerentes de projeto

Atividade 3.3 – Seleção de medidas e métricas relevantes encontradas

Fase 4 – Preparação do *dataset* – etapa de modelagem do *dataset* com as medidas e métricas encontradas na literatura e nos *surveys* realizados.

Atividade 4.1 – Definição dos critérios de classificação das métricas

Atividade 4.2 – Classificação das métricas encontradas

Atividade 4.3 – Modelagem do *dataset*

Fase 5 – Desenvolvimento do Modelo de IA – etapa de treinamento do modelo de *IA* sobre o *dataset* de métricas criado.

Atividade 5.1 – Definição de critérios de treinamento

Atividade 5.2 – Definição de questionários para o uso da recomendação

Atividade 5.3 – Treinamento do modelo de IA baseado nos critérios definidos nas atividades 5.1 e 5.3

Fase 6 – Implementação da aplicação – etapa de implementação do Serviço de Recomendação (SR) de métricas criado.

Atividade 6.1 – Criação do sistema web para utilizar o SR

Atividade 6.2 – Avaliação do SR

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais conceitos abordados neste trabalho, relacionados ao gerenciamento de projetos e à abordagem de metodologias ágeis e métricas adotadas para a gestão dos mesmos, bem como uma introdução sobre o que é Machine Learning e como essa tecnologia pode ser integrada para otimizar a execução e avaliação de projetos.

2.1. Métodos ágeis

As metodologias ágeis surgiram como uma resposta à crescente taxa de falhas em projetos de desenvolvimento de software, tornando-se particularmente proeminentes após a formação da Aliança Ágil em 2001 e a publicação do Manifesto Ágil (BECK et al., 2001). Estas metodologias são fundamentadas em quatro pilares principais: valorizar indivíduos e interações mais do que processos e ferramentas; priorizar software funcional em detrimento de documentação extensiva; colocar a colaboração com o cliente acima da negociação de contratos; e responder a mudanças em vez de seguir rigidamente um plano.

Estes quatro pilares se dividem em doze princípios que clarificam os elementos essenciais do desenvolvimento ágil. Cada princípio das metodologias ágeis destaca um aspecto crucial do desenvolvimento de software, como publicado no Manifesto Ágil (BECK et al., 2001): garantem a satisfação do cliente através de entregas rápidas e regulares de soluções valiosas; seguem incentivando a adaptabilidade às mudanças, considerando-a essencial no processo de desenvolvimento; a metodologia ressalta a importância de entregas frequentes de software funcional, enquanto promove uma comunicação incessante e uma colaboração robusta entre as equipes de produto e engenharia; enfatiza a criação de um ambiente de trabalho saudável e inspirador, apoia a realização de interações pessoais diretas e incentiva um compromisso constante com a excelência técnica e um design simplificado.

De acordo com Pressman (2010), o processo de desenvolvimento ágil baseia-se em três premissas fundamentais: a dificuldade em prever requisitos e suas alterações, a necessidade de intercalar projeto e construção de software, e a imprevisibilidade nas etapas de análise, projeto, construção e testes.

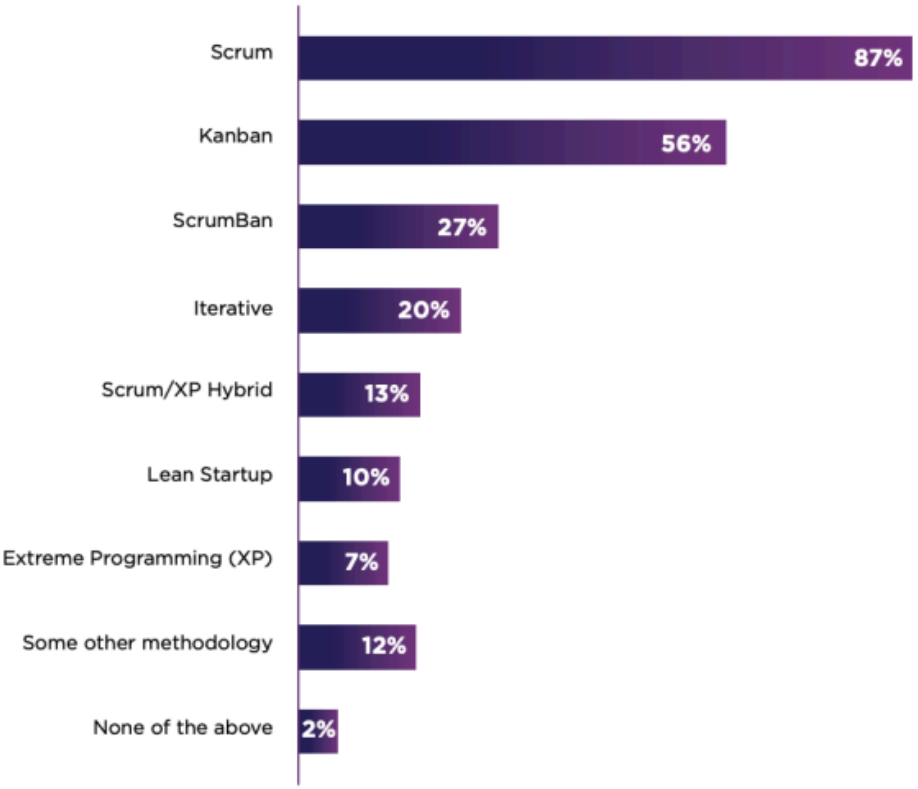


Figura 1: 16 Annual State of Agile Report (VERSIONONE, 2022).

As metodologias ágeis mais utilizadas em 2022, conforme identificado pela pesquisa do grupo QAgile (VERSIONONE, 2022) e mostrado na Figura 1, incluem Scrum, Kanban e ScrumBan, respectivamente, sendo o último a junção das metodologias Scrum e Kanban. A combinação de diferentes metodologias ágeis em um ambiente de desenvolvimento de software pode oferecer vantagens significativas, como maior colaboração e trabalho conjunto entre as equipes, clima organizacional positivo, projetos mais alinhados com os objetivos organizacionais e exigências comerciais e adaptação às mudanças no mercado, encorajamento de inovação entre as equipes e um aumento da vida útil do produto por incorporar feedbacks e adaptações continuamente. É importante ressaltar que para alcançar esses benefícios, é crucial que os participantes tenham um entendimento profundo das diferentes metodologias ágeis e de como elas podem ser eficazmente combinadas e aplicadas, além de conseguir medir estes resultados. Isso requer conhecimento técnico e uma visão estratégica para garantir que as práticas se complementem e conduzam os envolvidos aos resultados desejados.

2.2. Medição de Software

A medição de software é fundamental para compreender, controlar e melhorar os processos de desenvolvimento, especialmente em metodologias ágeis (CHORAŚ et al., 2020). O processo de Medição de Software, conforme delineado por Leal (2023), serve como uma ferramenta objetiva para avaliar o desenvolvimento e o cumprimento de metas e dimensões específicas. A eficácia deste processo depende crucialmente da sua capacidade de fornecer informações valiosas tanto para aqueles que executam a medição quanto para os que se beneficiam dos dados coletados. Segundo Softex (2016), algumas propriedades fundamentais devem orientar estas medições para garantir sua efetividade.

Primeiramente, as medições devem ser objetivas, evitando ambiguidades para permitir uma interpretação clara e precisa por todos os stakeholders. Elas também precisam ser relevantes, atendendo às necessidades específicas e aos objetivos da organização, o que as torna cruciais para o processo decisório. É vital que estejam alinhadas com os objetivos estratégicos da organização, contribuindo diretamente para a realização das metas estabelecidas.

Para serem verdadeiramente úteis, as medições devem manter consistência ao longo do tempo, permitindo análises comparativas de dados históricos. A reprodutibilidade é outro aspecto essencial, pois garante que medições feitas em circunstâncias similares por diferentes indivíduos produzam resultados comparáveis. A simplicidade nas definições das medições também é crucial, facilitando sua compreensão e aplicação. As medições devem ser contextualizadas, levando em consideração as particularidades do ambiente em que são aplicadas, para assegurar sua relevância e adequação às situações específicas.

Baseado no processo detalhado do trabalho e estudo de caso de medição de software documentado no artigo "Measuring and Improving Agile Processes in a Small-Size Software Development Company" (CHORAŚ et al., 2020), é possível caracterizar alguns aspectos cruciais para um processo de medição de software eficaz.

2.2.1. Métricas ágeis

As métricas ágeis de software, conforme descritas por Hartmann e Dymond (2006), são essencialmente projetadas para alinhar-se aos princípios ágeis e devem ser facilmente coletáveis. Estas métricas oferecem uma visão clara sobre o trabalho

que ainda não foi executado, concentrando-se na minimização do esforço e na maximização do valor entregue ao cliente. As métricas devem proporcionar indicadores claros que facilitam o entendimento global do projeto, suportando decisões estratégicas importantes.

Um aspecto crucial dessas métricas é a capacidade de responder de maneira direta às perguntas dos stakeholders que utilizam essas informações, evitando o excesso de dados que pode confundir ou desviar o foco das questões essenciais. Elas são projetadas para serem claras e diretas, minimizando a possibilidade de interpretações variadas e fomentando discussões produtivas. Além de medirem o valor do produto ou processo em avaliação, as métricas ágeis promovem um entendimento aprofundado e direcionado do desenvolvimento dentro do contexto especificado.

Métricas ágeis desempenham um papel crucial no apoio aos processos de monitoramento e controle em projetos que utilizam metodologias ágeis (MAS, A. et al., 2020). A implementação de métricas adequadas é vital, pois elas não apenas refletem o valor entregue ao cliente, mas também ajudam as equipes de projeto a prever conclusões com maior precisão. Estas métricas fornecem informações significativas que registram o histórico do projeto, possibilitando previsões aprimoradas e tomadas de decisão mais informadas.

As métricas ágeis são fundamentadas em medições empíricas e baseadas em valor, em contraste com as medições preditivas tradicionais. Elas são projetadas para adaptar-se às necessidades específicas de cada organização e equipe de trabalho. Para definir métricas eficazes, é essencial identificar os fatores que contribuem para a sua relevância. Segundo Mas, Mesquida e Pacheco (2020), quatro fatores devem ser considerados na definição de métricas:

- Emitir Alertas: As métricas devem fornecer alertas, indicando se algo significativo ocorreu, seja positivo ou negativo, ajudando assim a destacar mudanças importantes.
- Influenciar Comportamentos: As métricas devem impactar o comportamento das pessoas, incentivando mudanças e motivando os membros da equipe.
- Educar: As métricas são ferramentas de aprendizado, auxiliando as equipes a compreenderem o andamento dos projetos e o funcionamento do sistema.
- Expirar ou Reciclar: As métricas devem ser periodicamente revisadas para verificar se ainda são pertinentes, sendo substituídas ou adaptadas conforme necessário.

Estes princípios garantem que as métricas não apenas monitorem o progresso, mas também promovam um ambiente de melhoria contínua e adaptabilidade.

As métricas comuns no desenvolvimento ágil incluem gráficos de "Burndown", taxas de aprovação em testes e ritmo adequado de trabalho, que ajudam no planejamento e rastreamento de sprints ágeis, monitoramento da qualidade do produto e identificação e correção de problemas relacionados ao processo (CHORAŚ et al., 2020).

Segundo Softex (2016), diversas características são associadas ao processo de medição, destacando-se:

1. As métricas podem ser organizadas em escalas, como nominal, ordinal ou de razão (proporcional), e definidas dentro de um intervalo;
2. Uma métricas representa uma variável que necessita ser vinculada a um resultado específico;
3. As métricas podem ser classificadas em básicas ou derivadas.

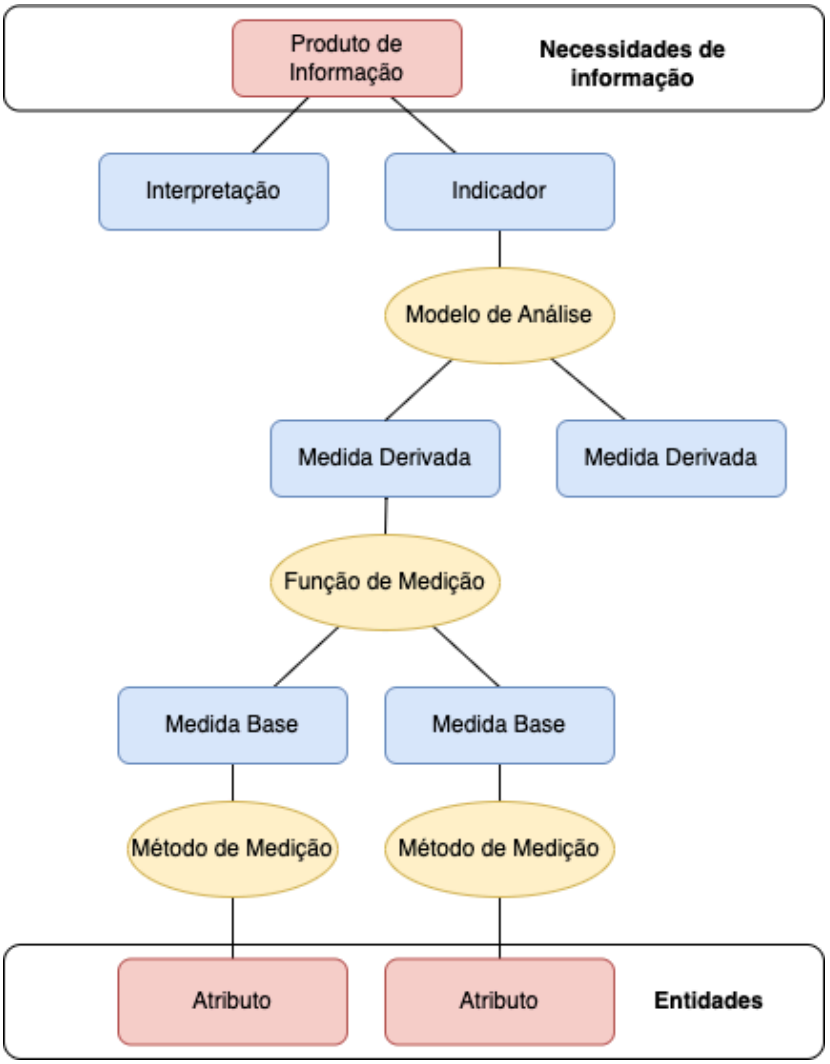


Figura 2: Formação das informações sobre o produto. Fonte: ISO/IEC/IEEE 15939:2017(E) (adaptado pelo Autor)

Este modelo da Figura 2 é descrito em Softex 2016, adaptado de ISO/IEC/IEEE 15939:2017(E)(2017) e fornece uma definição de um modelo de medição de informação, caracterizando três tipos de mensuração: básicas, derivadas e indicadores. Os componentes da Figura 2 são descritos a seguir:

1. Entidade: descreve um objeto, como um processo ou projeto, definido por seus atributos de medição. Uma entidade pode apresentar múltiplas propriedades de interesse para o modelo de informações.
2. Atributo: propriedade relevante para as necessidades de informação. Uma propriedade ou característica de uma entidade, a qual pode possuir diversos atributos. Podem ser identificados tanto manualmente quanto por processos automatizados.

3. Medida Base: uma medida de um único atributo por um método específico. Segundo Softex (2013), uma medida base é independente de outras medidas e tem sua essência capturada em um único atributo. Exemplo: linhas de código.
4. Medida Derivada: quantidade definida como uma função de duas ou mais medidas. Estas medidas são definidas pela junção de dois ou mais valores básicos ou derivados. Exemplo: linhas de código/horas trabalhadas (Softex (2013) e ISO/IEC/IEEE 15939:2017(E) (2017)).
5. Indicador: estimativa ou avaliação que serve de base para a tomada de decisões. Segundo ISO/IEC/IEEE 15939:2017(E)(2017), permite quantificar incerteza ou acurácia e Teixeira (2018) explica que indicadores são criados ao combinar dados com contextos e regras específicas, o que resulta em números com significados. Os resultados obtidos produzem informações valiosas para a tomada de decisão.
6. Modelo de Análise: algoritmo que combina medidas e critérios de decisão.
7. Função de Medição: algoritmo que combina duas ou mais medidas base.
8. Método de Medição: operação de um único atributo por um método específico.

2.2.2. Seleção de métricas

As métricas devem estar alinhadas com os objetivos estratégicos da organização. A definição deve ser clara do que se espera alcançar com a medição, seja para melhorar a qualidade do produto, otimizar os processos, aumentar a satisfação do cliente, ou reduzir custos.

Dentre as diversas medidas que podem ser utilizadas para o gerenciamento dos processos de desenvolvimento de software, pode ser difícil para uma organização ou até para uma equipe, selecionar aquelas medidas que melhor atendem às suas necessidades de informação. Assim, para realizar a seleção das métricas mais apropriadas podem ser utilizados os seguintes critérios (CHORAŚ et al., 2020):

- Relevância: as métricas escolhidas devem ser relevantes para as necessidades do negócio e devem contribuir para o alcance dos objetivos definidos.
- Simplicidade e Compreensibilidade: as métricas devem ser fácil de entender e de comunicar, evitando complexidades desnecessárias que podem levar a interpretações erradas

- Capacidade de Ação: métricas que possam influenciar na tomada de decisão e em ações concretas devem ser prioridades, em detrimento de métricas que apenas informam sem levar a ações corretivas.

2.2.2.1. Coleta de Dados

Nesta seção serão exploradas duas metodologias de extração de métricas relevantes:

- Automação: utilizar ferramentas que automatizam a coleta de dados sempre que possível, para que o trabalho manual seja evitado.
- Fontes de Dados: as fontes dos dados coletados devem ser confiáveis e consistentes. Os dados podem vir de ferramentas de controle de versão, rastreamento de problemas e tarefas e ferramentas de desenvolvimento, por exemplo.

2.2.3. Análise dos Dados e Iteração

Os dados devem ser processados e agregados após sua coleta, para fornecerem informações úteis. O uso de ferramentas adequadas para a visualização dos dados podem incluir *dashboards* que permitam que os *stakeholders* visualizem métricas importantes de maneira fácil e intuitiva. A alimentação e atualização dos dados deve ser frequente e em tempo real sempre que possível. Isso fornece feedback e interação contínua sobre sobre o processo de medição e desenvolvimento do produto, permitindo ajustes e melhorias de forma ágil. As métricas devem ser revisitadas e ajustadas periodicamente, para garantir que continuem relevantes e úteis à medida que o projeto e o ambiente de negócios evoluem.

Estes aspectos garantem que as métricas escolhidas sejam relevantes, aplicáveis e úteis para melhorar tanto a qualidade do produto quanto a eficiência do processo. Quando aplicadas ao contexto de desenvolvimento ágil de software, a eficácia da medição é vital para entender os custos e a qualidade do desenvolvimento, apoiando as atividades de planejamento, monitoramento, controle e avaliação dos processos de software (P. RAM et. al., 2018).

2.3. Inteligência Artificial

O termo Inteligência Artificial (IA) foi primeiramente introduzido pelo cientista da computação John McCarthy em 1955. Hoje, este é um campo abrangente e multifacetado que engloba uma variedade de disciplinas e tecnologias, estendendo-se

desde a lógica e a matemática até a eletrônica e a robótica (RUSSEL & NORVIG, 2013). Atualmente, a IA encapsula o desenvolvimento de agentes inteligentes capazes de receber e processar percepções e dados do ambiente para executar ações eficazes, refletindo a complexidade e a amplitude da aplicabilidade da IA no mundo moderno. O presente tópico irá tratar sobre Inteligência Artificial e Machine Learning e suas derivações, seguindo uma ordem lógica apresentada na Figura 3.

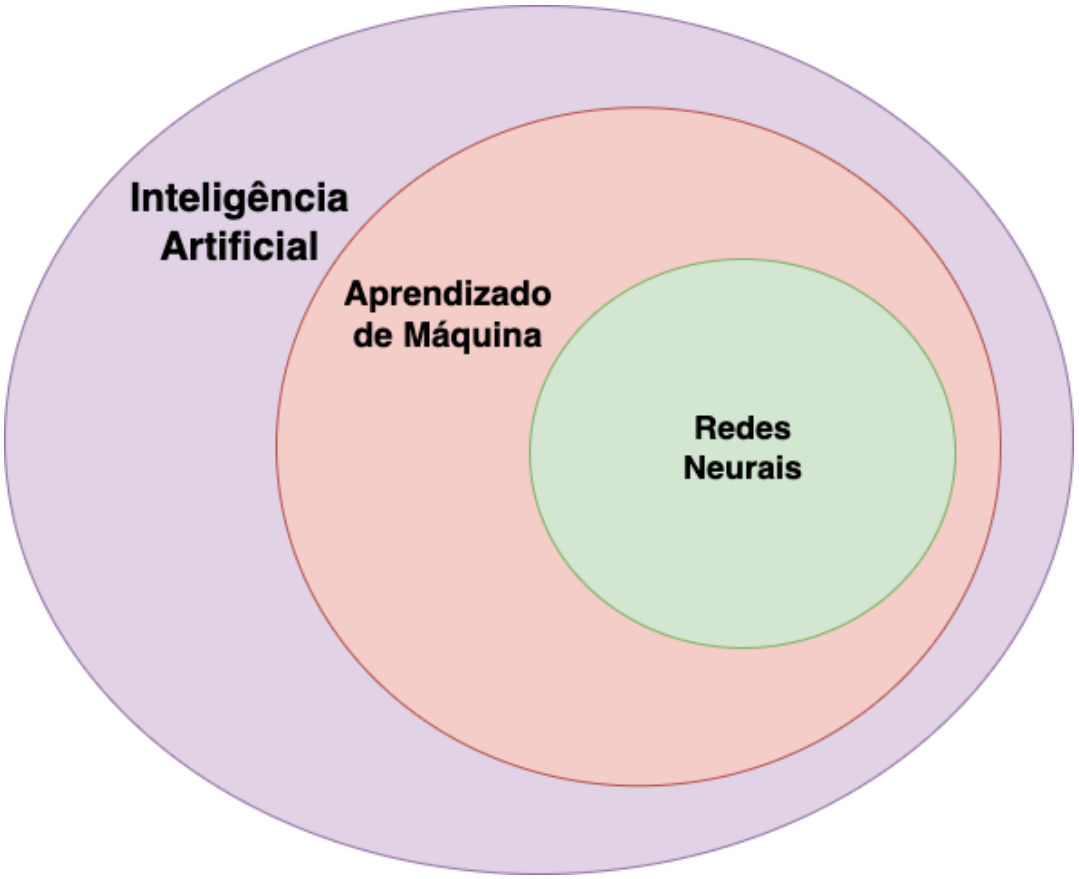


Figura 3: Diagrama de Euler para Aprendizado de Máquina (GAVRILOVA, 2020). Adaptado pelo autor.

Conforme definido por Arthur Samuel em 1959, o *Machine Learning* (ML), também conhecido como Aprendizado de Máquina, é uma disciplina dentro do campo maior da Inteligência Artificial que se dedica ao desenvolvimento de algoritmos capazes de aprender a partir de dados. É o campo de estudo que confere aos computadores a capacidade de aprender sem que para isso precisem ser explicitamente programados. Tom Mitchell, em 1997, oferece uma perspectiva complementar, definindo Machine Learning como um ramo da IA focado em

programas que aprendem a partir de experiências e aprimoram seu desempenho em tarefas específicas, com base em uma medida de desempenho. A capacidade de adaptação e aprimoramento contínuo das respostas e comportamentos são os objetivos centrais do ML. De acordo com Escovedo e Koshiyama (2020), o processo de aprendizagem em ML ocorre quando o algoritmo busca identificar padrões em um conjunto de dados fornecido, processo frequentemente referido como treinamento, construção ou formulação. Este esforço para discernir estruturas e regularidades nos dados é fundamental para que o algoritmo possa posteriormente aplicar esses aprendizados de maneira eficaz em situações novas ou desconhecidas. O ML é distintivamente iterativo e progressivo, caracterizando-se pelo modo como os algoritmos continuamente ajustam e aprimoram seu comportamento à medida que recebem novas informações e resultados ao longo do processo de treinamento

O campo do Machine Learning é comumente dividido em três grandes grupos de aprendizado: supervisionado, não supervisionado e por reforço. O aprendizado supervisionado e o não supervisionado são também conhecidos como modelos preditivos e modelos descritivos, respectivamente (FACELI et al., 2015).

2.3.1. Aprendizado supervisionado

No aprendizado supervisionado, os algoritmos são treinados usando um conjunto de dados que já inclui as respostas desejadas, conhecidas como rótulos. O objetivo é que o modelo aprenda a mapear as entradas para as saídas corretas, facilitando a realização de previsões ou classificações precisas em novos dados com base no que foi aprendido durante o treinamento. Este método é amplamente utilizado para tarefas como reconhecimento de fala, diagnóstico médico e classificação de e-mails.

2.3.1.1. Features e Seleção de Features

Features são atributos ou variáveis que representam as características de um conjunto de dados, servindo como entrada para algoritmos de aprendizado de máquina ou outras análises. Elas podem ser categóricas, contínuas ou derivadas de transformações, e seu papel é descrever os padrões relevantes nos dados para facilitar a modelagem e a extração de conhecimento. A seleção de features é o processo de identificar os atributos mais relevantes e excluir aqueles que são redundantes ou irrelevantes (LANGLEY, 1994).

2.3.2. Aprendizado não supervisionado

O aprendizado não supervisionado envolve trabalhar com dados que não possuem rótulos pré-definidos. O algoritmo tenta organizar os dados ou descrever sua estrutura de maneira que faça sentido inferir propriedades ou segmentações úteis, como agrupar consumidores com preferências semelhantes sem conhecer essas categorias antecipadamente. Este tipo de aprendizado é ideal para explorar padrões ocultos ou intrínsecos nos dados, como na segmentação de mercado ou na detecção de atividades anômalas.

2.3.3. Aprendizado por reforço

O terceiro grande grupo, o aprendizado por reforço, descrito por Sutton e Barto (1998), foca na tomada de decisões sequenciais. O algoritmo aprende a alcançar um objetivo através de tentativa e erro, recebendo recompensas ou punições com base na eficácia de suas ações (WINDER, 2020). Esta forma de aprendizado é particularmente útil em cenários que requerem uma série de decisões interconectadas, como nos jogos de estratégia, na navegação de robôs ou na gestão de investimentos.

2.3.4. Redes Neurais

Segundo Chevitarese (2010), uma rede neural consiste em um sistema computacional formado por elementos processadores interligados, chamados neurônios, que trabalham simultaneamente para realizar tarefas específicas (SANTOS, et al., 2019). As redes neurais foram desenvolvidas com o intuito de emular o funcionamento dos neurônios humanos de maneira artificial. Essa inspiração catalisou a criação das redes neurais artificiais, sistemas altamente robustos capazes de aproximar funções que assumem valores reais, discretos ou vetoriais. Como resultado, elas fornecem métodos eficazes para a interpretação de dados complexos provenientes de sensores no mundo real, conforme destacado por Russell e Norvig (2013).

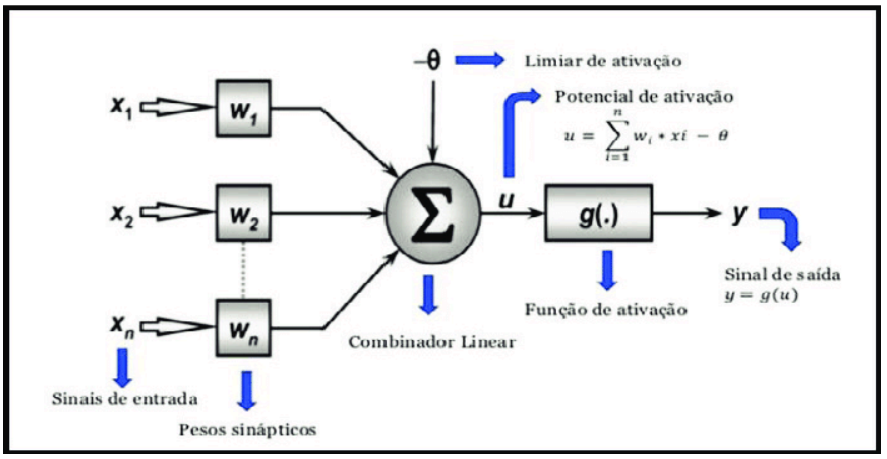


Figura 4: Esquema de um neurônio artificial de única camada (HAYKIN, 2001).

Em uma rede neural, a conexão entre neurônios é estruturada de forma intrínseca, onde cada neurônio é interligado aos outros por meio de ligações direcionadas. Essas ligações possuem um peso numérico associado, conhecido como "peso sináptico" conforme explicado por Russell e Norvig (2013) e ilustrado na Figura 4. O peso de cada ligação determina tanto a força quanto o sinal da conexão, podendo amplificar ou atenuar o sinal transmitido entre as unidades. Cada neurônio na rede recebe um ou mais sinais de entrada, seja de outros neurônios ou de fontes externas, processa esses sinais e, em seguida, emite um único sinal de saída. Este sinal de saída é então transmitido adiante na rede, servindo como entrada para um ou mais neurônios nas camadas subsequentes (PIO, 2009).

A Figura 5 apresenta tipos diferentes de arquiteturas de redes neurais.

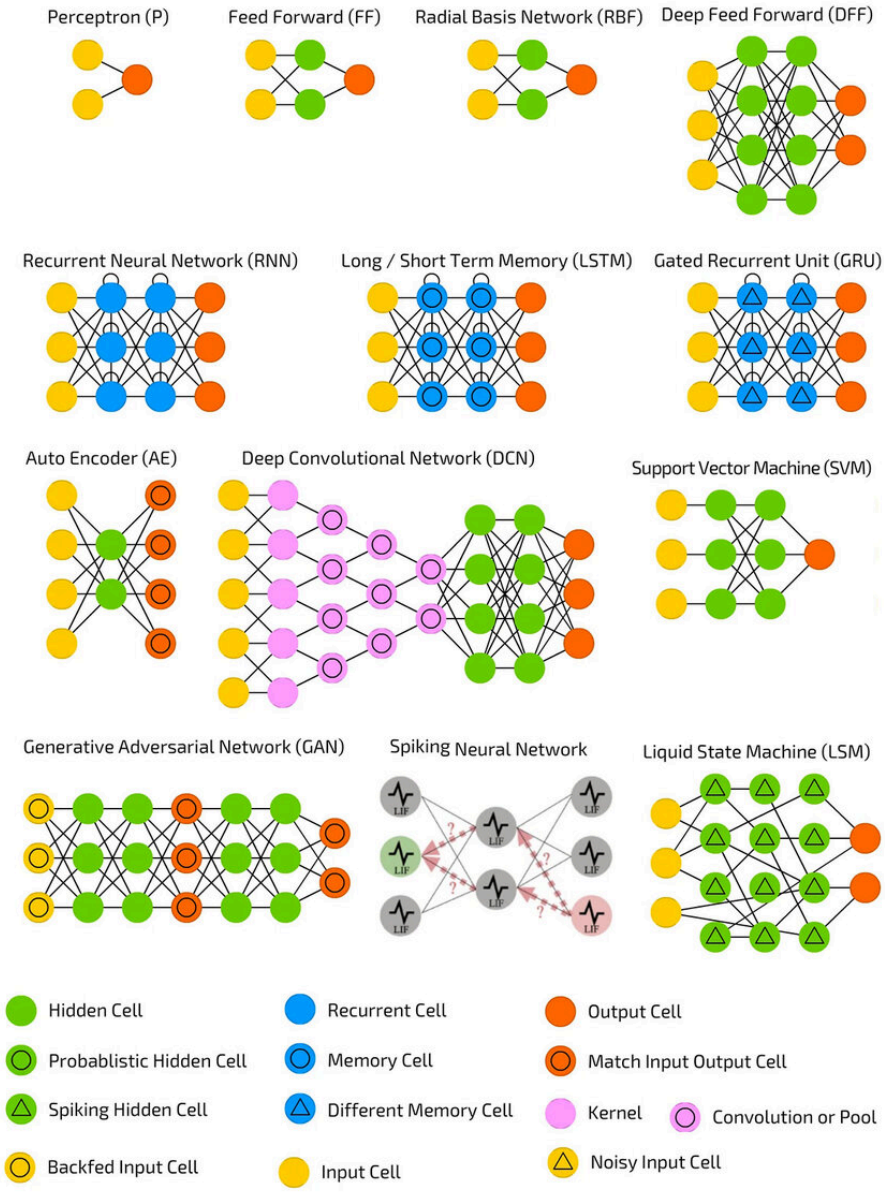


Figura 5: Diferentes arquiteturas de redes neurais (VAN VEEN, 2019).

Algumas das arquiteturas apresentadas na Figura 5 podem ser descritas da seguinte forma (VAN VEEN, 2019):

- **Feed Forward (FF ou FFNN):** Estas são as formas mais básicas de redes neurais, onde a informação flui em apenas uma direção — da entrada, passando pelas camadas ocultas, até a saída. Não há ciclos ou loops na rede, o que simplifica a arquitetura e a implementação.
- **Radial Basis Function Networks (RBF):** Uma variação das redes FFNN, estas utilizam funções de base radial como funções de ativação. São

especialmente úteis para problemas que requerem a captura de relações não lineares fortes entre os atributos de entrada e a saída.

- **Recorrente Neural Networks (RNN):** Diferentemente das redes FFNN, as RNNs têm conexões que formam ciclos, permitindo que informações de saídas anteriores sejam usadas como entradas para a rede, criando um sistema dinâmico e "memorizador". Isso as torna ideais para tarefas que envolvem sequências de dados, como linguagem natural e séries temporais.
- **Convolutional Neural Networks (CNN):** Estas redes são estruturalmente diferentes das FFNNs, sendo otimizadas para capturar padrões espaciais e temporais em dados dimensionais, como imagens e áudio. Utilizam um processo matemático chamado convolução que efetivamente identifica e isola características significativas dos dados para classificação ou análise mais aprofundada.

2.4. Sistemas de Recomendação

Sistemas de recomendação são ferramentas de inteligência artificial que ajudam a filtrar e apresentar informações ou itens que possam interessar aos usuários. Eles são usados em uma variedade de domínios, incluindo filmes, música, livros, produtos de varejo e mais (SHAFFER, 2001). Prevalentemente, os Sistemas de Recomendação utilizam majoritariamente diferentes técnicas de machine learning para prever as preferências dos usuários, ajudando-os a descobrir conteúdos ou produtos que provavelmente irão apreciar. Estes sistemas se utilizam de dados históricos de um usuário para filtrar conteúdos, apresentando apenas os itens mais relevantes e interessantes para esse usuário (ABBAR et al., 2009). Ainda, segundo Primo (2013) sistemas de recomendação atuam de forma pró-ativa sugerindo novos itens ao usuário, com o objetivo de facilitar a seleção de itens em sistemas que possuem uma grande quantidade de informações.

2.4.1. Sistemas de recomendação de abordagem clássica

Na abordagem clássica de sistemas de recomendação, esses sistemas utilizam técnicas de aprendizado de máquina para analisar dados e aplicar algoritmos que preveem a avaliação ou a preferência de um usuário por um item específico. Esses sistemas são categorizados com base no tipo de algoritmo empregado para fazer as recomendações. Segundo Morandino e Rodrigues (2020), as principais técnicas clássicas incluem:

- Filtragem Colaborativa (*Collaborative Filtering*): Esta abordagem foca em recomendar itens com base na similaridade entre usuários, independentemente das características específicas dos itens. Ela identifica padrões de comportamento compartilhados entre os usuários para fazer suas recomendações.
- Filtragem Baseada em Conteúdo (*Content-Based Filtering*): Essa técnica tem a vantagem de não requerer uma grande quantidade de *feedback* dos usuários para fazer recomendações úteis. Ela se baseia nas características dos itens para recomendar produtos semelhantes aos que um usuário já demonstrou interesse.
- Sistemas Híbridos (*Hybrid Systems*): Combinam elementos das abordagens de filtragem baseada em conteúdo e colaborativa. Esses sistemas híbridos buscam aproveitar as vantagens de ambas as técnicas para proporcionar recomendações mais precisas e abrangentes.

A seguir, as próximas seções serão direcionadas para explicar melhor o uso de metodologias para o desenvolvimento de um Sistema de Recomendação, se aprofundando melhor em filtragem colaborativa, filtragem baseada em conteúdo e sistemas híbridos.

2.4.2. Filtragem Colaborativa

A filtragem colaborativa é uma abordagem central nos sistemas de recomendação modernos, baseando-se na premissa de que usuários que concordaram no passado tendem a ter opiniões semelhantes no futuro. Este método utiliza o histórico de interações dos usuários para prever itens que possam ser de interesse, sem a necessidade de entender o conteúdo dos itens em si. A abordagem é predominantemente utilizada em plataformas como a Netflix e a Amazon, onde o comportamento do usuário em relação a avaliações de filmes ou produtos é analisado para recomendar novos itens para outros usuários semelhantes (SANTANA, 2018). Esta abordagem não considera diretamente o conteúdo dos itens recomendados, mas sim padrões de comportamento e preferências dos usuários. Eles são amplamente utilizados em datasets onde os dados podem não estar perfeitamente estruturados, mas ainda assim requerem algum nível de processamento para gerar recomendações eficazes.

Segundo Adomavicius e Tuzhilin, 2005, a filtragem colaborativa pode ser explicada desta forma: a função de utilidade *u(a,b)* de um item *b* para um usuário *a* é

determinada com base nas utilidades previamente calculadas desse mesmo item *b* para um subconjunto *C'* do conjunto total de usuários *C*. Esse subconjunto *C'* é composto por usuários que são de alguma forma semelhantes ao usuário *a*, para quem se deseja fazer a recomendação. Ou seja, para recomendar um item a um usuário *a,* primeiro o Sistema de Recomendação identifica usuários semelhantes ao usuário *a.* A medida de similaridade pode ser obtida de diversas formas, como por exemplo, se estivermos tratando de um filme ou de uma compra, comparando as avaliações dos itens já avaliados por *a* com as avaliações dos demais usuários. Em seguida, o sistema de recomendação sugere ao usuário *a* os itens mais bem avaliados por esse grupo de usuários semelhantes a ele. O grau de similaridade entre os usuários é definido, dentro deste exemplo, do quanto mais estas avaliações forem semelhantes.

Existem muitas formas e abordagens para a construção de algoritmos de filtragem colaborativa. De acordo com Primo, 2013, grande parte destes algoritmos são construídos com a técnica dos *K* vizinhos mais próximos (*K-nearest-neighbours*), sendo o número de *k* vizinhos determinado no momento do desenvolvimento do algoritmo pelo próprio desenvolvedor. Os passos para a execução do algoritmo são:

1. determinar os *k* vizinhos para o usuário *x*;
2. implementar uma abordagem para agregar as avaliações desses *k* vizinhos para um item não avaliado por *x*;
3. escolher as *n* recomendações melhores avaliadas baseadas no passo 2. (Bobadilla, 2013).

Outra abordagem bastante utilizada para este tipo de filtragem é a utilização da matriz de interação usuário - item, onde as interações feitas por cada usuário são mapeadas de forma que cada coluna representa uma interação com um item específico. Segundo Das et al., 2017, é possível prever a avaliação de um determinado item com base nas avaliações de outros usuários considerados similares.

2.4.3. Baseada em Conteúdo

Estratégias de recomendação baseadas em filtragem por conteúdo recomendam itens ao usuário levando em consideração o seu perfil (PRIMO, 2013). Para realizar a recomendação de um item *a* para um usuário *b,* o sistema compara as características do item *a* com outros itens que o usuário interagiu no passado (SHU et al., 2017). Por exemplo, se o usuário consome filmes do gênero ação em uma plataforma de reprodução de filmes, o sistema pode continuar recomendando filmes

deste mesmo gênero para ele. Muitos sistemas requerem que o usuário continue dando *feedback* constante para melhorar a acurácia da recomendação dos itens, como marcar um item como interessante ou não, uma vez que os sistemas construídos na abordagem de filtragem por conteúdo recomendam itens que são relacionados a conteúdos conhecidos do usuário (CARRER-NETO et al. 2012). De acordo com Adomavicius e Tuzhilin, 2005 o grau de similaridade entre entre um item *i* e um usuário *a* pode se dar através da função utilidade *u(a, i).* O resultado é estimado com base em *u(a, I')*, onde *I'* é subconjunto de todos os itens *I* que o usuário já interagiu e avaliou e que são similares a *i.*

Os sistemas construídos nesta abordagem geralmente calculam a similaridade entre itens utilizando a medida TF-IDF (*Text frequency, Inverse document frequency*), por serem focados na recomendação de elementos textuais, como documentos (ADOMAVICIUS, TUZHILIN, 2005). TF-IDF é uma técnica comum em mineração de texto para representar documentos como vetores numéricos com base na importância de palavras ou termos.

Sistemas de Recomendação com abordagem de filtro baseado em conteúdo não dependem de outros usuários para gerar recomendações, o que os torna capazes de recomendar itens que podem não ser tão populares Entretanto, estes sistema podem sofrer com problemas de *cold start,* onde usuários novos que ainda não interagiram com o sistema não teriam informações suficientes para gerar recomendações (DAS et al., 2017) e ainda problemas de superespecialização, uma vez que dependendo fortemente de dados históricos do usuário (CARRER-NETO et al., 2012).

2.4.4. Sistemas Híbridos

Sistemas híbridos surgem para resolver os problemas que não podem ser resolvidos tão somente por um tipo de abordagem. Estes sistemas combinam uma ou mais técnicas para obter um melhor resultado, bem como reduzir efeitos negativos de uma abordagem específica (BURKE, 2002). Sistemas híbridos podem ser compostos por técnicas de filtragem colaborativa e filtragem baseada em conteúdo, por exemplo, para recomendar conteúdo baseado em similaridade entre os usuários combinando com o feedback do usuário sobre determinados itens. Este modelo é utilizado em plataformas como a Netflix, para a recomendação de filmes e séries em seu catálogo.

Existem diversos métodos de desenvolver um sistema híbrido, de acordo com Primo (2013), eles são respectivamente: Método Balanceado, Método de Permuta, Método Mesclado, Método de Combinação de Características, Método em Cascata, Método de Acréscimo de Características e Método em níveis.

1. Balanceado: as avaliações das diferentes técnicas são combinadas para produzir uma única avaliação (BURKE, 2002). Cada componente do sistema avalia itens de forma independente e os resultados são combinados ao final para serem apresentados de forma única (PRIMO, 2013).
2. Permuta: este método gera recomendações a nível de item, onde as técnicas são permutadas entre si de acordo com um critério definido. Por exemplo, um sistema que se utiliza de filtragem colaborativa e por conteúdo pode avaliar um item utilizando filtragem colaborativa, e caso o resultado fique abaixo de um limiar, o sistema então avalia o item com a técnica baseada em conteúdo. (BURKE, 2002)
3. Mesclado: o método mesclado é parecido com o método de permuta, porém não se baseia em nenhum critério para alternar entre as técnicas de recomendação (PRIMO, 2013).
4. Combinação de Características: neste método, o algoritmo de recomendação é construído com características de diferentes técnicas visando uma recomendação única com propriedades de diferentes (PRIMO, 2013).
5. Cascata: as técnicas de recomendação são aplicadas em estágios, formando grupos preliminares, que então tem outras técnicas de recomendação aplicadas sobre eles. (BURKE, 2002).
6. Acréscimo de Características: são incorporadas técnicas de recomendação diferentes em cada etapa do processo de recomendação (BURKE, 2002).
7. Em níveis: junta diferentes técnicas de recomendação utilizando o modelo completo gerado por uma técnica como um parâmetro de entrada para outro modelo com uma técnica diferente (BURKE, 2002)

3. PROPOSTA

Nesta seção será abordada a proposta de desenvolvimento do Sistema de Recomendação de Métricas Ágeis, bem como os primeiros passos dados em direção a sua elaboração, que ajudaram a restringir o escopo, os métodos e as ferramentas que serão utilizadas na construção no projeto final.

3.1. Preparação do Dataset

Os dados que formam o dataset utilizado neste projeto foram colhidos em conjunto com a mestranda Stéphanie da Silva Leal por meio de um *survey,* tendo como objetivo investigar os processos de seleção de métricas e as métricas utilizadas nas empresas desenvolvedoras de software no Brasil (LEAL, 2023).

O autor deste trabalho de conclusão de curso participou ativamente de todas as etapas de planejamento, execução e análise de dados do *survey* sob coordenação da mestranda.

A execução do *survey* foi autorizada pelo Comitê de Ética em Pesquisa com Seres Humanos da UFSC (CEPESH-UFSC), sob número 55851622.8.0000.0121.

3.1.1. Planejamento do survey

O desenvolvimento do *survey* foi fundamentado na abordagem proposta por Kasunic (2005), a qual define os seguintes passos:

- **Objetivo da pesquisa**: Nesta fase, estabelece-se o objetivo principal da pesquisa, que é posteriormente subdividido em perguntas específicas que, em conjunto, respondem à questão central do estudo.
- **Público-alvo e amostra**: Esta etapa envolve a definição do público-alvo e da amostra necessária para representar adequadamente a população.
- **Design do instrumento de coleta de dados**: O objetivo desta fase é determinar o instrumento que será utilizado para a coleta das respostas, bem como a estruturação das perguntas.
- **Distribuição do questionário**: Nesta etapa, define-se a forma de distribuição do questionário, seja por meios físicos ou virtuais, e o método de disseminação, como o uso de ferramentas específicas ou redes de contatos.
- **Análise dos resultados e desenvolvimento de relatórios**: Esta fase consiste na análise das respostas obtidas, visando verificar se estas atendem à questão

de pesquisa, além do desenvolvimento de gráficos e relatórios sobre os resultados encontrados.

3.1.2. Desenvolvimento do Instrumento de Coleta de Dados

As perguntas foram elaboradas com o objetivo de responder às necessidades levantadas pelo *survey*, de analisar os diferentes perfis de empresas que fazem uso de métricas, identificando quais métricas utilizam e como ocorre o processo de seleção destas. Com isto, os seguintes questionamentos deveriam ser respondidos com base nos resultados do survey:

- P1: Qual o perfil dos respondentes que fazem uso de métricas ágeis?
- P2: Qual o perfil das organizações que fazem uso de métricas ágeis?
- P3: Como as métricas são selecionadas?
- P4: Como as métricas são utilizadas?
- P5: Quais são as métricas utilizadas?
- P6: Qual o impacto do perfil organizacional no método ágil escolhido?
- P7: Qual o impacto do perfil organizacional em como as métricas são utilizadas?
- P8: Qual o impacto do perfil organizacional entre os grupos de métricas mais utilizados?

Com base nestes questionamentos, um questionário foi preparado para o *survey* e o Quadro 1 a seguir demonstra seu resultado:

Pergunta 1	Confirmo meu interesse em participar desta pesquisa tendo lido e aprovado o Termo de Consentimento disponível em https://bit.ly/3uE0i4O
Pergunta 2	Em qual região fica a sede da empresa que trabalha?
Pergunta 3	Qual seu papel na equipe que atua?
Pergunta 4	Há quanto tempo (em anos) você trabalha com desenvolvimento de software?
Pergunta 5	Aproximadamente quantos colaboradores tem sua organização?
Pergunta 6	Para quais áreas de domínio a sua organização desenvolve software?
Pergunta 7	Qual(is) métodos ágeis seu time utiliza?
Pergunta 8	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de planejamento da sprint]
Pergunta 9	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de revisão da sprint]

Pergunta 10	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião semanal]
Pergunta 11	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião diária]
Pergunta 12	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de retrospectiva da sprint]
Pergunta 13	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Nenhuma das respostas acima]
Pergunta 14	Você utiliza métricas para acompanhamento das metas e objetivos do(s) time(s) ou da organização?
Pergunta 15	Em quais dos itens abaixo você aplica métricas para o gerenciamento?
Pergunta 16	Quais as 5 principais métricas que você utiliza em seu(s) time(s) e/ou organização?
Pergunta 17	Existe algum método ou processo estabelecido para definição ou seleção das métricas a serem utilizadas?
Pergunta 18	Se sim, como é esse processo?

Quadro 1: Perguntas formuladas para o survey. Fonte: Leal (2023), adaptado pelo autor.

3.1.3. Coleta de dados

O questionário foi desenvolvido utilizando o aplicativo *Google Forms*, que fornece as ferramentas necessárias para a criação de questionários e está atrelado ao *Google Drive*, facilitando o envio para os participantes através de um link gerado. Os participantes do survey foram encontrados através da rede social profissional *LinkedIn,* que permite com que profissionais de diversas áreas façam conexões entre si. Esta plataforma permitiu com que profissionais que trabalham em gestão e utilizam de alguma forma métricas ágeis nos seus processos fossem contactados.

3.1.4. Amostragem

O público alvo da pesquisa foram colaboradores de empresas de software brasileiras que utilizam métricas ágeis e possuem conhecimento sobre as métricas ágeis empregadas na empresa (LEAL, 2023). A pesquisa teve amostragem probabilística, com os participantes sendo selecionados aleatoriamente e balanceados por estados brasileiros. A distribuição seguiu a pesquisa da ABES (2022), que detalha a quantidade de empresas por região no Brasil. Foi definido que para a pesquisa ter um resultado de amostragem significativo, seriam necessárias 273 respostas (BERNDT, 2020), com profissionais distribuídos proporcionalmente entre as regiões

brasileiras da seguinte forma: 2,5% na região Norte, 7% no Nordeste, 12% no Centro-Oeste, 13,5% no Sul e 65% no Sudeste.

3.1.5. Resultados e Dados Brutos

Ao todo, foram convidados 1043 profissionais para participarem da pesquisa, ao longo do período de um ano. Destes, 309 responderam, resultando em uma taxa de adesão à participação da pesquisa de 29,6%. Foi então realizada uma sanitização dos dados brutos, removendo participantes que não aceitaram o termo de consentimento, formando agrupamentos para facilitar a análise posterior que utilizou a técnica Open Coding para classificar as respostas de perguntas abertas (LEAL, 2023). Esta técnica consiste na análise aberta e sistemática de dados brutos, codificando-os em categorias, identificando padrões e conceitos emergentes para obter uma compreensão aprofundada dos dados (CHARMAZ, 2014).

As figuras a seguir foram extraídas do trabalho de Leal, que ilustram alguns demográficos dos respondentes da pesquisa e que são encontrados nos dados brutos que foram gerados através deste survey.

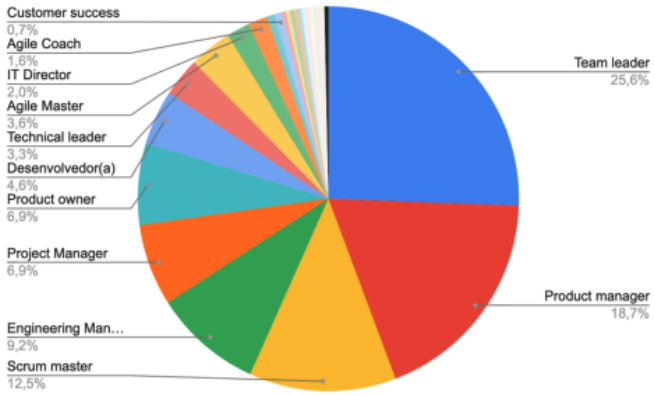


Figura 6: Perfil atuante do respondente em sua organização. Fonte: Leal, 2023.

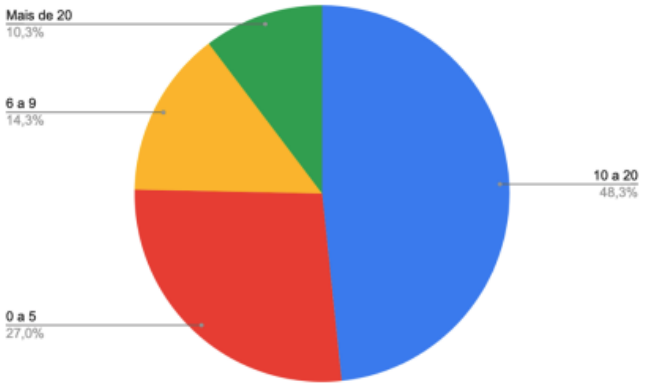


Figura 7: Tempo que atua com desenvolvimento de software. Fonte: Leal, 2023.

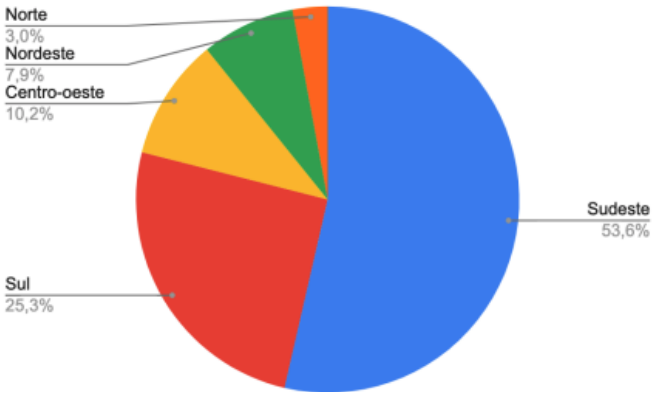


Figura 8: Região em que a empresa que trabalha é sediada. Fonte: Leal, 2023.

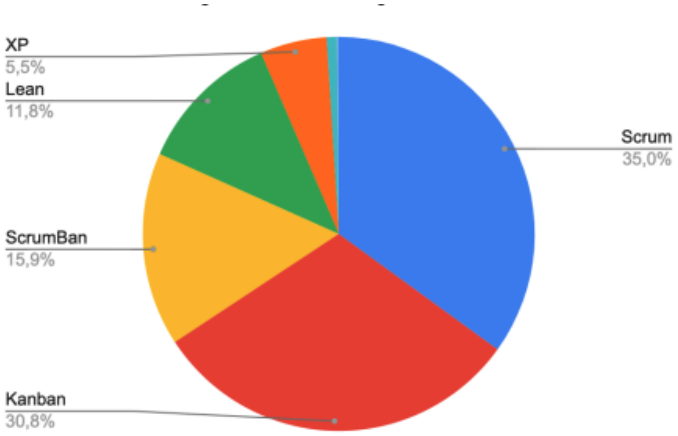


Figura 9: Métodos ágeis utilizados. Fonte: Leal, 2023.

As métricas ágeis que foram utilizadas como base foram extraídas de um mapeamento sistemático da literatura realizado pela mestranda Stéphanie da Silva Leal (LEAL, 2023) e disponibilizado ao autor para a elaboração deste trabalho. O autor não participou deste mapeamento sistemático. Ao todo, a autora recolheu 319 métricas ágeis em diferentes estudos, resultando 132 métricas únicas (LEAL, 2023). A autora então mapeou todas as respostas das métricas obtidas com os resultados do *survey* com as métricas extraídas de literatura equivalentes.

Metric #	Metric Name	Source (primary study #)	Name of metric by primary study	Metric Description	Common Issue Area (PSI Classification)
M01	Individual effectiveness	S12	Individual effectiveness	n/i	Resources and Cost
M01		S13	Individual performance	Individual performance: con	Resources and Cost
M02	Weekly working hours of ind	S12	Weekly working hours of ind	n/i	Resources and Cost
M03	Individual effective available	S12	Individual effective available	n/i	Resources and Cost
M04	Individual Contribution	S11	Contribution	Measures the direct particip	Resources and Cost
M05	Individual Influence	S11	Influence	Measures if and how an indi	Resources and Cost
M06	Individual Impact	S11	Impact	Documents the causal relati	Resources and Cost
M07	Individual Impression	S11	Impression	Measures how well team m	Resources and Cost

Figura 10: Ilustração da planilha resultante do mapeamento sistemático. Fonte: Leal, 2023.

Os dados brutos e o survey completo, além de outros recursos disponibilizados pela autora, como o mapeamento sistemático da literatura relacionado a métricas podem ser acessados no link https://zenodo.org/records/10578050 disponibilizado pela autora (LEAL, 2023).

3.2. Preparação dos dados

Nesta seção serão detalhados os passos realizados que foram necessários para determinar a abordagem final e a abordagem utilizadas para a criação do Sistema de Recomendação proposto. Será dividida da seguinte forma: análise exploratória dos dados brutos, normalização dos dados, transformação dos dados, tratamento de colunas e tradução de valores utilizando *one hot encoding*.

3.2.1. Análise Exploratória dos Dados Brutos

A seguir são apresentados os passos que compõem a análise exploratória efetuada sobre os dados brutos coletados por meio do *survey* conduzido, fornecendo uma visão geral do processo seguido para examinar e interpretar os dados obtidos.

A análise exploratória foi realizada utilizando a biblioteca *pandas*[1], uma biblioteca de código aberto desenvolvida para manipulação e análise de dados em linguagem de programação *Python*. *Pandas* proporciona estruturas de dados e ferramentas de análise de alto desempenho, sendo amplamente utilizada em áreas como ciência de dados, estatística, economia e aprendizado de máquina. Com esta biblioteca é possível realizar operações como limpeza e transformação de dados, e extrair informações significativas a partir dos dados brutos. O código foi escrito em *Jupyter Notebooks*[2], que é uma aplicação web interativa que permite a criação e compartilhamento de documentos com código executável. Essa escolha foi feita pela facilidade em visualizar os resultados, documentar as etapas e interagir com os resultados obtidos através dos comandos executados em *pandas*.

Reading the dataset with raw data

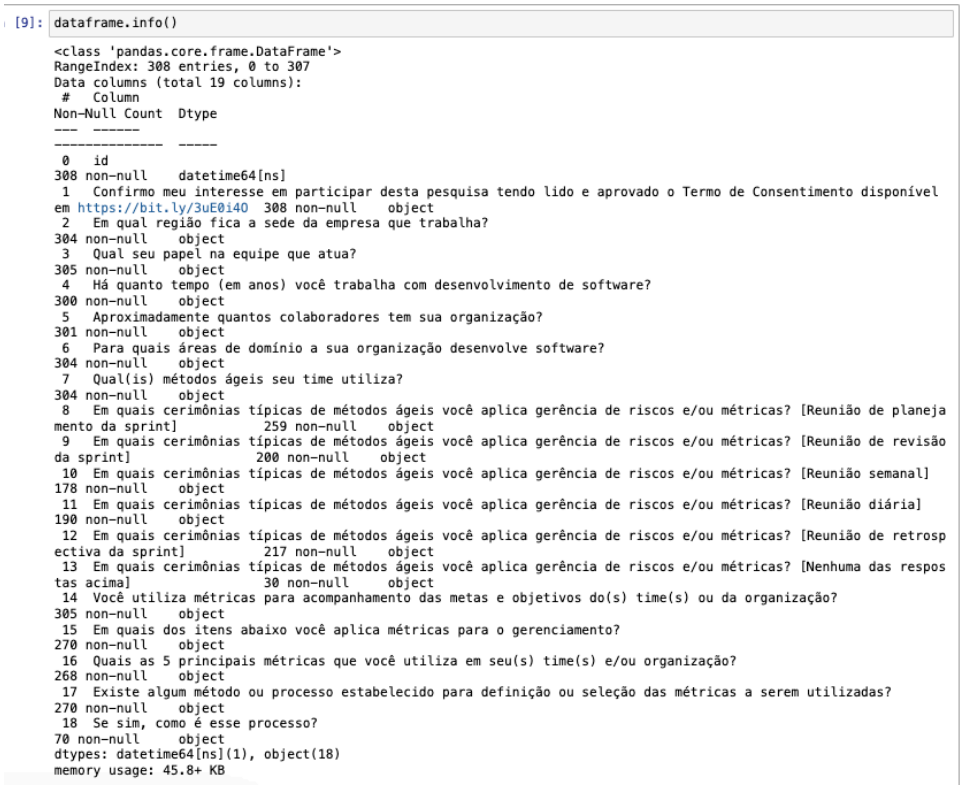
```
In [12]: dataframe = pd.read_excel('dados_brutos.xlsx', sheet_name='DADOS_TRATADOS')

In [13]: dataframe.head()
Out[13]:
```

	id	Confirmo meu interesse em participar desta pesquisa tendo lido e aprovado o Termo de Consentimento disponível em https://bit.ly/3uE0i4O	Em qual região fica a sede da empresa que trabalha?	Qual seu papel na equipe que atua?	Há quanto tempo (em anos) você trabalha com desenvolvimento de software?	Aproximadamente quantos colaboradores tem sua organização?	Para quais áreas de domínio a sua organização desenvolve software?	Qual(is) métodos ágeis seu time utiliza?	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de planejamento da sprint]	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de revisão da sprint]
0	2022-07-13 17:34:48.518	Sim	Sul	Product manager	0 a 5	Microempresa	energia	Scrum, Kanban	Métricas	NaN
1	2022-07-13 17:57:22.611	Sim	Sul	Desenvolvedor(a)	0 a 5	Microempresa	Financeiro	Scrum, Kanban	Métricas	Métricas
2	2022-07-13 20:50:58.250	Sim	Sul	Product manager	0 a 5	Pequena empresa	Eventos	Scrum	Gerenciamento de riscos	Métricas
3	2022-07-15 22:26:06.418	Sim	Sul	Team leader	6 a 9	Microempresa	gestão	Scrum, Kanban	Gerenciamento de riscos, Métricas	Gerenciamento de riscos G
4	2022-07-17 19:07:10.867	Sim	Sul	Product manager	0 a 5	Pequena empresa	Agronegocio	Scrum, Kanban	Gerenciamento de riscos, Métricas	Métricas

Figura 11: Leitura dos dados brutos com pandas e a execução do comando *dataframe.head()*.

A figura 11 acima demonstra a facilidade de trabalhar em combinação com *pandas* e *Jupyter Notebooks*. Ali é possível observar a leitura dos dados brutos a partir da fonte de dados fornecida em formato Excel. A execução do comando *dataframe.head()* dá uma visão inicial das cinco primeiras linhas da planilha, bem como todas as suas colunas, que no total são 19. É possível notar já que a maior parte dos dados já está normalizada, seguindo um padrão criado pela mestranda que realizou a primeira preparação do dataframe bruto, porém em formato de texto.

[1] https://pypi.org/project/pandas/
[2] https://jupyter.org/



```
[9]: dataframe.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308 entries, 0 to 307
Data columns (total 19 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   id
308 non-null    datetime64[ns]
 1   Confirmo meu interesse em participar desta pesquisa tendo lido e aprovado o Termo de Consentimento disponível
em https://bit.ly/3uE0i4O  308 non-null    object
 2   Em qual região fica a sede da empresa que trabalha?
304 non-null    object
 3   Qual seu papel na equipe que atua?
305 non-null    object
 4   Há quanto tempo (em anos) você trabalha com desenvolvimento de software?
300 non-null    object
 5   Aproximadamente quantos colaboradores tem sua organização?
301 non-null    object
 6   Para quais áreas de domínio a sua organização desenvolve software?
304 non-null    object
 7   Qual(is) métodos ágeis seu time utiliza?
304 non-null    object
 8   Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de planeja
mento da sprint]          259 non-null    object
 9   Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de revisão
da sprint]                200 non-null    object
 10  Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião semanal]
178 non-null    object
 11  Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião diária]
190 non-null    object
 12  Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de retrosp
ectiva da sprint]         217 non-null    object
 13  Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Nenhuma das respos
tas acima]                30 non-null     object
 14  Você utiliza métricas para acompanhamento das metas e objetivos do(s) time(s) ou da organização?
305 non-null    object
 15  Em quais dos itens abaixo você aplica métricas para o gerenciamento?
270 non-null    object
 16  Quais as 5 principais métricas que você utiliza em seu(s) time(s) e/ou organização?
268 non-null    object
 17  Existe algum método ou processo estabelecido para definição ou seleção das métricas a serem utilizadas?
270 non-null    object
 18  Se sim, como é esse processo?
70 non-null     object
dtypes: datetime64[ns](1), object(18)
memory usage: 45.8+ KB
```

Figura 12: execução do comando *dataframe.info().*

O comando *dataframe.info()* fornece um resumo dos dados. Interpretando a saída de sua execução, consegue-se identificar que existem 308 entradas (linhas) não nulas no dataframe, bem como a quantidade de cada tipo de dado. A coluna ID usada para identificar unicamente cada respondente foi a data e hora exata que sua resposta foi submetida, portanto, existem 308 colunas com o formato *datetime64[ns].* Todas as outras colunas foram interpretadas como sendo do tipo "*object*", que no contexto do *pandas* é um tipo de dado genérico que pode armazenar qualquer tipo de dado.

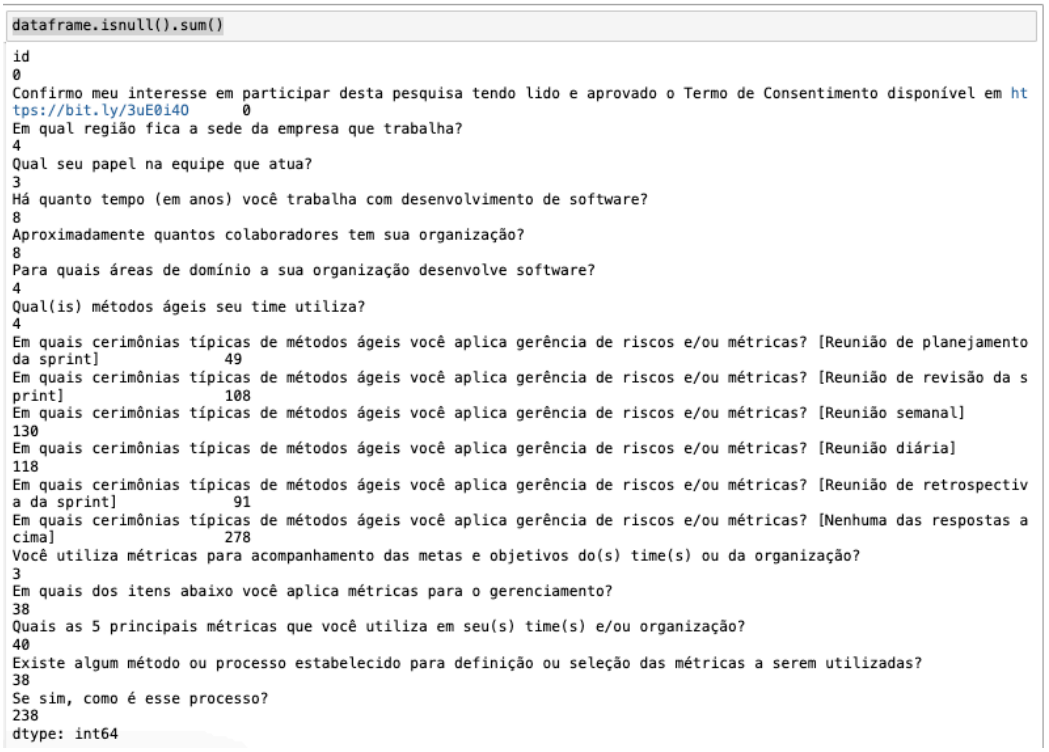


Figura 13: execução do comando *dataframe.isnull().sum().*

A excecução do comando *dataframe.isnull().sum()* dá a somatória de quantos valores ausentes estão em cada coluna, mostrado na Figura 13. Interpretando os resultados, percebe-se que existem alguns valores ausentes em determinadas colunas do dataframe, como 40 colunas sem resposta para a pergunta "Quais as 5 principais métricas que você utiliza em seu(s) time(s) e/ou organização?".

A normalização das métricas colhidas feitas no dataset original está separada em outra planilha. Cada *timestamp* único na coluna ID serve como referência para o dataset principal, pois cada respondente poderia fornecer até cinco métricas que utiliza em sua organização, numa única resposta. É possível visualizar esta estrutura na Figura 14 apresentada a seguir.

```
[16]: dataframe_metrics = pd.read_excel('dados_brutos.xlsx', sheet_name='MÉTRICAS DE-PARA')

[17]: dataframe_metrics.head()

[17]:
```

	ID BRUTO	DE BRUTO	id	ID BANCO (timestamp)	DE	PARA	DE.1	PARA.1
0	2.022071e+13	Prazo para entrega	2022-11-04 11:47:33	2.022110e+13	Valor monetário, de aprovação de transações, N...	Valor agregado	9box	9box
1	2.022071e+13	Valores financeiros	2022-11-04 11:47:33	2.022110e+13	Valor monetário, de aprovação de transações, N...	NPS	NaN	NaN
2	2.022071e+13	Recursos necessários	2022-11-04 11:47:33	2.022110e+13	Valor monetário, de aprovação de transações, N...	Cycle time	NaN	NaN
3	2.022071e+13	NPS, Burndown, Bugs por versão, Tickets/client...	2022-11-04 11:49:56	2.022110e+13	NPS, TMR, CSAT, TMI, BPS	NPS	A	NaN
4	2.022071e+13	Burndown da sprint, Throughput da sprint, Quan...	2022-11-04 11:49:56	2.022110e+13	NPS, TMR, CSAT, TMI, BPS	Cycle time	AUM	AUM

Figura 14: leitura da planilha de dados com as métricas.

3.2.2. Normalização dos dados

Neste tópico será abordada a forma que os dados brutos foram transformados e geraram o dataset pronto para começar a ser manipulado para o desenvolvimento do Sistema de Recomendação.

3.2.2.1. Transformação de dados

Para facilitar a manipulação dos dados e sua visualização, os IDs únicos originalmente feitos com o tipo de dado *datetime64[ns]* foram utilizados como base para a criação de uma nova coluna no dataset chamada "id_integer", que funcionará da mesma forma, mas facilitará a manipulação dos dados por ser um tipo de dado mais simples. Essa coluna foi criada diretamente no Excel, inicialmente adicionando um valor inteiro sequencial para cada linha do dataset original na nova coluna, e em seguida utilizando a função PROCV, que procura o valor idêntico em uma coluna e retorna outro valor em uma outra coluna. O resultado pode ser visto a seguir nas figuras 15 e 16.

Reading dataset with new int IDs

```
: dataframe_ids = pd.read_excel('dados_brutos_ids_int.xlsx', sheet_name='DADOS_TRATADOS')

: dataframe_ids.head()
```

0	17:34:49	1.0	Sim	Sul	Product manager	0 a 5	Microempresa	energia	Scrum, Kanban	Métricas	
1	17:57:23	2.0	Sim	Sul	Desenvolvedor(a)	0 a 5	Microempresa	Financeiro	Scrum, Kanban	Métricas	
2	20:50:58	3.0	Sim	Sul	Product manager	0 a 5	Pequena empresa	Eventos	Scrum	Gerenciamento de riscos	
3	22:26:06	4.0	Sim	Sul	Team leader	6 a 9	Microempresa	gestão	Scrum, Kanban	Gerenciamento de riscos, Métricas	Gerer
4	19:07:11	5.0	Sim	Sul	Product manager	0 a 5	Pequena empresa	Agronegocio	Scrum, Kanban	Gerenciamento de riscos, Métricas	

Figura 15: visualizando os dados iniciais do dataset com a nova coluna.

```
dataframe_ids_metrics = pd.read_excel('dados_brutos_ids_int.xlsx', sheet_name='MÉTRICAS DE-PARA')
```

```
dataframe_ids_metrics.head()
```

	ID BRUTO	DE BRUTO	ID	ID BANCO (timestamp)	DE	PARA	DE.1	PARA.1	id_integer	Unnamed: 9	Unnamed: 10	Unnamed: 11	Unnamed: 12
0	2.022071e+13	Prazo para entrega	11:47:33	2.022110e+13	Valor monetário, de aprovação de transações, N...	Valor agregado	9box	9box	190.0	1.0	NaN	Valor agregado	97.0
1	2.022071e+13	Valores financeiros	11:47:33	2.022110e+13	Valor monetário, de aprovação de transações, N...	NPS	NaN	NaN	190.0	1.0	NaN	NPS	NaN
2	2.022071e+13	Recursos necessários	11:47:33	2.022110e+13	Valor monetário, de aprovação de transações, N...	Cycle time	NaN	NaN	190.0	1.0	NaN	Cycle time	NaN
3	2.022071e+13	NPS, Burndown, Bugs por versão, Tickets/client...	11:49:56	2.022110e+13	NPS, TMR, CSAT, TMI, BPS	NPS	A	NaN	256.0	1.0	NaN	CSAT	NaN
4	2.022071e+13	Burndown da sprint, Throughput da sprint, Quan...	11:49:56	2.022110e+13	NPS, TMR, CSAT, TMI, BPS	Cycle time	AUM	AUM	256.0	1.0	NaN	BPS	NaN

Figura 16: visualizando os dados iniciais de métricas com a nova coluna.

O próximo passo foi remover as respostas "Não" para a Pergunta 14 do *survey*: "Você utiliza métricas para acompanhamento das metas e objetivos do(s) time(s) ou da organização?", uma vez que respondentes que não utilizam métricas não servirão para a montagem final do dataset para o propósito deste projeto. Posteriormente, essa coluna será removida.

3.2.2.2. Tratamento de Colunas

Para facilitar mais a interpretação dos dados e sua leitura para posterior manipulação, algumas colunas foram removidas para manter no dataset final somente o que será necessário para a criação do Sistema de Recomendação. Utilizando a função *drop()* é possível remover determinadas colunas do dataset, e a combinando-a com o parâmetro "inplace" setado para verdadeiro, o dataframe original é modificado, sem a necessidade de declarar novas variáveis. Para a planilha que está sendo utilizada como apoio e que mantém as respostas das métricas normalizadas, as colunas removidas podem ser visualizadas na Figura 17, bem como o resultado.

Removing columns that will not be used

```
# removing columns from the metrics dataframe
dataframe_ids_metrics.drop(columns=['ID BRUTO',
                                    'DE BRUTO',
                                    'ID',
                                    'ID BANCO (timestamp)',
                                    'DE.1',
                                    'PARA.1',
                                    'Unnamed: 9',
                                    'Unnamed: 10',
                                    'Unnamed: 11',
                                    'Unnamed: 12'
                                    ], inplace=True)
```

```
dataframe_ids_metrics.head()
```

	DE	PARA	id_integer
0	Valor monetário, de aprovação de transações, N...	Valor agregado	190.0
1	Valor monetário, de aprovação de transações, N...	NPS	190.0
2	Valor monetário, de aprovação de transações, N...	Cycle time	190.0
3	NPS, TMR, CSAT, TMI, BPS	NPS	256.0
4	NPS, TMR, CSAT, TMI, BPS	Cycle time	256.0

Figura 17: removendo colunas da planilha de métricas

Na planilha com os dados brutos originais, as colunas que foram julgadas como não necessárias para obter resultados significativos para o Sistema de Recomendação estão apresentadas no Quadro 2 a seguir. Para sua remoção, o mesmo comando *drop()* foi executado, dessa vez para cada respectiva coluna.

Pergunta 1	Confirmo meu interesse em participar desta pesquisa tendo lido e aprovado o Termo de Consentimento disponível em https://bit.ly/3uE0i4O
Pergunta 2	Em qual região fica a sede da empresa que trabalha?
Pergunta 6	Para quais áreas de domínio a sua organização desenvolve software?
Pergunta 13	Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Nenhuma das respostas acima]
Pergunta 14	Você utiliza métricas para acompanhamento das metas e objetivos do(s) time(s) ou da organização?
Pergunta 17	Existe algum método ou processo estabelecido para definição ou seleção das métricas a serem utilizadas?
Pergunta 18	Se sim, como é esse processo?

Quadro 2: colunas removidas do dataset. Fonte: elaborada pelo autor.

Após isso, as colunas foram renomeadas para ser mais simples seu entendimento para o desenvolvimento. O Quadro 3 a seguir demonstra quais colunas foram renomeadas e sua nova nomenclatura.

Coluna Original	Coluna Renomeada
Qual seu papel na equipe que atua?	role
Há quanto tempo (em anos) você trabalha com desenvolvimento de software?	years_exp
Aproximadamente quantos colaboradores tem sua organização?	org_size
Qual(is) métodos ágeis seu time utiliza?	agile_methods
Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de planejamento da sprint]	use_metrics_planning
Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de revisão da sprint]	use_metrics_review
Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião semanal]	use_metrics_weekly
Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião diária]	use_metrics_daily
Em quais cerimônias típicas de métodos ágeis você aplica gerência de riscos e/ou métricas? [Reunião de retrospectiva da sprint]	use_metrics_retro
Em quais dos itens abaixo você aplica métricas para o gerenciamento?	metrics_category
Quais as 5 principais métricas que você utiliza em seu(s) time(s) e/ou organização?	sanitized_metrics

Quadro 3: renomeação de colunas. Fonte: elaborada pelo autor.

3.2.2.3. Primeira abordagem de *One Hot Encoding*

No início do desenvolvimento e da análise exploratória de dados, o método de criação do modelo do Sistema de Recomendação havia sido pensado para ser desenvolvido utilizando somente Redes Neurais. Para isso, o treinamento deveria ser facilitado, representando de forma matemática os dados rotulados do dataset. Uma forma simples de atingir este objetivo é o *One Hot Encoding* (OHE), uma técnica comum para converter dados categóricos em formato numérico (SOUZA, 2022). Em OHE, cada categoria ou coluna é representada de forma binária, sendo 1 o valor correspondente a categoria e 0 a todas as outras.

Para ilustrar como o OHE foi aplicado ao dataset original, será utilizada a coluna "agile_methods" como referência. Existiam sete possíveis respostas nesta coluna no dataset original: Scrum, Kanban, ScrunBan, Lean, XP, Safe e XGH. Todas estas respostas são referentes a métodos ágeis que os respondentes utilizam em suas organizações. A partir disso, foi preciso derivar essa coluna para a criação de novas

colunas: *method_scrum, method_kanban, method_scrumban, method_lean, method_xp, method_safe, method_xgh*. Após isso, caso o respondente utilize um dos métodos, a nova coluna é preenchida com o valor 1, do contrário, a coluna é preenchida com 0. O trecho de código adaptado que gerou estas novas colunas e realizou o OHE pode ser visto a seguir.

```
import pandas as pd

# #
# [... trechos de código omitido]
# #

def update_method_columns(df, methods, prefix="method_"):
   for method in methods:
       # Criar ou atualizar a coluna para cada método
       col_name = f"{prefix}{method}"
       # Verifica se o método está em alguma das colunas e atribui 1
se verdadeiro, 0 se falso
       df[col_name] = (df[methods_columns].apply(lambda x:
x.str.lower() == method, axis=1).any(axis=1)).astype(int)

# #
# [... trechos de código omitido]
# #
```

Código 1: Trecho de código adaptado para uma parte do OHE

Esta abordagem foi realizada ao longo de todas as colunas do dataset, com suas particularidades. O resultado parcial pode ser visto a seguir.

```
In [55]: dataframe_ohe = pd.read_excel('OHE.xlsx', sheet_name='OHE')

In [56]: dataframe_ohe.head()
Out[56]:
```

	id	role	years_exp	org_size	use_metrics_planning	use_metrics_review	use_metrics_weekly	use_metrics_daily	use_metrics_retro	use_metrics_na	...	metr
0	1	1.0	1	1	0	1	1	0	1	0	...	
1	2	2.0	1	1	1	0	0	0	1	0	...	
2	3	3.0	1	1	1	1	0	0	1	0	...	
3	4	4.0	2	1	1	1	1	0	1	1	...	
4	5	3.0	1	1	0	1	0	0	1	1	...	

5 rows × 110 columns

Figura 18: Data Frame com One Hot Encoding parcial.

Os dados brutos e o estado atual do dataset estão disponíveis através do link https://drive.google.com/drive/folders/1RfOpLwYV6TvJgdPFJ7frAydKOH-ZxUxe?usp=sharing. Outros recursos disponibilizados pelo autor, como os *Jupyter Notebooks* utilizados até agora e o estudo de métricas extraídas fornecidas por Leal também estão presentes.

3.3. Implementação Inicial

Para o escopo deste trabalho, serão desenvolvidos dois modelos de recomendação com duas técnicas diferentes: um baseado em filtragem colaborativa, outro de categorização *multi-label*. Por fim, serão avaliados os resultados dos dois modelos por meio de métricas qualitativas e testes manuais através da interface disponibilizada pelo sistema web de recomendação.

O ambiente de desenvolvimento do sistema de recomendação será o *Jupyter Notebook*, previamente utilizado para a análise exploratória e transformação dos dados brutos, pela facilidade de experimentação e interação com o código e por ter um ambiente de documentação integrado. Para a manipulação dos dados, o *pandas* que também foi utilizado na análise exploratória, continuará sendo utilizado, uma vez que fornece estruturas de dados flexíveis e expressivas, como *DataFrames*.

A biblioteca de aprendizado de máquina *Scikit-learn*[3] será utilizada para modelar o sistema de recomendação. Esta biblioteca oferece diversas ferramentas e algoritmos para modelagem estatística e análise de dados, e facilita a implementação de algoritmos de aprendizado de máquina e técnicas de processamento de dados. Outras bibliotecas auxiliares poderão vir a ser utilizadas, como por exemplo *NumPy,*[4] biblioteca em linguagem de programação *Python* amplamente utilizada para a realização de cálculos científicos e *Matplotlib*[5]*,* biblioteca de plotagem 2D em *Python* que permite criar visualizações estáticas, animadas e interativas de dados. Para salvar os modelos e poder carregá-los para usá-los posteriormente, será utilizada a biblioteca *Joblib*[6], bem difundida em tarefas de aprendizado de máquina, pois possibilita a para serialização de objetos grandes.

Todo o sistema de recomendação será desenvolvido em linguagem de programação *Python.* O usuário utilizará o Sistema de Recomendação por meio de um

[3] https://scikit-learn.org/stable/
[4] https://numpy.org/
[5] https://matplotlib.org/
[6]https://joblib.readthedocs.io/en/stable/

Sistema Web, separado em *backend* e *frontend*. O *backend* foi desenvolvido utilizando *Python* com a biblioteca *Flask*[7] que permite criar aplicações web de forma simplificada, enquanto o *frontend* foi desenvolvido em forma de *Single Page Application* utilizando a biblioteca JavaScript *React*[8] para construir a interface de uso do Sistema de Recomendação.

3.3.1. Exemplo de Recomendação Baseada em Conteúdo

Esta seção detalha um exemplo simples de um sistema de recomendação baseado em conteúdo, para ilustrar como as tecnologias escolhidas funcionam na prática. Foi escolhido trabalhar com uma quantidade limitada de dados, e os dados foram criados pelo autor, para facilitar a interpretação. O sistema de exemplo deseja encontrar a similaridade entre diferentes filmes baseado no seu gênero, para posteriormente ser utilizado o índice de similaridade para recomendar filmes para diferentes usuários.

Title	Genres
The Matrix	Action Sci-Fi
Titanic	Romance Drama
The Avengers	Action Adventure Sci-Fi
Shrek	Animation Comedy
Toy Story	Animation Adventure Comedy
Avatar	Action Adventure Sci-Fi
The Dark Knight	Action Drama

Quadro 4: dados de exemplo.

As bibliotecas utilizadas neste exemplo foram o *pandas* e o *Scikit-learn.* Após a criação dos dados e sua leitura, o próximo passo é converter os gêneros dos filmes em uma representação numérica que possa ser utilizada para calcular similaridades. Para isso, utilizamos a técnica TF-IDF *(Term Frequency-Inverse Document Frequency)*, que transforma os gêneros textuais em vetores numéricos.

[7] https://flask.palletsprojects.com/en/stable/
[8] https://react.dev/

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words='english') tfidf_matrix
= tfidf.fit_transform(df['genres'])
tfidf_matrix.shape # Verifica as dimensões da matriz
```

Código 2: Computando a matriz TF-IDF (adaptado)

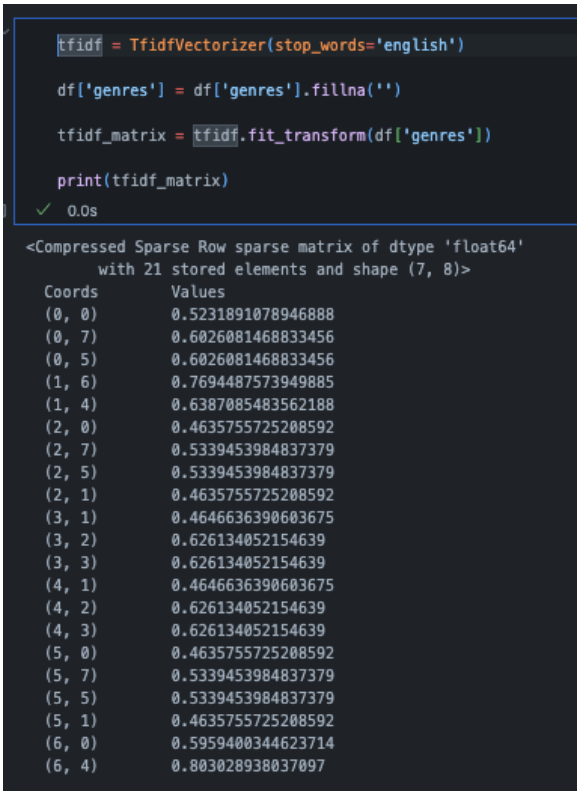


Figura 19: Saída da matriz de vetores calculada

Com a matriz TF-IDF calculada, o próximo passo é computar a similaridade cosseno entre todos os pares de filmes. A similaridade cosseno é uma métrica que mede o ângulo entre dois vetores, fornecendo uma medida de quão similares eles são. Após esta etapa, os valores de similaridade já estão computados e é possível ter um sistema de recomendação baseado em conteúdo funcional que pode sugerir filmes similares com base nos gêneros. Este sistema pode ser facilmente expandido e refinado com mais dados e ajustes nas técnicas de processamento de texto e cálculo de similaridade.

```
from sklearn.metrics.pairwise import linear_kernel

cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

Código 3: Calculando a similaridade cosseno (adaptado)

A fim de testar a implementação, foi simulada a execução do modelo. Foram comparadas as similaridades dos filmes "Shrek" e "The Matrix" com os outros filmes do dataset. Os resultados mostraram que "The Avengers" e "Avatar" são altamente similares a "The Matrix", com pontuações de similaridade de aproximadamente 0.8565, indicando que compartilham muitos gêneros em comum, como "Action" e "Sci-Fi". Em contraste, "Titanic" e "Shrek" não mostraram similaridade com "The Matrix", refletindo seus gêneros distintos. Para "Shrek", "Toy Story" foi altamente similar, com uma pontuação de 0.8558, devido aos gêneros compartilhados de "Animation" e "Comedy", enquanto filmes como "The Matrix" e "Titanic" não apresentaram similaridade. Os resultados podem ser observados no Quadro 5.

Filme	Filme Recomendado	Similaridade
The Matrix	The Avenger	85,65%
The Matrix	Avatar	85,65%
The Matrix	The Dark Knight	31,18%
The Matrix	Titanic	0,00%
The Matrix	Shrek	0,00%
Shrek	Toy Story	85,58%
Shrek	The Matrix	0,00%
Shrek	Titanic	0,00%
Shrek	The Avengers	0,00%
Shrek	Avatar	0,00%

Quadro 5: similaridades entre os filmes computadas pelo sistema de recomendação

4. DESENVOLVIMENTO

Este capítulo trata do processo de desenvolvimento dos dois modelos de recomendação, a detecção de remoção de outliers do dataset, o pré-processamento, os problemas encontrados e as alterações necessárias para conseguir completar o desenvolvimento, e por fim o desenvolvimento do Sistema Web para o uso do Sistema de Recomendação.

Todos os *datasets* que serviram de base para o desenvolvimento e os *datasets* manipulados no estado que se encontraram após cada etapa de pré-processamento podem ser encontrados emi: https://doi.org/10.17632/st64hhxhhy.1. O código fonte para os Sistemas de Recomendação e o Sistema Web podem ser encontrados em https://codigos.ufsc.br/gqs/sistema-recomendacao-de-metricas.

4.1. Tratamento de Outliers

Outliers são pontos de dados que se desviam significativamente do restante das observações em um dataset. Segundo Ghosh e Vogt (2012), um outlier em uma amostra de pesquisa é uma observação que está muito distante da maioria ou de todas as outras observações. Esses pontos, quando não tratados, podem distorcer modelos preditivos, especialmente em sistemas de recomendação, levando a resultados enviesados ou não confiáveis. Nesta seção, serão apresentadas as abordagens utilizadas para detectar e remover os outliers presentes no dataset original.

4.1.1. Detecção dos outliers

Ao analisar o dataset em seu estado antes da remoção dos outliers, foi determinado que os possíveis dados que iriam influenciar negativamente nos resultados dos modelos de recomendação estariam essencialmente nas respostas contidas na coluna “*sanitized_metrics*”, onde os participantes do *survey* responderam quais métricas ágeis utilizam nas suas organizações, e na coluna de “*agile_methods*”, em que os respondentes informaram quais métodos ágeis utilizam, uma vez que as demais respostas já estavam categorizadas e agrupadas no dataset original.

4.1.2. Remoção dos outliers

Manipulando o dataset utilizando as bibliotecas *Pandas* e *Matplotlib* foi possível obter o resultado gráfico da frequência dos valores contidos em métodos ágeis. O resultado da plotagem gráfica pode ser visualizado na Figura 20 a seguir.

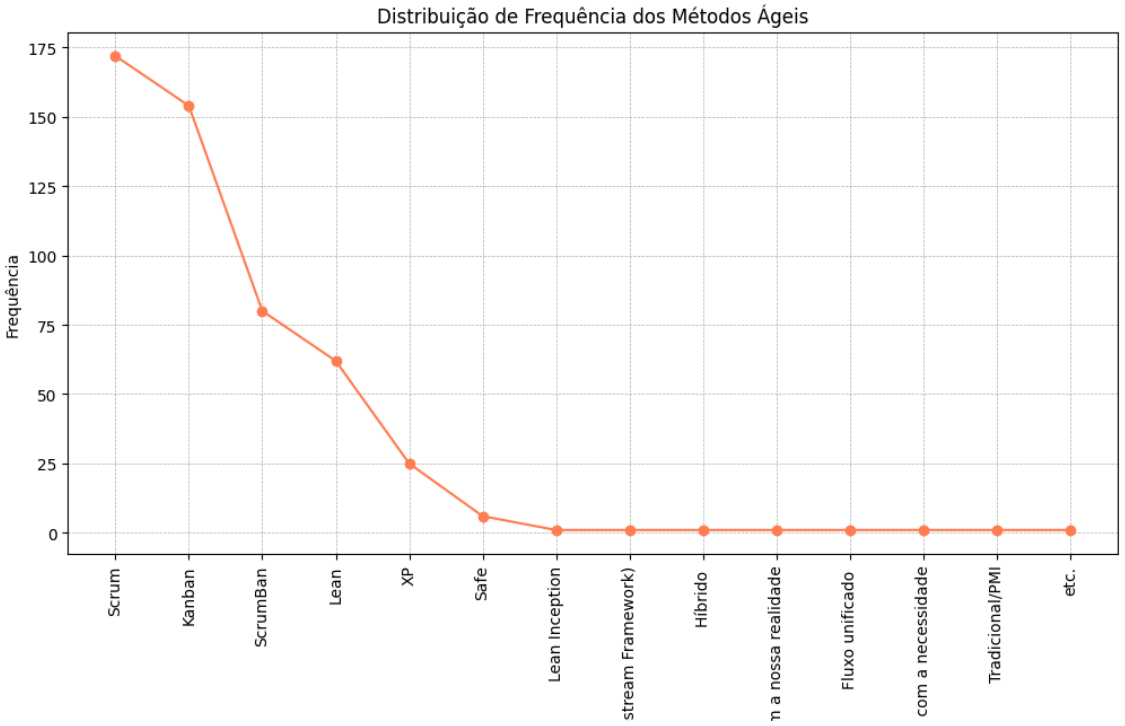


Figura 20: Gráfico da frequência de métodos ágeis. Elaborado pelo autor.

Ao observar o gráfico, nota-se que a maior parte das respostas obtidas para os métodos ágeis utilizados têm uma frequência relativamente alta, com média aproximada de 36, e que os métodos ágeis que menos se repetem sempre tem uma frequência igual a 1. Portanto, o critério para remoção de outliers na coluna "*agile_methods*" foi remover todas as respostas que contêm valores de frequência menores que 2. O resultado obtido após a remoção pode ser visualizado no gráfico abaixo, na Figura 21.

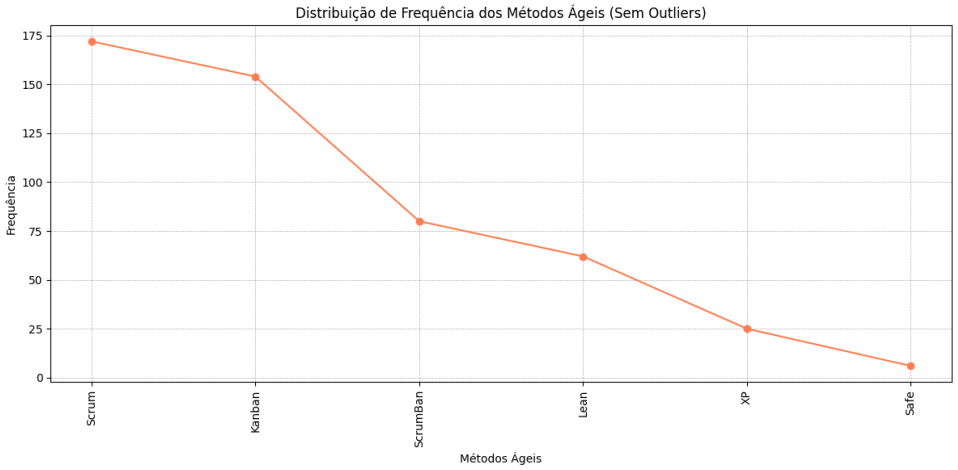


Figura 21: Gráfico da frequência de métodos ágeis sem outliers. Elaborado pelo autor.

O critério definido para determinar se uma métrica era outlier ou não foi a métrica estar presente no dataset de métricas extraídas da literatura, elaborado no trabalho de Leal “Desenvolvimento de um modelo de seleção e avaliação de métricas de software para contextos ágeis” (LEAL, 2023). A métrica que não estivesse presente no estudo sistemático feito pela autora foi removida do dataset. Esta abordagem foi feita para que a recomendação utilizando o sistema de classificação multi-label pudesse se basear nas informações contidas sobre cada categoria de métrica para fornecer as recomendações, além de disponibilizar ao usuário mais detalhes de uso de cada métrica, uma vez que a maior parte das métricas contidas no dataset de Leal possuem um descrição robusta.

4.2. Modelo de Recomendação de Filtragem Colaborativa

O modelo de recomendação de Filtragem Colaborativa foi desenvolvido para considerar a semelhança entre o perfil do usuário do Sistema de Recomendação com os perfis presentes no dataset e recomendar as métricas utilizadas por estes perfis, apresentando a similaridade entre o perfil do usuário e o perfil do respondente do *survey* identificado no dataset, utilizando similaridade dos cossenos para fazer este cálculo. A similaridade de cossenos mede o ângulo entre dois vetores no espaço de características, proporcionando uma medida de quão semelhantes são os perfis (NGUYEN; AMER, 2019).

O Quadro 6 representa, de forma adaptada, as perguntas que foram utilizadas para elaborar este modelo e definir o perfil do usuário, além da quantidade máxima de recomendações de métricas que o usuário deseja.

Pergunta	Coluna no dataset
Função na organização	*role*
Anos de experiência	*years_exp*
Tamanho da organização	*org_size*
Métodos ágeis utilizados	*agile_methods*
Rituais ágeis utilizados	*use_metrics_**
Onde gostaria de aplicar métricas	*metrics_category*
Quantidade máxima de recomendações	N/A

Quadro 6: perguntas utilizadas para definir o perfil do usuário do sistema de recomendação de filtro colaborativo. Elaborado pelo autor.

O desenvolvimento do modelo de recomendação de filtragem colaborativa pode ser dividido em duas etapas, pré-processamento e implementação, apresentados a seguir.

4.2.1. Pré-processamento

A primeira etapa do pré-processamento do dataset, se constituiu em traduzir todas as respostas que existiam nas colunas *'use_metrics_planning', 'use_metrics_review', 'use_metrics_weekly', 'use_metrics_daily', 'use_metrics_retro'*, de forma binária, com “Sim” ou “Não”, para representar em qual ritual ágil o respondente do survey utiliza métricas. No dataset original, o valor preenchido caso o respondente utilize métricas num destes rituais estava preenchido como “Métricas”, este valor foi substituído por “Sim", e “Não” caso fosse ausente. A Figura 22 demonstra a tradução mencionada.

```
columns_to_update = ['use_metrics_planning', 'use_metrics_review', 'use_metrics_weekly', 'use_metrics_daily', 'use_metrics_retro']

for column in columns_to_update:
    df[column] = df[column].apply(lambda x: 'Sim' if x == 'Métricas' else 'Não' if x == '' else x)

df.head()
```

use_metrics_planning	use_metrics_review	use_metrics_weekly	use_metrics_daily	use_metrics_retro	metrics_category	sanitized_metrics	agile
Sim	Não	Sim	Não	Não	Cronograma e progresso, Produto, Tecnologia, C...	Ideal team capacity	
Sim	Sim	Não	Não	Não	Cronograma e progresso, Produto, Cliente	NPS, Burndown, Outstanding bugs, Throughput, N...	
Não	Sim	Não	Não	Sim	Cronograma e progresso, Pessoas, Produto	Burndown, OKR, Quantidade de solicitações de c...	
Sim	Não	Não	Não	Sim	Cronograma e progresso, Produto, Cliente	CAC, Throughput, Roadmaps, Conversão	
Sim	Sim	Não	Não	Sim	Cronograma e progresso	User Story Points, NPS	

Figura 22: trecho de código do pré-processamento.

Para obter mais resultados no dataset e multiplicar os perfis para construção do modelo de recomendação, a função *explode()* da biblioteca *Pandas* expande a coluna *matched_metrics,* que contém as métricas sem outliers, para que cada entrada dessa coluna esteja em uma linha distinta. A Figura 23 contém um trecho de código adaptado e a saída resultante no dataset após aplicar esta técnica.

```
df = df.explode('matched_metrics')
df.head()
```

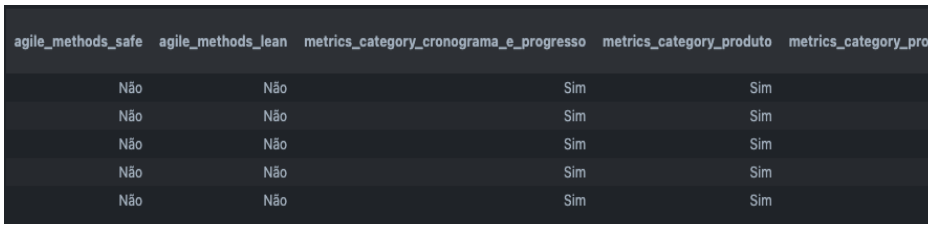
	Carimbo de data/hora	id_integer	role	years_exp	org_size	agile_methods	use_metrics_planning	u
0	17:34:49	1	Product manager	0 a 5	Microempresa	Scrum, Kanban	Sim	
1	17:57:23	2	Desenvolvedor(a)	0 a 5	Microempresa	Scrum, Kanban	Sim	
1	17:57:23	2	Desenvolvedor(a)	0 a 5	Microempresa	Scrum, Kanban	Sim	
1	17:57:23	2	Desenvolvedor(a)	0 a 5	Microempresa	Scrum, Kanban	Sim	
1	17:57:23	2	Desenvolvedor(a)	0 a 5	Microempresa	Scrum, Kanban	Sim	

Figura 23: trecho adaptado demonstrando o uso do método de explosão do dataset.

Para as respostas contidas em “*agile_methods*” e “*metrics_category*", foi realizada a expansão do dataset e criação de novas colunas binárias atribuindo “Sim” para a presença do método ágil e da categoria de métrica na resposta do survey, e não para sua ausência.

```
agile_methods_list = ['Scrum', 'Kanban', 'ScrumBan', 'XP', 'Safe',
'Lean']
metrics_category_list = ['Cronograma e progresso', 'Produto', 'Processo',
'Tecnologia', 'Pessoas', 'Cliente']


# Função para verificar se um valor está presente na lista da coluna
def expand_column(df, column, values_list, prefix):
   for value in values_list:
       column_name = f'{prefix}_{value.lower().replace(" ", "_")}'
       df[column_name] = df[column].apply(lambda x: 'Sim' if value in
str(x) else 'Não')
       if column_name not in df.columns:
           print(f"Erro: coluna {column_name} não foi criada.")
   return df

df = expand_column(df, 'agile_methods', agile_methods_list,
'agile_methods')

df = expand_column(df, 'metrics_category', metrics_category_list,
'metrics_category')
```

Código 4: expansão de colunas.



agile_methods_safe	agile_methods_lean	metrics_category_cronograma_e_progresso	metrics_category_produto	metrics_category_pro
Não	Não	Sim	Sim	
Não	Não	Sim	Sim	
Não	Não	Sim	Sim	
Não	Não	Sim	Sim	
Não	Não	Sim	Sim	

Figura 24: ilustração do resultado da expansão de colunas.

Como a implementação do modelo utiliza a técnica TF-IDF para analisar as respostas do survey e determinar a semelhança do perfil, todos os valores do dataset devem estar em formato textual. Para isso, todos os valores nulos e de outros tipos foram convertidos para *string*, como demonstrado no trecho de código adaptado a seguir.

```
df = df.dropna(subset=['matched_metrics'])
df = df.fillna('')
```

Código 5: adaptação de trecho do pré-processamento para transformação de dados.

O pré-processamento resultou em um dataset novo, pronto para ser utilizado para a implementação do modelo de recomendação de filtragem colaborativa.

4.2.2. Implementação

A implementação do modelo de filtro colaborativo utilizou da biblioteca *Scikit-learn* as ferramentas TF-IDF para criação de vetores de texto e cálculo de similaridade cosseno, a biblioteca *numpy* para operações numéricas e a biblioteca *joblib* para salvar o modelo criado.

4.2.2.1. Definição de features

A primeira etapa da implementação foi a definição dos atributos que descrevem as características do perfil – em aprendizado de máquina denominado de *features* – que vão ser consideradas para calcular a similaridade entre perfis. As features extraídas do dataset e seus tipos são apresentados no Quadro 7.

Coluna	Tipo
role	Textual
years_exp	Textual
org_size	Textual
use_metrics_planning	Binário
use_metrics_review	Binário
use_metrics_weekly	Binário
use_metrics_daily	Binário
use_metrics_retro	Binário
agile_methods_scrum	Binário
agile_methods_kanban	Binário
agile_methods_scrumban	Binário
agile_methods_xp	Binário
agile_methods_safe	Binário
agile_methods_lean	Binário
metrics_category_cronograma_e_progresso	Binário
metrics_category_produto	Binário
metrics_category_processo	Binário
metrics_category_tecnologia	Binário
metrics_category_pessoas	Binário
metrics_category_cliente	Binário

Quadro 7: Features e seus valores.

4.2.2.2. Vetorização dos valores

A seguir, para cada feature é criada uma representação numérica, utilizando TF-IDF para os valores textuais e convertendo diretamente as colunas binárias. Os trechos de código serão analisados ponto a ponto.

```
tfidf_matrices = {}
tfidf_vectorizers = {}
```

Código 6: dicionários de representações numéricas.

Os dicionários `tfidf_matrices` e `tfidf_vectorizers` são criados para armazenar as representações vetoriais das features e os objetos de transformação utilizados. O dicionário `tfidf_matrices` armazena as matrizes de vetores TF-IDF para features textuais ou os valores binários para variáveis categóricas enquanto o dicionário `tfidf_vectorizers` armazena os vetorizadores *`TfidfVectorizer`* aplicados a cada feature textual, permitindo a reutilização do mesmo processo de vetorização para novos dados, garantindo assim consistência na aplicação do modelo.

```
for feature in features:
    if df[feature].dtype == 'object':  #
        tfidf_vectorizer = TfidfVectorizer()
        tfidf_matrices[feature] = tfidf_vectorizer.fit_transform(df[feature])
        tfidf_vectorizers[feature] = tfidf_vectorizer

    else:
        # Para colunas binárias, converte diretamente em matriz numérica
        tfidf_matrices[feature] = df[[feature]].values
```

Código 7: aplicação da transformação numérica dos valores.

Para features cujo tipo de dados é `object` (indicando uma variável textual), é instanciado um objeto `TfidfVectorizer`, que calcula a ponderação TF-IDF de cada termo presente na coluna. A transformação `fit_transform` gera uma matriz esparsa TF-IDF, onde cada linha representa uma observação (por exemplo, um perfil) e cada coluna corresponde a um termo único da coluna textual. A magnitude de cada elemento na matriz reflete o peso TF-IDF do termo para a observação específica, capturando, assim, a relevância e a especificidade do termo. Essa matriz TF-IDF é então armazenada em `tfidf_matrices[feature]`, com a chave sendo o nome da feature correspondente. O vectorizer é salvo em `tfidf_vectorizers[feature]`, permitindo a aplicação do mesmo modelo de transformação TF-IDF em novos dados, mantendo a coerência semântica e dimensionalidade para entradas adicionais. O

TF-IDF calcula a relevância dos termos em um *corpus* com base em dois componentes: Term Frequency (TF), que mede a frequência de ocorrência de um termo em um documento, e Inverse Document Frequency (IDF), que atribui menor peso a termos comuns e maior peso a termos específicos, conferindo maior discriminação entre observações. Para colunas não textuais, o código assume que são variáveis categóricas binárias e, portanto, converte os valores diretamente em uma matriz numérica. Isso assegura uma representação consistente de variáveis binárias, onde os valores mantêm sua interpretação original ("Sim"/"Não"). Essa matriz numérica é armazenada em `tfidf_matrices[feature]` para uso posterior na etapa de similaridade.

4.2.2.3. Função de Cálculo de Similaridade

A função `calculate_total_similarity` calcula a similaridade ponderada entre um perfil e todos os outros perfis no conjunto de dados. Inicialmente, `total_similarity` é definido como um vetor de zeros, representando a similaridade acumulada entre o perfil de interesse e os demais. Para cada feature, é calculado o valor da similaridade cosseno usando as matrizes TF-IDF previamente geradas para valores textuais e diretamente sobre os valores binários. Caso o valor da feature seja "Sim" no perfil de interesse, o peso da feature é duplicado, amplificando a contribuição dessa característica na similaridade total, para garantir que valores presentes no perfil do usuário serão levados em conta com mais intensidade. A função aceita um dicionário de pesos (`feature_weights`), que permite ajustar a influência de cada feature, e pode ser explorado no futuro para permitir que o usuário do Sistema de Recomendação informe quais features devem ser mais levadas em conta para uma recomendação mais customizada.

```
def calculate_total_similarity(profile, feature_weights=None):
    total_similarity = np.zeros(df.shape[0])

    # Definir pesos padrão para cada feature se não forem fornecidos
    if feature_weights is None:
        feature_weights = {feature: 1.0 for feature in features}  # Pesos iguais

    # Loop sobre cada feature para calcular sua similaridade e aplicar o peso
    for feature in features:
        if df[feature].dtype == 'object':
            # Similaridade com TF-IDF para colunas textuais
            cosine_sim = cosine_similarity(tfidf_matrices[feature][profile],
tfidf_matrices[feature]).flatten()
        else:
            # Similaridade para colunas binárias
```

```
            cosine_sim = cosine_similarity([tfidf_matrices[feature][profile]],
tfidf_matrices[feature]).flatten()

            # Aplicar peso maior para colunas binárias com valor 1
            if df[feature].iloc[profile] == 1:
                feature_weights[feature] *= 2.0  # Aumenta o peso se for 1 (ou
"Sim")

        # Amplificar o impacto dos pesos para as features binárias e métricas
importantes
        total_similarity += cosine_sim * feature_weights[feature]

    return total_similarity
```

Código 8: função para calcular a similaridade dos perfis (adaptado).

4.2.2.4. Recomendação de Métricas

A função `recommend_metrics` usa a similaridade total calculada para identificar os perfis mais semelhantes ao perfil de interesse. A função ordena os perfis em ordem decrescente de similaridade, coleta as métricas dos perfis mais similares, evitando duplicatas e limitando a recomendação ao número de métricas explicitadas pelo usuário. Esse processo é iterado até que o limite de métricas únicas sejam identificadas, promovendo a diversificação nas sugestões de métricas.

```
def recommend_metrics(profile, top_n=5, feature_weights=None):
   # Calcular a similaridade total ponderada
   total_similarity = calculate_total_similarity(profile, feature_weights)

   # Obter os índices dos perfis mais similares
   similar_indices = total_similarity.argsort()[-(top_n+1):-1]
   # Obter as similaridades dos perfis selecionados
   similar_profiles = df.iloc[similar_indices]
   similar_affinity = total_similarity[similar_indices]

   # Exibir as afinidades diretamente sem normalização
   recommended_metrics = []

   # Loop para recomendar as métricas dos perfis mais similares
   for idx, metrics in enumerate(similar_profiles['matched_metrics']):
       if metrics not in [rec[0] for rec in recommended_metrics]:
           affinity = similar_affinity[idx]  # Usar similaridade absoluta, sem
normalização
           recommended_metrics.append((metrics, affinity, similar_indices[idx]))

       # Parar após encontrar limite de métricas diferentes
       if len(recommended_metrics) == top_n:
           break

   return recommended_metrics
```

Código 9: função para recomendar métricas (adaptado).

4.2.2.5. Ajuste de Pesos

Durante o desenvolvimento do sistema, verificou-se que os resultados de similaridade e recomendação melhoraram significativamente ao ajustar os pesos de certas features, como métodos ágeis (`agile_methods_*`) e categorias de métricas. A atribuição de pesos mais elevados a essas variáveis específicas elevou a similaridade entre perfis com características essenciais semelhantes, o que, por sua vez, gerou recomendações de métricas mais precisas e relevantes para o perfil de interesse. Para identificar a configuração ideal, diversos valores de pesos foram testados até que os resultados de similaridade fossem considerados satisfatórios. Quando os valores de similaridade atingiram níveis elevados e as métricas recomendadas refletiram com precisão as características desejadas, os pesos foram então definidos com valores específicos, como 3.5 para variáveis de planejamento, revisão e retrospectiva, e 5.0 para métodos ágeis e categorias de métricas. A definição de pesos teve caráter extremamente experimental, onde cada peso foi ajustado de acordo com os resultados de recomendação obtidos, até se tornarem satisfatórios.

Os pesos definidos para um modelo inicial foram os seguintes:

```
feature_weights = {
    'role': 0.5,      'years_exp': 0.5,
    'org_size': 0.5,
    'use_metrics_planning': 3.5,
    'use_metrics_review': 3.5,
    'use_metrics_weekly': 3.5,
    'use_metrics_daily': 3.5,
    'use_metrics_retro': 3.5,
    'agile_methods_scrum': 5.0,
    'agile_methods_kanban': 5.0,
    'agile_methods_scrumban': 5.0,
    'agile_methods_lean': 5.0,
    'agile_methods_xp': 5.0,
    'agile_methods_safe': 5.0,
    'metrics_category_cronograma_e_progresso': 3.5,
    'metrics_category_produto': 3.5,
    'metrics_category_processo': 3.5,
    'metrics_category_tecnologia': 3.5,
    'metrics_category_pessoas': 3.5,
    'metrics_category_cliente': 3.5
}
```

Código 10: Pesos definidos para as features

A configuração de pesos futuramente poderia ser ajustada pelo usuário do sistema, permitindo uma personalização conforme as necessidades específicas de

cada caso de uso. Essa flexibilidade possibilita que os usuários modifiquem os pesos das features para obter resultados de similaridade e recomendação que melhor atendam aos seus objetivos, otimizando ainda mais a aplicabilidade do sistema de recomendação. Em seu estado atual, o sistema de recomendação ainda não permite que o usuário ajuste os pesos para cada feature, sendo uma melhoria possível de ser implementada em versões posteriores.

4.2.2.6. Persistência do Modelo

O modelo e seus componentes são salvos usando a biblioteca `joblib`. Um dicionário armazena as matrizes TF-IDF, os pesos das features, o `DataFrame` original, a lista de features, e os vetorizadores TF-IDF. O modelo é salvo em um arquivo `.joblib`, permitindo sua reutilização em futuras recomendações sem necessidade de reprocessamento.

4.3. Modelo de Recomendação de Classificação Multi-label

Diante da necessidade de diversificar as estratégias de recomendação, uma segunda abordagem, inicialmente planejada como um modelo baseado em conteúdo, foi considerada para sugerir métricas com base nas características do perfil de cada usuário. No entanto, apesar de o dataset de métricas conter descrições relativamente robustas, a implementação desse modelo apresentou dificuldades. A ausência de metadados adicionais, como a frequência de uso de cada métrica, o contexto específico de aplicação, ou até características mais detalhadas dos respondentes, impediu a criação de uma representação suficientemente rica para realizar uma recomendação eficaz. Para superar essas limitações, optou-se por desenvolver uma abordagem de Classificação *Multi-label*. Em vez de confiar exclusivamente nas descrições das métricas, essa nova abordagem utiliza características organizacionais e práticas ágeis dos respondentes para prever diretamente as categorias de métricas que podem ser relevantes para cada perfil. A classificação multi-label é uma técnica de aprendizado de máquina na qual cada instância pode ser associada a múltiplas etiquetas simultaneamente, diferentemente da classificação tradicional, onde cada instância é atribuída a uma única classe (ACTIVELOOP.AI, 2023).

A seguir, é apresentada uma adaptação das perguntas utilizadas como base para a classificação do usuário no Quadro 8.

Pergunta	Coluna no dataset

Função na organização	*role*
Anos de experiência	*years_exp*
Tamanho da organização	*org_size*
Métodos ágeis utilizados	*agile_methods*
Rituais ágeis utilizados	*use_metrics_**
Valor mínimo de acurácia da classificação	N/A

Quadro 8: perguntas utilizadas para definir o perfil do usuário do sistema de recomendação de classificação multi-label. Elaborado pelo autor.

O desenvolvimento deste modelo seguiu duas tentativas. A primeira adotou uma estratégia de classificação multi-label direta, enquanto a segunda incorporou técnicas de balanceamento de dados e ajustes de hiperparâmetros para otimizar o desempenho. A seguir, são descritas cada uma dessas abordagens, com a justificativa para a escolha da segunda tentativa como implementação final.

4.3.1. Primeira Tentativa de Implementação do Modelo de Classificação Multi-label

Na primeira tentativa de desenvolvimento do modelo de recomendação, o objetivo foi explorar uma abordagem de classificação multi-label que pudesse identificar as categorias de métricas relevantes para cada usuário do sistema. Essa abordagem permitiu prever múltiplas categorias de métricas para um único perfil, o que reflete melhor a realidade de que um usuário pode estar associado a diversas práticas e métricas organizacionais ao mesmo tempo. Para implementar essa estratégia, foram utilizadas técnicas de pré-processamento de dados e um classificador multi-label com *Random Forest*, que tentava prever a presença de várias categorias simultaneamente.

O Random Forest é um algoritmo de aprendizado de máquina amplamente utilizado para tarefas de classificação e regressão, baseado na construção de múltiplas árvores de decisão que trabalham de forma conjunta (RIGATTI, 2017). Ele é capaz de modelar relações complexas e identificar interações entre os preditores, garantindo alta eficácia preditiva e ampla aplicabilidade em diferentes contextos (RIGATTI, 2017).

4.3.1.1. Pré-processamento

O processo de preparação dos dados começou com a transformação das categorias de métricas em rótulos binários. As features foram selecionadas a partir de atributos que caracterizam o perfil de cada usuário, incluindo fatores como função, experiência, e práticas de uso de métricas na organização. Foram filtrados os dados para incluir apenas as entradas que possuíam valores na coluna `metrics_category`, garantindo que cada instância no dataset tivesse ao menos uma categoria de métrica associada para permitir a classificação. Esse conjunto de dados filtrado foi então separado em duas partes principais: `X`, que representa as features, e `y`, que representa o target.

```
X = df[
    [
        'role',
        'years_exp',
        'org_size',
        'use_metrics_planning',
        'use_metrics_review',
        'use_metrics_weekly',
        'use_metrics_daily',
        'use_metrics_retro',
        'agile_methods']
    ].astype(str)


y = df['metrics_category'].str.get_dummies(sep=', ')
```

Código 11: Definição de features (X) e target (Y) adaptado

A coluna `metrics_category` foi convertida para um formato multi-label, onde cada entrada pode conter múltiplas categorias, de modo que uma codificação binária *(one-hot encoding)* foi aplicada para criar colunas separadas que indicavam a presença ou ausência de cada categoria para cada perfil. As características dos perfis foram combinadas em uma única representação textual. Para isso, informações como função, anos de experiência, tamanho da organização, uso de práticas ágeis e rituais em que métricas são aplicadas foram agregadas em uma única string. Essa representação compacta criou um bloco de texto consolidado para cada perfil, facilitando a análise das características organizacionais. Esse texto consolidado foi então transformado em vetores numéricos usando o TF-IDF, técnica que calculou a importância relativa dos termos no contexto de cada perfil. Esse processo ajudou o modelo a identificar padrões relevantes nas características textuais de cada usuário, preparando os dados para o treinamento.

```
X_combined = X.apply(lambda row: ' '.join(row.values), axis=1)
tfidf = TfidfVectorizer()

tfidf_matrix = tfidf.fit_transform(X_combined)
```

Código 12: Combinação de features e transformação em vetor numérico.

4.3.1.2. Treinamento

O conjunto de dados foi dividido em treinamento (80%), validação (10%) e teste (10%). O conjunto de treino foi utilizado para ajustar os parâmetros do classificador e ensinar o modelo a reconhecer padrões entre o perfil dos usuários e as categorias de métricas associadas. Durante essa fase, o modelo teve acesso apenas aos dados de treino, permitindo o ajuste de seus pesos internos para maximizar a previsão das categorias corretas. O conjunto de validação foi aplicado para observar o desempenho do modelo em dados novos, mas ainda durante o processo de treinamento. Com os dados de validação, foi possível identificar ajustes necessários no classificador, visando melhorias na precisão e no equilíbrio das previsões. Na fase de teste foi reservado para a etapa de avaliação final do modelo. Esse conjunto permaneceu separado durante todo o processo de treinamento e validação, sendo utilizado apenas após o ajuste completo do modelo. A aplicação do modelo nos dados de teste forneceu uma medida objetiva de sua capacidade de generalização, indicando seu desempenho em dados que ele ainda não havia visto.

```
X_train, X_temp, y_train, y_temp = train_test_split(tfidf_matrix, y, test_size=0.2,
random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)
```

Código 13: divisão do dataset entre treino, validação e teste.

Para conduzir o treinamento do modelo, foi utilizado *Random Forest Classifier* em conjunto com o `MultiOutputClassifier`. O `MultiOutputClassifier` é uma técnica que possibilita a aplicação de *Random Forest* em tarefas multi-label.

```
classifier = MultiOutputClassifier(RandomForestClassifier(n_estimators=100,
random_state=42))
classifier.fit(X_train, y_train)
y_val_pred = classifier.predict(X_val)
```

Código 14: treinamento do modelo

4.3.1.3. Avaliação

A avaliação do primeiro modelo foi realizada utilizando as métricas de precisão (*precision*), recall e f1-score (FILHO, 2023). Essa abordagem é amplamente utilizada na análise de modelos de classificação multi label, especialmente em cenários com desbalanceamento entre classes, devido à sua capacidade de fornecer uma visão mais equilibrada e detalhada do desempenho do modelo em relação a cada classe, em contraste com a acurácia, que oferece uma perspectiva geral. Precisão mede a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas pelo modelo. Essa métrica é particularmente relevante quando é necessário minimizar falsos positivos, ou seja, reduzir o número de casos em que o modelo prevê incorretamente uma categoria de métrica para um perfil onde ela não está presente (FILHO, 2023). Recall mede a proporção de instâncias verdadeiramente positivas que o modelo conseguiu identificar, capturando a sensibilidade do modelo em relação a cada classe. Essa métrica é útil para avaliar a capacidade do modelo de recuperar todas as ocorrências de uma determinada categoria, especialmente quando é crucial minimizar falsos negativos, ou seja, reduzir os casos em que o modelo deixa de identificar categorias presentes em um perfil (FILHO, 2023). F1-score é a média harmônica entre precisão e recall, sendo uma métrica eficaz em situações de desbalanceamento. Ele equilibra as duas métricas para fornecer uma única medida de desempenho, útil especialmente quando ambas são importantes para a tarefa (FILHO, 2023).

```
# Fazer previsões no conjunto de validação
y_val_pred = classifier.predict(X_val)
report_val = classification_report(y_val, y_val_pred, target_names=y.columns)
print(report_val)

# Fazer previsões no conjunto de teste
y_test_pred = classifier.predict(X_test)
report_test = classification_report(y_test, y_test_pred, target_names=y.columns)
print(report_test)
```

Código 15: resultados do treinamento

Os resultados obtidos para o conjunto de teste podem ser observados no Quadro 9.

Categoria	Precisão (Precision)	Recall	F1-Score
Cliente	56%	31%	40%
Cronograma e Progresso	80%	90%	85%
Pessoas	89%	40%	55%
Processo	40%	33%	36%

Produto	70%	97%	81%
Tecnologia	41%	50%	45%

Quadro 9: resultados obtidos com a implementação do modelo 1.

Os resultados do primeiro modelo indicaram um desempenho altamente variável entre as categorias. Enquanto "Cronograma e Progresso" e "Produto" apresentaram f1-scores satisfatórios de 0.85 e 0.81, respectivamente, outras categorias, como "Cliente" (f1-score 0.40), "Tecnologia" (0.45) e "Processo" (0.36), demonstraram desempenho insatisfatório. A análise detalhada sugere que a baixa precisão em algumas categorias levou a um número elevado de falsos positivos, enquanto o baixo recall em outras mostrou dificuldades em identificar corretamente as instâncias de certas classes. Esses problemas foram agravados pelo desbalanceamento entre as categorias e pela simplicidade do modelo, que não conseguiu capturar de maneira eficiente as nuances do conjunto de dados.

Diante dessas limitações, optou-se por desenvolver um modelo mais robusto na sequência.

4.3.2. Implementação do Modelo de Classificação Multi-label Final

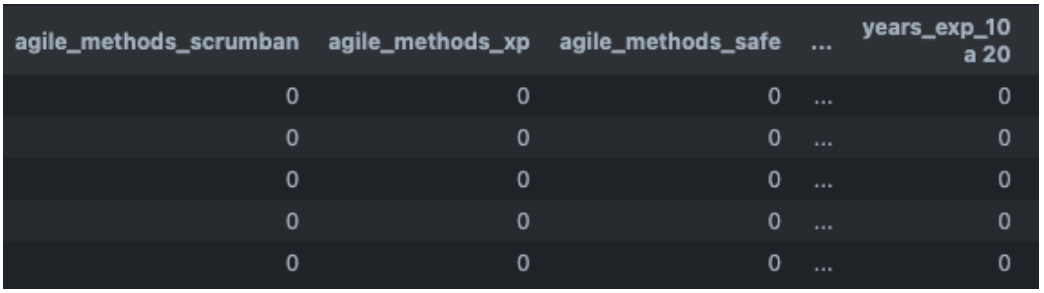
A segunda abordagem foi desenvolvida para superar as limitações observadas no primeiro modelo multi-label, focando principalmente em três aspectos fundamentais: refinamento do pré-processamento dos dados, aprimoramento da configuração do modelo de classificação e otimização das técnicas de treinamento e avaliação.

4.3.2.1. Pré-processamento

Foi identificado que um dos principais problemas para o desenvolvimento de um modelo de classificação mais robusto era o pré-processamento dos dados. O novo pré-processamento dos dados agora feito utilizando o mesmo pré-processamento do modelo de recomendação de filtragem colaborativa, principalmente para se utilizar do formato das colunas e estrutura dataset resultante deste tratamento. A base de dados utilizada para o modelo de filtragem colaborativa foi carregada, e as colunas binárias textuais foram convertidas em colunas binárias numéricas. As colunas categóricas (`role`, `years_exp` e `org_size`) foram convertidas para formato numérico utilizando one-hot-encoding.

```
[...]
código omitido
[...]

df[binary_columns] = df[binary_columns].replace({'Sim': 1, 'Não': 0})

categorical_columns = ['role', 'years_exp', 'org_size']
for col in categorical_columns:
    df[col] = df[col].astype(str)  # Garantir que são strings
df_encoded = pd.get_dummies(df, columns=categorical_columns,
prefix=categorical_columns)
```

Código 16: pré-processamento corrigido



agile_methods_scrumban	agile_methods_xp	agile_methods_safe	...	years_exp_10 a 20
0	0	0	...	0
0	0	0	...	0
0	0	0	...	0
0	0	0	...	0
0	0	0	...	0

Figura 25: dataset resultante

As categorias de métricas (o alvo da classificação) foram definidas como variáveis dependentes, isoladas das demais features do conjunto de dados. As colunas do dataset que representam o uso de cada categoria foram definidas como `metrics_category_cronograma_e_progresso`, `metrics_category_produto`, `metrics_category_processo`, `metrics_category_tecnologia`, `metrics_category_pessoas`, `metrics_category_cliente`. Cada coluna está preenchida no dataset com 1, caso a categoria estivesse presente na resposta do survey, e 0 caso ausente.

4.3.2.2. Treinamento

O processo de treinamento foi iterativo, tratando cada categoria de métrica como um problema binário distinto. No primeiro modelo, a abordagem adotada foi o uso do *Random Forest Classifie*r em conjunto com o `MultiOutputClassifier` para abordar todas as categorias de métricas simultaneamente. Este método tentava ajustar um único modelo para prever múltiplas categorias de métricas com base nos padrões encontrados no perfil dos usuários, tratando o problema como uma tarefa multi-label coletiva. O treinamento era realizado em um conjunto unificado de dados, e o modelo era avaliado em sua capacidade de generalizar a partir de um conjunto de validação e, posteriormente, de um conjunto de teste separado. Em contraste, o

segundo modelo introduziu uma mudança significativa na estratégia de modelagem e treinamento. Em vez de tratar todas as categorias de métricas como uma única tarefa multi-label, o segundo modelo adota uma abordagem individualizada, tratando cada categoria de métricas como um problema binário separado. Esse método envolve o treinamento de modelos distintos para cada categoria, o que permite um foco intensificado nas características únicas e necessidades de cada categoria de métricas. Tal abordagem facilita o ajuste mais fino dos parâmetros do modelo e a calibração da sensibilidade e especificidade para cada categoria, minimizando os efeitos do desbalanceamento das classes e melhorando a precisão das previsões.

```
[... loop sobre cada categoria ...]
model = RandomForestClassifier(random_state=42, n_estimators=200, max_depth=10,
class_weight='balanced')
model.fit(X_train, y_train)
```

Código 17: treinamento do segundo modelo.

No Código 16 é possível ver uma abordagem mais direcionada , utilizando o `RandomForestClassifier` sem o wrapper `MultiOutputClassifier`, como o primeiro. Este modelo é configurado com 200 árvores de decisão (`n_estimators=200`), o dobro do primeiro modelo, proporcionando uma maior robustez e capacidade de aprendizado, essencial para captar nuances mais sutis em cada categoria de métrica tratada individualmente. O parâmetro `max_depth` define uma profundidade máxima de 10 (`max_depth=10`), o que ajuda a prevenir o *overfitting*, permitindo que o modelo seja complexo o suficiente para aprender padrões detalhados, mas não tanto que memorize os dados de treino. A opção `class_weight='balanced'` ajusta os pesos das classes no processo de treinamento, compensando o desbalanceamento entre as categorias de métricas, garantindo que categorias menos representadas recebam atenção adequada durante o treinamento.

Para cada categoria, o conjunto de dados continuou sendo dividido novamente entre treino, validação e teste na mesma proporção (80%, 10% e 10%), utilizando o método `train_test_split` da biblioteca `sklearn`.

4.3.2.3. Avaliação

A avaliação do segundo modelo foi realizada de forma semelhante à do primeiro, utilizando métricas como precisão (precision), recall e f1-score para analisar

o desempenho de cada classificador binário treinado para as categorias de métricas. Os resultados obtidos estão exibidos no quadro abaixo.

```
# Fazer previsões no conjunto de validação
probabilities_val = model.predict_proba(X_val)[:, 1]
y_pred_val = model.predict(X_val)

# Fazer previsões no conjunto de teste
probabilities = model.predict_proba(X_test)[:, 1]
y_pred = model.predict(X_test)

print(classification_report(y_test, y_pred))
```

Código 18: treinamento do segundo modelo.

Categoria	Precisão (Precision)	Recall	F1-Score
Cronograma e Progresso	95%	100%	97%
Produto	94%	97%	96%
Processo	81%	83%	82%
Tecnologia	83%	88%	86%
Pessoas	91%	91%	90%
Cliente	86%	88%	87%

Quadro 10: resultados obtidos com a implementação do modelo 2.

Ao comparar os resultados do primeiro modelo de classificação multi-label com os do segundo modelo, observa-se uma melhoria significativa em todas as métricas de avaliação. No primeiro modelo, observou-se um desempenho altamente variável entre as categorias, com algumas, como "Cronograma e Progresso" e "Produto", apresentando resultados razoáveis, enquanto outras, como "Cliente", "Tecnologia" e "Processo", mostraram f1-scores marcadamente baixos. Este padrão indicava uma deficiência do modelo em lidar com o desbalanceamento das categorias e em capturar adequadamente as nuances específicas de cada grupo de métricas.

Na categoria "Cronograma e Progresso", o f1-score melhorou de 0.85 no primeiro modelo para 0.97 no segundo, refletindo um reconhecimento quase perfeito das métricas relevantes. Da mesma forma, na categoria "Produto", o f1-score aumentou de 0.81 para 0.96, evidenciando uma capacidade aprimorada do modelo em identificar com precisão as métricas aplicáveis. Significativas melhorias foram notadas especialmente nas categorias que anteriormente apresentavam os piores

desempenhos. Na categoria "Processo", o f1-score saltou de 0.36 para 0.82, e na "Tecnologia", de 0.45 para 0.86. A categoria "Cliente" também viu uma melhora, com o f1-score subindo de 0.40 para 0.87. Este resultado sugere que o segundo modelo foi eficiente em ajustar os pesos e em calibrar a sensibilidade do algoritmo para lidar com as peculiaridades e o desbalanceamento inerentes a essa categoria. Os resultados podem ser visualizados em formato gráfico por categoria na Figura 26.

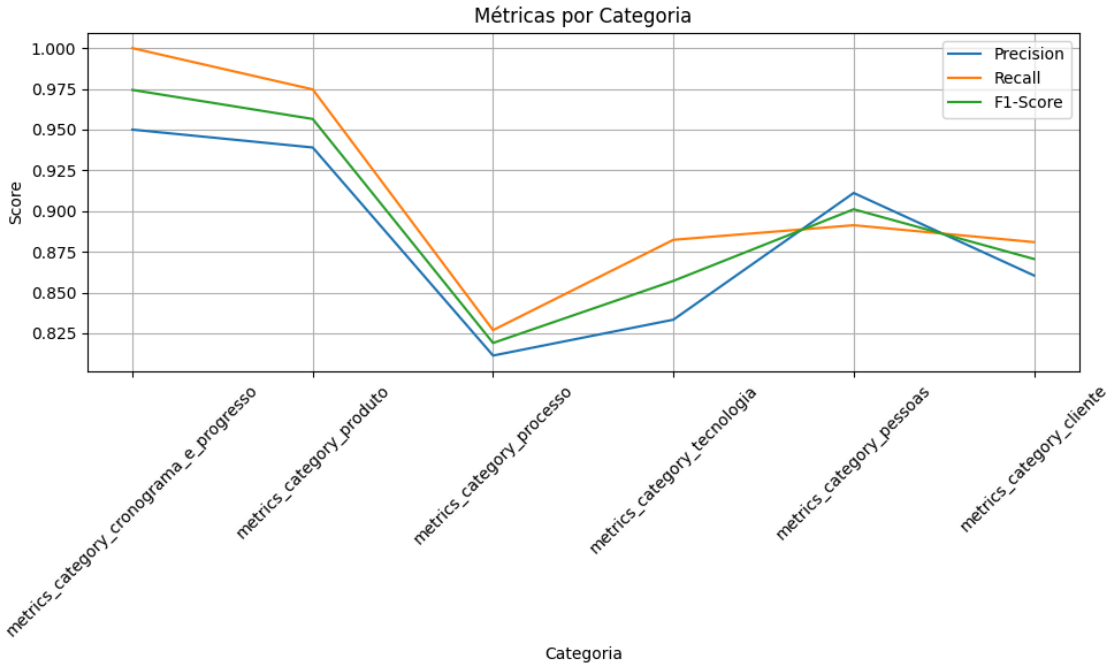


Figura 26: resultados.

4.3.2.4. Recomendação de Métricas

O sistema de recomendação opera classificando o perfil do usuário com base em características fornecidas, tais como função na organização (`role`), anos de experiência (`years_exp`), tamanho da organização (`org_size`), métodos ágeis utilizados (`agile_methods`) e o uso de métricas específicas em diferentes rituais ágeis (`use_metrics_*`). É utilizado o método `predict_proba`, que retorna a probabilidade direta do perfil pertencer ou não a classe (categoria de métricas), considerando somente a probabilidade positiva (de pertencer a classe), baseado no limiar definido como valor mínimo de similaridade. Com base na classificação resultante, o modelo recomenda métricas que correspondem às categorias às quais o usuário foi associado. Cada métrica recomendada é extraída do conjunto previamente definido por LEAL (2023) e está categorizada em uma das seguintes áreas: Cronograma e progresso (`metrics_category_cronograma_e_progresso`), Produto (`metrics_category_produto`), Processo (`metrics_category_processo`), Tecnologia

(metrics_category_tecnologia), Pessoas (metrics_category_pessoas) ou Ciente (metrics_category_cliente). O usuário recebe uma lista de todas as métricas pertinentes às categorias classificadas pelo modelo, acompanhadas de suas respectivas descrições. O sistema permite que o usuário defina um valor mínimo de similaridade para cada categoria, otimizando assim a relevância das métricas recomendadas de acordo com suas necessidades e preferências específicas.

```
metrics_data = pd.read_excel('extracted_metrics.xlsx', sheet_name="Extracted
metrics")

category_to_metrics = metrics_data.groupby('Translation')['Metric
Name'].apply(list).to_dict()

categories = [metrics_category_cronograma_e_progresso, metrics_category_produto,
metrics_category_processo, metrics_category_tecnologia, metrics_category_pessoas,
metrics_category_cliente]
for category in categories:
   model_filename = f"modelo_{category}.joblib"
   rf_binary = joblib.load(model_filename)
   user_affinities = rf_binary.predict_proba(user_df)[:, 1]
            for user_idx, affinity in enumerate(user_affinities):
                category_probabilities[user_idx][category] = affinity

threshold = 0.5
for category_affinities in category_probabilities.items():
            user_recommendations = []
            for categoria, afinidade in category_affinities.items():
                if categoria in category_to_metrics and afinidade >= threshold:
                    for metrica in category_to_metrics[categoria]:
                        if pd.notna(metrica):
                            user_recommendations.append({'metric': metrica,
'affinity': afinidade, 'category': categoria})
```

Código 19: recomendação de métricas (adaptado).

4.3.2.5. Persistência do Modelo

O modelo foi persistido de forma semelhante ao primeiro modelo, utilizando joblib para armazenar os binários dos modelos treinados para cada métrica, os resultados de classificação do modelo e a estrutura de dados do dataset tratado para uso posterior na etapa de recomendação.

4.4. Sistema Web para uso dos Modelos de Recomendação

O sistema web para recomendação de métricas foi projetado com o objetivo fornecer uma interface para o consumo de dois modelos distintos de recomendação.

A escolha das tecnologias utilizadas foi orientada pela necessidade de uma arquitetura moderna e de simples implementação. O frontend foi desenvolvido como uma *Single Page Application* (SPA)[9] utilizando a biblioteca em Javascript *React*[10]. O estilo e a experiência do usuário foram desenvolvidos com o uso da biblioteca *@mui/material*[11]. A navegação entre as diferentes funcionalidades da aplicação foi implementada com *react-router-dom*[12]. O backend foi implementado em *Flask*[13], um framework para o desenvolvimento de servidores web em *Python*, escolhido por sua compatibilidade com bibliotecas de aprendizado de máquina como *scikit-learn*[14]. A integração com os modelos foi feita utilizando *joblib*[15], permitindo o carregamento e a execução de modelos pré-treinados.

Para garantir que o Sistema Web forneça os resultados desejados, foram definidos uma série de requisitos funcionais e não funcionais, abordados nas seções seguintes.

4.4.1. Requisitos Funcionais

Os requisitos funcionais foram inicialmente definidos com base na fundamentação teórica e no estado da arte. Esses requisitos foram então refinados por meio de entrevistas com o orientador deste trabalho, os requisitos funcionais para o desenvolvimento do Sistema Web foram definidos como (Quadro 11):

ID	Tipo	Descrição
RF01	Funcional	O sistema deve permitir que o usuário selecione entre dois modelos de recomendação: Classificação Multilabel e Filtro Colaborativo.
RF02	Funcional	O sistema deve exibir informações contextuais, como título, descrição do trabalho e informações acadêmicas na página inicial.
RF03	Funcional	Na tela de classificação multilabel, o sistema deve permitir que o usuário preencha as informações de função, anos de experiência e tamanho da empresa.
RF04	Funcional	O sistema deve permitir a seleção de métodos ágeis utilizados e rituais ágeis.
RF05	Funcional	Na tela de classificação multilabel, o sistema deve incluir um controle deslizante (slider) para que o usuário ajuste o limite de similaridade (threshold).

[9] https://www.bairesdev.com/blog/react-spa-single-page-application/
[10] https://react.dev/
[11] https://www.npmjs.com/package/@mui/material
[12] https://github.com/remix-run/react-router
[13] https://flask.palletsprojects.com/en/stable/
[14] https://scikit-learn.org/stable/
[15] https://joblib.readthedocs.io/en/stable/

RF06	Funcional	O sistema deve exibir um botão para enviar os dados preenchidos e obter as recomendações baseadas no modelo selecionado.
RF07	Funcional	Na tela de filtro colaborativo, o sistema deve incluir todos os campos a de informações do perfil e um campo adicional para selecionar categorias onde aplicar métricas.
RF08	Funcional	Na tela de filtro colaborativo, o sistema deve permitir que o usuário defina o número máximo de métricas recomendadas.
RF09	Funcional	O sistema deve apresentar as métricas recomendadas em uma tabela com as colunas: Métrica, Afinidade, Categoria e Descrição.
RF10	Funcional	O sistema deve retornar à página inicial ao clicar no botão "Voltar para a Página Inicial".
RF11	Funcional	O backend deve processar os dados fornecidos pelo usuário e retornar as métricas recomendadas baseadas no modelo selecionado.
RF12	Funcional	O sistema deve fornecer uma API para recomendações baseadas no modelo de classificação multilabel.
RF13	Funcional	O sistema deve fornecer uma API para recomendações baseadas no modelo de filtro colaborativo.
RF14	Funcional	O backend deve validar os dados enviados pelo frontend e retornar mensagens de erro claras em caso de inconsistências.
RF15	Funcional	O backend deve agrupar métricas por categoria e utilizá-las para calcular as recomendações.

Quadro 11: requisitos funcionais

Estes requisitos foram definidos com o objetivo de garantir que o Sistema de Recomendação funcione corretamente e forneça os resultados mínimos desejáveis para ser utilizado, obtendo recomendações de métricas, abordando desde as funcionalidades mais básicas como navegação entre os sistemas de recomendação, a seleção das features específicas para cada tipo de modelo, e a implementação da API para a interação da interface Web com os modelos desenvolvidos.

4.4.2. Requisitos Não Funcionais

Os requisitos não funcionais para o desenvolvimento do Sistema Web foram definidos como:

ID	Tipo	Descrição
RNF01	Não Funcional	O sistema deve fornecer uma interface intuitiva e autoexplicativa, utilizando componentes modernos do @mui/material.
RNF02	Não Funcional	O sistema deve oferecer feedback visual durante o processamento, utilizando um indicador de carregamento (e.g., CircularProgress).
RNF03	Não Funcional	A interface deve ser responsiva e adaptada para diferentes dispositivos, como desktops, tablets e smartphones.

RNF04	Não Funcional	O tempo de resposta do backend deve ser inferior a 10 segundos para gerar recomendações.
RNF05	Não Funcional	O sistema deve validar as entradas do usuário no frontend e backend para evitar erros e inconsistências.
RNF06	Não Funcional	A arquitetura deve permitir a adição de novos modelos de recomendação com impacto mínimo no sistema existente.
RNF07	Não Funcional	A interface e os dados retornados devem estar em português
RNF08	Não Funcional	O backend deve registrar logs detalhados para monitoramento e depuração de erros.

Quadro 12: requisitos não funcionais

4.4.3. Interfaces desenvolvidas

As figuras 27, 28 e 29 demonstram as interfaces implementadas para o sistema web.

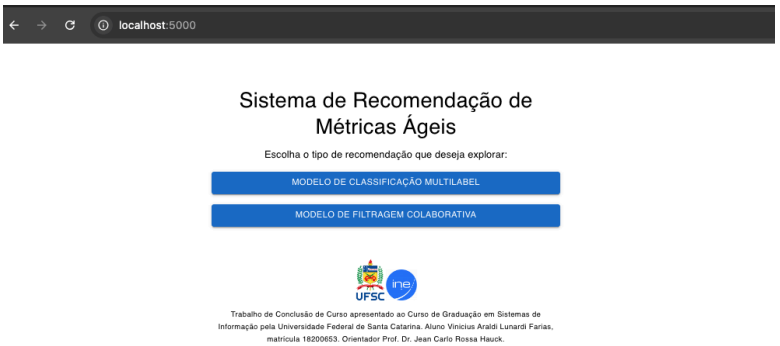


Figura 27: página inicial de seleção de modelo.

Figura 28: página de uso do modelo de classificação multilabel.

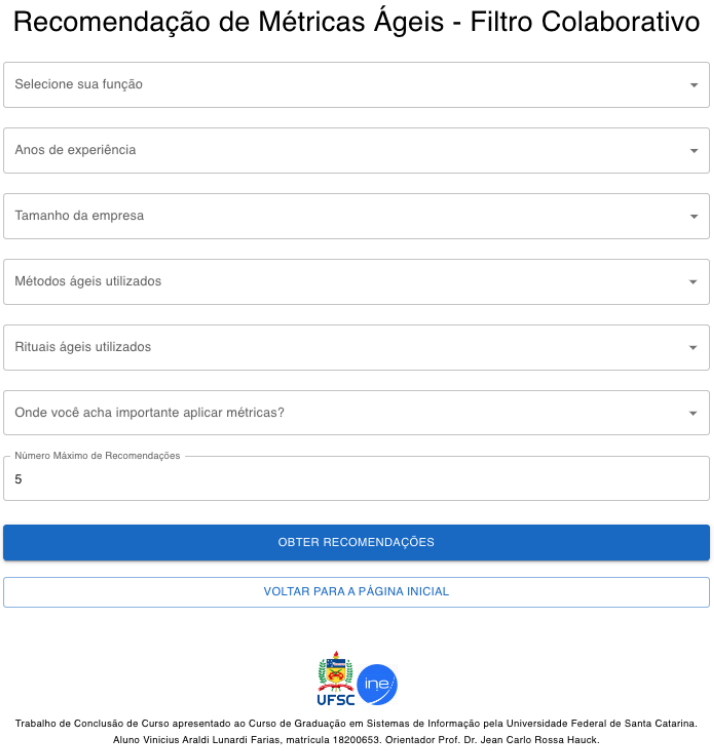


Figura 29: página de uso do modelo de filtro colaborativo.

4.5. Testes

O processo de testes desenvolvido para avaliar os modelos de recomendação implementados no sistema web teve como objetivo principal avaliar sua aplicabilidade em diferentes cenários de uso. Para isso, foram definidos casos de testes com perfis variados, representando diferentes combinações de características organizacionais e práticas ágeis, com base nos dados de entrada esperados pelo sistema, que contemplassem os requisitos funcionais e não funcionais definidos. Os testes foram projetados para cobrir uma ampla gama de possibilidades, desde perfis bem definidos e especializados até perfis intermediários e genéricos, permitindo uma análise abrangente do desempenho de cada modelo em situações reais. A análise comparativa considerou métricas de afinidade e categorias atribuídas pelo modelo de classificação multi-label, bem como as recomendações específicas e direcionadas geradas pelo modelo de filtro colaborativo.

4.5.1. Casos de teste

A seguir são apresentados casos de teste definidos com base em perfis de usuários que refletem diferentes combinações de papéis, níveis de experiência, tamanhos de empresas e práticas ágeis. Cada teste utiliza tanto o modelo de classificação multilabel quanto o modelo de filtro colaborativo, validando diferentes cenários do sistema. Para cada caso de teste, são descritos os dados de entrada e os requisitos funcionais e não funcionais contemplados. Também são identificados os passos para efetuar o caso de teste, e os resultados esperados para cada modelo.

1. **T01 - Gerente de Projeto com Grande Experiência em Empresa Grande**

Cenário de teste: este teste avalia o comportamento do sistema para um profissional com alta experiência e papel gerencial em uma grande organização.

ID	Dados de Entrada	Requisitos Funcionais	Requisitos Não Funcionais
T01	role: "Project Manager", years_exp: "Mais de 20", org_size: "Grande empresa (>500 empregados)", agile_methods: ["Scrum", "Safe", "Kanban"], rituals_to_use: ["Reunião de Planejamento", "Sprint Review", "Reunião Diária (daily)"], categories (apenas para filtro colaborativo): "Gestão de Tempo e Progresso", "Eficiência dos Processos”.	Todos	RNF01, RNF02, RNF04, RNF04, RNF07

Quadro 13: Dados de entrada e cobertura de requisitos do teste T01

2. **T02 - Scrum Master com Experiência Moderada em Empresa Média**

Cenário de teste: este teste avalia o comportamento do sistema para um profissional de nível médio, atuando em uma organização de porte intermediário.

ID do Caso de Teste	Dados de Entrada	Requisitos Funcionais	Requisitos Não Funcionais
T02	role: "Scrum master", years_exp: "6 a 9", org_size: "Média empresa (100-500 empregados)", agile_methods: ["Scrum", "XP"], rituals_to_use: ["Sprint Review", "Reunião Diária (daily)"], categories (apenas para filtro colaborativo): "Gestão de Equipes", "Desempenho do Produto"	Todos	RNF01, RNF02, RNF04, RNF04, RNF07

Quadro 14: Dados de entrada e cobertura de requisitos do teste T02

3. T03 - Desenvolvedor Júnior em Microempresa

Cenário de teste: este teste avalia como o sistema se comporta para um profissional iniciante, em uma empresa de pequeno porte, com práticas ágeis limitadas.

ID do Caso de Teste	Dados de Entrada	Requisitos Funcionais	Requisitos Não Funcionais
T03	role: "Desenvolvedor(a)", years_exp: "0 a 5", org_size: "Microempresa (< 10 empregados)", agile_methods: ["Lean"], rituals_to_use: ["Retrospectiva"], categories (apenas para filtro colaborativo): "Soluções Tecnológicas".	Todos	RNF01, RNF02, RNF04, RNF04, RNF07

Quadro 15: Dados de entrada e cobertura de requisitos do teste T03

4. T04 - Agile Coach com Alta Experiência

Cenário de teste: este teste avalia o sistema para um profissional de alta senioridade, focado em práticas ágeis diversificadas em uma grande empresa.

ID do Caso de Teste	Dados de Entrada	Requisitos Funcionais	Requisitos Não Funcionais

T04	role: "Agile Coach", years_exp: "10 a 20", org_size: "Grande empresa (>500 empregados)", agile_methods: ["Scrum", "Kanban", “ScrumBan”], rituals_to_use: ["Reunião de Planejamento", "Retrospectiva”, Reunião Semanal"], categories (apenas para filtro colaborativo): "Gestão de Equipes", "Gestão de Tempo e Progresso".	Todos	RNF01, RNF02, RNF04, RNF04, RNF07

Quadro 16: Dados de entrada e cobertura de requisitos do teste T04

4.5.2. Aplicação e Resultados

Os passos para reprodução dos cenários de teste e os resultados esperados para cada modelo são apresentados a seguir. (Quadro 17):

ID	Passos para reproduzir	Resultados Esperados Multi-Label	Resultados Esperados Filtro Colaborativo
T01	Acessar tela de seleção de modelo de recomendação	Obter recomendações de métricas relacionadas a produtividade e gestão estratégica. Ser categorizado com as categorias: "Cronograma e Progresso", "Processo".	Recomendações de métricas relacionadas à produtividade e gestão estratégica, como "Cycle Time" e "Throughput".
	Acessar a tela de recomendação utilizando Modelo Multilabel		
	Preencher os dados do perfil		
	Configurar o limite de similaridade para categorização		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		
	Voltar para a tela de seleção de modelo de recomendação		
	Acessar a tela de recomendação utilizando Modelo de Filtro Colaborativo		
	Preencher os dados do perfil		
	Configurar o limite de métricas para serem recomendadas		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		
T02	Acessar tela de seleção de modelo de recomendação	Obter recomendação de métricas relacionadas a	Recomendações de métricas relacionadas à eficiência
	Acessar a tela de recomendação utilizando Modelo Multilabel		

	Preencher os dados do perfil	qualidade do produto e gestão de tecnologia. Ser categorizado com as categorias: "Produto" e "Tecnologia".	operacional e inovação. Métricas como "User Story Points" e "Velocity".
	Configurar o limite de similaridade para categorização		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		
	Voltar para a tela de seleção de modelo de recomendação		
	Acessar a tela de recomendação utilizando Modelo de Filtro Colaborativo		
	Preencher os dados do perfil		
	Configurar o limite de métricas para serem recomendadas		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		
T03	Acessar tela de seleção de modelo de recomendação	Obter recomendações de métricas relacionadas a soluções tecnológicas. Ser categorizado com a categoria "Tecnologia".	Recomendações de métricas relacionadas à tecnologia, como "Test Coverage" e "Number of commits".
	Acessar a tela de recomendação utilizando Modelo Multilabel		
	Preencher os dados do perfil		
	Configurar o limite de similaridade para categorização		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		
	Voltar para a tela de seleção de modelo de recomendação		
	Acessar a tela de recomendação utilizando Modelo de Filtro Colaborativo		
	Preencher os dados do perfil		
	Configurar o limite de métricas para serem recomendadas		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		
T04	Acessar tela de seleção de modelo de recomendação	Obter recomendações de métricas relacionadas a eficiência dos processos, planejamento estratégico e	Obter recomendações mais específicas, de métricas relacionadas à gestão de escopo, entregas no prazo e qualidade do
	Acessar a tela de recomendação utilizando Modelo Multilabel		
	Preencher os dados do perfil		
	Configurar o limite de similaridade para categorização		

	Clicar no botão "Obter recomendações"	cronograma. Classificação nas categorias "Processos" e "Cronograma e Progresso".	produto, como "Burndown", "Bugs density" e "Number of completed tasks".
	Receber recomendações de métricas		
	Voltar para a tela de seleção de modelo de recomendação		
	Acessar a tela de recomendação utilizando Modelo de Filtro Colaborativo		
	Preencher os dados do perfil		
	Configurar o limite de métricas para serem recomendadas		
	Clicar no botão "Obter recomendações"		
	Receber recomendações de métricas		

Quadro 17: Passos para reproduzir e resultados esperados.

Todos os casos de testes foram aplicados sobre os dois modelos. A seguir, são discutidos os resultados obtidos e demonstrados os retornos de forma adaptada.

4.5.2.1. Resultados Para o Caso de Teste T01

Os resultados para o Caso de Teste T01 (Gerente de Projeto com Grande Experiência em Empresa Grande) mostram que ambos os modelos oferecem recomendações alinhadas às necessidades típicas de um gerente de projeto com grande experiência em uma empresa de grande porte. O modelo de filtro colaborativo identificou métricas com alta afinidade, destacando aquelas relacionadas à produtividade, eficiência de custos e desempenho do time, o que é consistente com a necessidade de otimização de processos em empresas maiores. O modelo de classificação multi-label categorizou o perfil com maior probabilidade nas categorias de "Produto" e "Cronograma e Progresso", 57.97% e 66.17% de probabilidade de pertencer a esta categoria, respectivamente, O segundo modelo recomenda métricas amplas e diversas dessas categorias.

OBTER RECOMENDAÇÕES

Métrica	Afinidade	Categoria	Descrição
Number of completed tasks	66.17 %	Cronograma e progresso	O número de histórias planejadas que foram concluídas em uma sprint dividido pelo número total de histórias planejadas.
Scope growth	66.17 %	Cronograma e progresso	"PBIs Adicionados" é a porcentagem de PBIs que são novos no backlog de uma determinada iteração de desenvolvimento, ou seja, que não existiam no backlog da iteração anterior.
Priority Shift	66.17 %	Cronograma e progresso	A métrica "Mudança de Prioridade" considera a alteração relativa na prioridade e a adiciona ao resultado geral. Uma mudança da prioridade 1 ("essencial") para 4 ("desejável") significaria numericamente uma mudança de 3.
Rejected Product Backlog Items	66.17 %	Cronograma e progresso	A métrica de PBIs Rejeitados mostra, para uma determinada iteração, a porcentagem do Tamanho do Projeto que foi rejeitada desde a iteração anterior.
Indicator about risky user stories	66.17 %	Cronograma e progresso	"Governança do projeto: Gestão de riscos: ciclo de histórias de usuário: indicador sobre histórias de usuário arriscadas"

Figura 30: Resultados modelo multi-label

OBTER RECOMENDAÇÕES

Métrica	Afinidade	Descrição
Velocity	99.50 %	A Velocidade mede o trabalho concluído em uma Sprint. Em algumas equipes, a velocidade é contada como o número de Itens do Backlog do Produto concluídos em uma Sprint. Em outras, é a soma dos pontos de estimativa que a equipe atribuiu aos Itens do Backlog do Produto concluídos em uma Sprint. Pode ser medida com Pontos de História ou por métodos de medição de tamanho funcional mais sistemáticos e objetivos, como Pontos de Função Cósmicos.
NPS	99.50 %	Densidade de feedback do usuário final relacionado a um problema em um período determinado. A modificação no projeto é conforme os novos requisitos recebidos do cliente no local ou para corrigir bugs/erros nas versões anteriores dentro das sprints. Ambas as métricas de satisfação do cliente e do desenvolvedor são definidas por questionários. A satisfação do cliente/desenvolvedor é avaliada entre 0-100%. Mede a satisfação do cliente/desenvolvedor com o projeto como um todo e/ou durante o projeto com cada release, iteração, etc., dependendo da empresa.
Indicator about risky user stories	99.50 %	"Governança do projeto: Gestão de riscos: ciclo de histórias de usuário: indicador sobre histórias de usuário arriscadas"
Throughput	99.50 %	Densidade de problemas resolvidos em um determinado período de tempo. Densidade de problemas resolvidos com tempo de vida abaixo do limite definido. Densidade de tarefas cuja resolução não excedeu o limite de duração definido. Uma Análise de Throughput revela a taxa de fluxo de trabalho através do sistema ao longo do tempo. Combinada com a Análise de Demanda, mostra quanto do trabalho é Demanda de Valor versus Demanda de Falha. Unidades de trabalho que a equipe concluiu dentro de um período de tempo definido.
Project avg. cost	96.75 %	Custo originado do DEP: (DEP × VAHC) / 100 Custo originado do custo de horas virtuais: (VAHC – VPHC) / 100
Lead Time	96.75 %	O Lead Time (às vezes chamado de Lead Time do Cliente) indica quanto tempo leva para itens de trabalho individuais passarem por todo o sistema, desde o momento em que algo se torna uma solicitação, ideia ou relatório de bug, até o momento em que o item concluído está nas mãos dos usuários. O Lead Time é o tempo total entre a concepção e a realização de qualquer incremento.
Cycle Time	96.75 %	O Tempo de Ciclo indica quanto tempo leva para itens de trabalho individuais passarem por um estado específico do fluxo de trabalho ou por um conjunto de estados do fluxo de trabalho. O Tempo de Ciclo é o período de tempo decorrido desde que a equipe começou a trabalhar na tarefa, ou seja, desde que a tarefa foi movida para o status "Em Progresso".

Figura 31: Resultados modelo de filtro colaborativo

4.5.2.2. Resultados Para o Caso de Teste T02

Os resultados para o Caso de Teste T02 (Scrum Master com Experiência Moderada em Empresa Média) mostram que ambos os modelos oferecem recomendações alinhadas às responsabilidades e objetivos típicos de um Scrum Master em uma organização de médio porte. O modelo de filtro colaborativo identificou métricas com afinidades consistentes, destacando indicadores como eficiência técnica e gerenciamento de tarefas, refletindo a importância de medir produtividade e acompanhar o progresso contínuo das sprints. O modelo de classificação multi-label categorizou o perfil com maior probabilidade nas categorias de "Cronograma e Progresso" (74.01%), seguido por "Processo" (56.51%), "Tecnologia" (53.98%), e "Cliente" (54.57%), sugerindo um foco equilibrado entre planejamento, execução de processos ágeis e interação com stakeholders.

Métrica	Afinidade	Categoria	Descrição
Number of completed tasks	74.01 %	Cronograma e progresso	O número de histórias planejadas que foram concluídas em uma sprint dividido pelo número total de histórias planejadas.
Scope growth	74.01 %	Cronograma e progresso	"PBIs Adicionados" é a porcentagem de PBIs que são novos no backlog de uma determinada iteração de desenvolvimento, ou seja, que não existiam no backlog da iteração anterior.
Priority Shift	74.01 %	Cronograma e progresso	A métrica "Mudança de Prioridade" considera a alteração relativa na prioridade e a adiciona ao resultado geral. Uma mudança da prioridade 1 ("essencial") para 4 ("desejável") significaria numericamente uma mudança de 3.
Rejected Product Backlog Items	74.01 %	Cronograma e progresso	A métrica de PBIs Rejeitados mostra, para uma determinada iteração, a porcentagem do Tamanho do Projeto que foi rejeitada desde a iteração anterior.
Indicator about risky user stories	74.01 %	Cronograma e progresso	"Governança do projeto: Gestão de riscos: ciclo de histórias de usuário: indicador sobre histórias de usuário arriscadas"
Reopened Tickets	74.01 %	Cronograma e progresso	Essa métrica mostra quantas tarefas foram reabertas durante o período do tempo de ciclo. As reaberturas podem vir de outros desenvolvedores durante as revisões de código ou da equipe de QA durante os testes. Os produtos e o ambiente da empresa são muito estáveis, mas o tamanho (em termos de esforço) dos itens de trabalho pode ter mudado ao longo do tempo. Não há controle de horas indicando quanto tempo cada desenvolvedor ou testador gastou em cada item de trabalho. Em vez disso, use o retrabalho no código (code churn) como uma medida substituta de esforço. Ao usar o churn nas análises, remova itens de trabalho que não envolvam alterações de código (ou seja, aqueles com churn = 0). Densidade de tickets que foram reabertos pelos testadores.

Figura 32: Resultados modelo multi-label

Métrica	Afinidade	Descrição
Technical Efficiency	65.50 %	Technical efficiency (output/input) AH/C
Number of remaining tasks	65.50 %	A métrica "Tamanho Restante do Projeto" está intimamente relacionada ao Tamanho do Projeto, pois utiliza o resultado dessa métrica juntamente com o progresso realizado. O objetivo dessa medida é mostrar o progresso da equipe durante a sprint atual. Ela exibe o número de pontos de história estimados e restantes para a sprint em andamento. Número de pontos de história estimados e restantes.
Lead Time	65.11 %	O Lead Time (às vezes chamado de Lead Time do Cliente) indica quanto tempo leva para itens de trabalho individuais passarem por todo o sistema, desde o momento em que algo se torna uma solicitação, ideia ou relatório de bug, até o momento em que o item concluído está nas mãos dos usuários. O Lead Time é o tempo total entre a concepção e a realização de qualquer incremento.
Throughput	65.11 %	Densidade de problemas resolvidos em um determinado período de tempo. Densidade de problemas resolvidos com tempo de vida abaixo do limite definido. Densidade de tarefas cuja resolução não excedeu o limite de duração definido. Uma Análise de Throughput revela a taxa de fluxo de trabalho através do sistema ao longo do tempo. Combinada com a Análise de Demanda, mostra quanto do trabalho é Demanda de Valor versus Demanda de Falha. Unidades de trabalho que a equipe concluiu dentro de um período de tempo definido.
Cycle Time	65.11 %	O Tempo de Ciclo indica quanto tempo leva para itens de trabalho individuais passarem por um estado específico do fluxo de trabalho ou por um conjunto de estados do fluxo de trabalho. O Tempo de Ciclo é o período de tempo decorrido desde que a equipe começou a trabalhar na tarefa, ou seja, desde que a tarefa foi movida para o status "Em Progresso".
Lead Time	58.50 %	O Lead Time (às vezes chamado de Lead Time do Cliente) indica quanto tempo leva para itens de trabalho individuais passarem por todo o sistema, desde o momento em que algo se torna uma solicitação, ideia ou relatório de bug, até o momento em que o item concluído está nas mãos dos usuários. O Lead Time é o tempo total entre a concepção e a realização de qualquer incremento.

Figura 33: Resultados modelo de filtro colaborativo

4.5.2.3. Resultados Para o Caso de Teste T03

Os resultados para o Caso de Teste T03 (Desenvolvedor Júnior em Microempresa) mostram que o modelo de filtro colaborativo encontrou dificuldades em fornecer recomendações com afinidade suficientemente alta, limitando-se a um nível médio de 31% para todas as métricas identificadas, como Lead Time, Delivery to Customer e Throughput. O modelo de classificação multi-label categorizou o perfil com maior probabilidade nas categorias de "Cronograma e Progresso" (76.21%), seguido por "Produto" (58.17%) e "Cliente" (51.82%). Essas categorias destacam a importância de métricas voltadas para o acompanhamento de prazos, qualidade das entregas e relacionamento com clientes

WIP (work in progress)	76.21 %	Cronograma e progresso	O WIP (Work in Progress) é definido como o número de itens de trabalho em andamento, independentemente dos estados intermediários em que os itens se encontram no quadro. Trabalho em andamento: Variabilidade no tempo.
Blocked tasks	76.21 %	Cronograma e progresso	Essa métrica mostra as tarefas com maior tempo gasto em status onde há trabalho ativo sendo realizado (como "Em Progresso") e maior tempo gasto em status onde não há trabalho ativo sendo realizado (como "Progresso Interrompido"). Dado o número de tarefas concluídas, essa métrica indica quantas delas foram interrompidas ou bloqueadas em algum momento durante o desenvolvimento e quantas não foram. Horas que não produzem um resultado tangível.
Merge Request Life Time	76.21 %	Cronograma e progresso	Mede o tempo médio de vida de uma merge request, desde a criação da solicitação até o momento em que ela foi mesclada.
Merge Request Review	76.21 %	Cronograma e progresso	Para avaliar se as equipes estão revisando o código de maneira criteriosa, calculamos a porcentagem de merge requests com pelo menos um comentário ou tarefa criada durante a semana atual.
Outstanding bugs	58.17 %	Produto	O objetivo dessa métrica é mostrar o número de defeitos abertos que a equipe deve corrigir antes do final da sprint. Número de defeitos abertos. Densidade de erros identificados em um determinado período de tempo. Densidade de bugs identificados após o lançamento. O número de defeitos gerados por uma equipe e por cada programador durante cada iteração e/ou lançamento, bem como ao longo de todo o projeto.
Critical bugs	58.17 %	Produto	Não informado.
Known bugs	58.17 %	Produto	Densidade de problemas de uma determinada prioridade com idade superior ao limite definido de criação. Densidade de bugs identificados antes da data de lançamento.
Project avg. time-to-market	58.17 %	Produto	O Time-to-Market e a Produtividade são calculados como uma média ponderada, com o Tamanho sendo o fator de ponderação. O Time-to-Market é medido em número de dias de calendário por PF (Ponto de Função).

Figura 34: Resultados modelo multi-label

Métrica	Afinidade	Descrição
Lead Time	31.00 %	O Lead Time (às vezes chamado de Lead Time do Cliente) indica quanto tempo leva para itens de trabalho individuais passarem por todo o sistema, desde o momento em que algo se torna uma solicitação, ideia ou relatório de bug, até o momento em que o item concluído está nas mãos dos usuários. O Lead Time é o tempo total entre a concepção e a realização de qualquer incremento.
Delivery to customer	31.00 %	Com este indicador, determinamos a cada semana se algum cliente fez o download de uma versão da aplicação da equipe pelo menos uma vez. Número de unidades da equipe/entregas a serem adquiridas do fornecedor.
Throughput	31.00 %	Densidade de problemas resolvidos em um determinado período de tempo. Densidade de problemas resolvidos com tempo de vida abaixo do limite definido. Densidade de tarefas cuja resolução não excedeu o limite de duração definido. Uma Análise de Throughput revela a taxa de fluxo de trabalho através do sistema ao longo do tempo. Combinada com a Análise de Demanda, mostra quanto do trabalho é Demanda de Valor versus Demanda de Falha. Unidades de trabalho que a equipe concluiu dentro de um período de tempo definido.
NPS	31.00 %	Densidade de feedback do usuário final relacionado a um problema em um período determinado. A modificação no projeto é conforme os novos requisitos recebidos do cliente no local ou para corrigir bugs/erros nas versões anteriores dentro das sprints. Ambas as métricas de satisfação do cliente e do desenvolvedor são definidas por questionários. A satisfação do cliente/desenvolvedor é avaliada entre 0-100%. Mede a satisfação do cliente/desenvolvedor com o projeto como um todo e/ou durante o projeto com cada release, iteração, etc., dependendo da empresa.

Figura 35: Resultados modelo de filtro colaborativo

4.5.2.4. Resultados Para o Caso de Teste T04

Os resultados para o Caso de Teste T04 (Agile Coach com Alta Experiência) mostram que o modelo de filtro colaborativo foi altamente eficaz em identificar métricas diretamente relevantes, atingindo uma afinidade máxima de 100% para diversas métricas importantes, como Cycle Time, Throughput, Lead Time, Technical Efficiency e Project Average Cost. O modelo de classificação multi-label, por sua vez, categorizou o perfil com alta probabilidade nas categorias de "Produto" (89.05%), seguida por "Tecnologia" (84.16%), "Processo" (83.74%), "Cliente" (81.00%) e "Cronograma e Progresso" (67.06%). Essa classificação demonstra uma abordagem robusta para identificar categorias de métricas que refletem o foco de um Agile Coach em entregar valor contínuo, otimizar processos, adotar soluções tecnológicas inovadoras e garantir a satisfação do cliente.

Test Coverage	84.16 %	Tecnologia	Cobertura de testes: número de testes em cada nível de teste e área do software. Número de requisitos ou número de alterações cobertos pela equipe.
Test success rate	84.16 %	Tecnologia	Não informado.
Running Tested Features Metric (RTF)	84.16 %	Tecnologia	É definido como o número de funcionalidades que passaram em seus testes de aceitação.
Security Test Pass Rate (security)	84.16 %	Tecnologia	Não informado.
Commit Review Performance	84.16 %	Tecnologia	Densidade de revisões de commits com duração abaixo do limite definido.
NPS	81.00 %	Cliente	Densidade de feedback do usuário final relacionado a um problema em um período determinado. A modificação no projeto é conforme os novos requisitos recebidos do cliente no local ou para corrigir bugs/erros nas versões anteriores dentro das sprints. Ambas as métricas de satisfação do cliente e do desenvolvedor são definidas por questionários. A satisfação do cliente/desenvolvedor é avaliada entre 0-100%. Mede a satisfação do cliente/desenvolvedor com o projeto como um todo e/ou durante o projeto com cada release, iteração, etc., dependendo da empresa.

Figura 36: Resultados modelo multi-label

Project avg. cost	100.00 %	Custo originado do DEP: (DEP * VAHC) / 100 Custo originado do custo de horas virtuais: (VAHC – VPHC) / 100
NPS	100.00 %	Densidade de feedback do usuário final relacionado a um problema em um período determinado. A modificação no projeto é conforme os novos requisitos recebidos do cliente no local ou para corrigir bugs/erros nas versões anteriores dentro das sprints. Ambas as métricas de satisfação do cliente e do desenvolvedor são definidas por questionários. A satisfação do cliente/desenvolvedor é avaliada entre 0-100%. Mede a satisfação do cliente/desenvolvedor com o projeto como um todo e/ou durante o projeto com cada release, iteração, etc., dependendo da empresa.
Cycle Time	100.00 %	O Tempo de Ciclo indica quanto tempo leva para itens de trabalho individuais passarem por um estado específico do fluxo de trabalho ou por um conjunto de estados do fluxo de trabalho. O Tempo de Ciclo é o período de tempo decorrido desde que a equipe começou a trabalhar na tarefa, ou seja, desde que a tarefa foi movida para o status "Em Progresso".
Test Coverage	100.00 %	Cobertura de testes: número de testes em cada nível de teste e área do software. Número de requisitos ou número de alterações cobertos pela equipe.
Lead Time	100.00 %	O Lead Time (às vezes chamado de Lead Time do Cliente) indica quanto tempo leva para itens de trabalho individuais passarem por todo o sistema, desde o momento em que algo se torna uma solicitação, ideia ou relatório de bug, até o momento em que o item concluído está nas mãos dos usuários. O Lead Time é o tempo total entre a concepção e a realização de qualquer incremento.
Ideal team capacity	100.00 %	A capacidade ideal da equipe em horas que o fornecedor oferece é calculada pela fórmula: C = S * (DS - (NW + (DS * KO / 20))) * TS * DH

Figura 37: Resultados modelo de filtro colaborativo

4.5.3. Análise Comparativa dos Modelos de Recomendação

A análise comparativa dos modelos de recomendação utilizados no sistema web revelou diferenças significativas em suas abordagens, resultados e aplicabilidades, destacando vantagens e limitações complementares. A diferença principal entre os dois modelos está na granularidade: o filtro colaborativo prioriza métricas específicas com altíssima afinidade em maior parte dos casos de teste, enquanto o multi-label classifica o perfil em categorias e expande as recomendações de forma mais abrangente. O modelo de classificação multi-label baseia-se em aprendizado supervisionado, onde o perfil do usuário é categorizado em uma ou mais classes de métricas, com uma afinidade percentual atribuída a cada categoria. A recomendação é derivada diretamente dessas categorias, abrangendo todas as métricas associadas. Esse modelo apresentou uma abordagem generalista, ideal para fornecer um conjunto abrangente de métricas em cenários onde o perfil se encaixa claramente em categorias predefinidas. O modelo de filtro colaborativo fundamenta-se na similaridade entre perfis e utiliza características textuais e numéricas ponderadas para calcular afinidades, resultando em recomendações mais específicas e adaptadas às necessidades de perfis semelhantes.

Os resultados obtidos nos casos de teste destacaram a consistência do modelo multi-label na classificação das categorias para todos os perfis analisados. Para perfis

bem definidos, como os casos de teste T03 e T04, o modelo apresentou classificações com alta afinidade em categorias relevantes, oferecendo um conjunto amplo de métricas que, embora abrangente, podem incluir métricas menos diretamente aplicáveis devido à abordagem genérica. O modelo de filtro colaborativo demonstrou excelente desempenho em perfis bem estruturados, identificando métricas altamente específicas com afinidades elevadas, como no caso de teste T04, onde afinidades de 100% resultaram em recomendações precisas e direcionadas. Contudo, em perfis intermediários ou menos definidos, como o caso de teste T03, o modelo colaborativo encontrou dificuldade em identificar métricas com afinidades significativas, resultando em recomendações menos impactantes. As principais vantagens do modelo multi-label incluem sua capacidade de fornecer um panorama geral de métricas relevantes para uma ampla gama de cenários e perfis iniciais. Entretanto, sua abordagem generalista pode comprometer a especificidade das recomendações, especialmente em casos onde métricas mais direcionadas são necessárias. Já o modelo de filtro colaborativo destaca-se pela afinidade alta em alguns casos de recomendação, de acordo com a similaridade do perfil definido do usuário com os perfis do dataset, fornecendo uma personalização maior em determinadas recomendações, sendo ideal para usuários avançados ou perfis com características bem definidas. No entanto, seu desempenho é limitado pela densidade e qualidade do dataset utilizado, sendo menos eficiente em casos de perfis genéricos ou com pouca representatividade no histórico de dados.

Os dois modelos demonstram complementaridade significativa, com o modelo multi-label fornecendo uma abordagem abrangente e introdutória, enquanto o filtro colaborativo oferece recomendações mais refinadas e específicas. Essa complementaridade sugere que uma abordagem híbrida, combinando ambos os modelos, poderia maximizar os benefícios de cada um, oferecendo ao usuário tanto uma visão geral quanto recomendações detalhadas e personalizadas, dependendo de suas necessidades e do contexto de aplicação.

Caso de Teste	Modelo Multi-Label	Modelo de Filtro Colaborativo
T01	Categorizado em "Produto" com probabilidade de 57.97% e "Cronograma e Progresso" com probabilidade de 66.17%. Recomendação abrangente de métricas relacionadas a estas categorias.	Recomendou métricas com alta afinidade, focadas em produtividade, eficiência de custos e desempenho do time.

T02	Categorizado em "Cronograma e Progresso" com probabilidade de 74.01%, "Processo" com probabilidade de 56.51%, "Tecnologia" com probabilidade de 53.98% e "Cliente" com probabilidade de 54.57%. Oferece uma visão equilibrada entre planejamento e execução.	Recomendou métricas com afinidades consistentes, focando em indicadores como eficiência técnica e gerenciamento de tarefas.
T03	Categorizado em "Cronograma e Progresso" com probabilidade de 76.21%, "Produto" com probabilidade de 58.17% e "Cliente" com probabilidade de 51.82%. Foco em prazos, qualidade das entregas e relacionamento com clientes.	Encontrou dificuldades, com afinidade média de 31% para métricas como Lead Time, Delivery to Customer e Throughput.
T04	Categorizado em "Produto" com probabilidade de 89.05%, "Tecnologia" com probabilidade de 84.16%, "Processo" com probabilidade de 83.74%, "Cliente" com probabilidade de 81.00% e "Cronograma e Progresso" com probabilidade de 67.06%. Ofereceu uma abordagem robusta.	Recomendou métricas com afinidade máxima de 100% (ex.: Cycle Time, Lead Time, Technical Efficiency). Foco em otimização de processos e valor contínuo.

Quadro 18: comparativo entre os modelos

5. CONCLUSÃO

Este trabalho propôs o desenvolvimento e avaliação de dois modelos de recomendação de métricas ágeis, com o objetivo de aprimorar a tomada de decisão de gerentes de projeto e equipes ágeis em diversos contextos organizacionais. Foram conduzidas análises da base teórica, abordando conceitos relacionados a metodologias ágeis, métricas ágeis e medição de software, bem como aprendizado de máquina e sistemas de recomendação. Este processo orientou o desenvolvimento dos dois modelos apresentados.

O modelo de classificação multi-label demonstrou sua eficácia ao fornecer uma visão abrangente das categorias de métricas aplicáveis, enquanto o modelo de filtro colaborativo destacou-se por suas recomendações específicas e altamente personalizadas, particularmente para perfis bem definidos. A análise comparativa dos resultados evidenciou a complementaridade entre os dois modelos. O modelo multi-label se mostrou mais adequado para cenários iniciais ou genéricos, nos quais é necessário um conjunto diversificado de métricas para apoiar diferentes aspectos do gerenciamento ágil. O modelo de filtro colaborativo apresentou maior afinidade entre os perfis, ao recomendar métricas específicas, sendo particularmente eficaz para usuários avançados ou com características bem definidas. Os resultados obtidos nos casos de teste destacaram tanto os pontos fortes quanto as limitações de cada abordagem. Enquanto o modelo multi-label oferece uma cobertura abrangente, ele carece de especificidade em alguns casos. O modelo de filtro colaborativo, apesar de sua afinidade elevada em perfis estruturados, teve desempenho limitado em contextos onde os dados eram escassos ou menos representativos.

Como contribuição científica, este trabalho apresenta duas abordagens diferentes para a recomendação de métricas ágeis, combinando aprendizado supervisionado e técnicas colaborativas. É possível obter contribuições significativas tanto para o campo acadêmico quanto para aplicações práticas em ambientes corporativos. A comparação entre dois modelos distintos permite compreender suas implicações e potenciais. No contexto corporativo, a solução facilita a busca por métricas ágeis, que podem ser aplicadas por profissionais em seus diferentes contextos, recebendo informações detalhadas de cada métrica. A integração desses modelos em um sistema web facilita o uso do modelo, ampliando o impacto na melhoria da gestão ágil em diferentes setores. O trabalho também envolveu a estruturação de um dataset de métricas mais refinado, desenvolvido a partir do

processamento dos dados coletados por meio do survey aplicado. A etapa de refinamento incluiu a normalização de métricas, reestruturação de colunas, categorização detalhada com descrições e características associadas traduzidas e remoção de potenciais outliers.

5.1. Trabalhos Futuros

Trabalhos futuros poderão explorar a construção de um sistema híbrido, combinando os dois modelos de recomendação para aumentar a capacidade de atender a diferentes perfis e se adaptar a mudanças nos contextos organizacionais. Essa abordagem pode permitir que o modelo de classificação multi-label e o modelo de filtro colaborativo atuem de forma complementar, aproveitando as vantagens de ambos. Para o modelo de filtro colaborativo, a criação de uma meta-heurística para realizar o ajuste de pesos de forma automática, buscando a otimização dos valores dos pesos de cada feature, pode ser abordada. Estes pesos também poderiam ser fornecidos pelo usuário, de acordo com suas necessidades, para que cada feature tenha um impacto melhor na recomendação.

O sistema web poderia fornecer um detalhamento mais claro de como funciona cada modelo de recomendação, para que um usuário mais leigo consiga tirar proveito de cada modelo da melhor forma, através de algum campo que contenha a explicação e principais características de cada modelo.

A ausência de informações mais completas sobre as métricas e a limitação no volume de dados podem ter afetado os resultados, especialmente no modelo de filtro colaborativo, que depende diretamente da qualidade e da amplitude dos dados disponíveis para identificar padrões e similaridades. Uma possível abordagem seria revisar as análises realizadas no dataset atual para identificar lacunas e direcionar um novo levantamento de dados. Esse novo survey poderia ser planejado com foco em capturar informações mais específicas e detalhadas sobre as métricas ágeis, incluindo contextos de aplicação, impacto nas equipes e sua relação com práticas ágeis. Os modelos desenvolvidos podem ser revisados, com ajustes nos parâmetros de treinamento e mudanças nas metodologias empregadas.

Uma reavaliação do processo de pré-processamento de dados também pode ser explorado para refinar ainda mais o desempenho das recomendações. A remoção de outliers poderia ser revista, uma vez que com um conjunto de dados pequeno,

como o utilizado para realização do trabalho, qualquer dado a menos pode ser significativo para resultados melhores. Para não remover outliers e aproveitar todos os dados, poderiam ser realizadas técnicas de oversampling no conjunto de dados, para construir mais informações de modo com que os outliers não tivessem impactos significativos. Com um conjunto de dados mais robusto e uma análise mais detalhada dos resultados obtidos, seria possível não apenas melhorar a precisão das recomendações, mas também ampliar o impacto prático dos modelos em ambientes organizacionais reais.

REFERÊNCIAS

ABBAR, S; et al. Context-aware recommender systems: A service-oriented approach. VLDB PersDB workshop. 2009. August 24-28, 2009, Lyon, France.

ABES (Associação Brasileira das Empresas de Software). Estudo do Mercado Brasileiro de Software. 2022. Disponível em: https://abes.com.br/dados-do-setor/. Acesso em: junho de 2024

ACTIVELOOP.AI. Multilabel Classification. Disponível em: https://www.activeloop.ai/resources/glossary/multilabel-classification/. Acesso em: novembro de 2024.

ADOMAVICIUS, G; TUZHILIN A. Context-Aware Recommender Systems. 2010.

BECK, Kent et al. Manifesto for agile software development, 2001.

BOBADILLA, J. Recommender systems survey, 2013.

BROWNLEE, Jason. SMOTE for Imbalanced Classification with Python. Disponível em: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. Acesso em: novembro de 2024.

BRUNIALTI, Lucas et al. Aprendizado de Máquina em Sistemas de Recomendação Baseados em Conteúdo Textual: Uma Revisão Sistemática, 2015.

BURKE, R. Hybrid Recommender Systems: Survey and Experiments Department of Information Systems and Decision Sciences, California State University, Fullerton, EUA, 2002.

CARDOZO, Luiz Fernando. Um guia para aplicação de métricas ágeis de gerenciamento de projetos para organizações de desenvolvimento de software, 2020.

CARRER-NETO, W. et al. Social knowledge-based recommender system. Application to the movies domain 2012.

CHARMAZ, Kathy. Constructing grounded theory. sage, 2014.

CHEVITARESE, Daniel Salles. AnnCom – Biblioteca de Redes Neurais Artificiais com Treinamento Acelerado por Placas Gráficas, 2010.

CHORAŚ, Michał et al. Measuring and Improving Agile Processes in a Small-Size Software Development Company, 2020.

DAS, D. et al., 2017. A Survey on Recommendation System. International Journal of Computer Applications, 160(7), pp.6-10.

FILHO, Mario. O que é precisão, recall e f1-score em machine learning?. Disponível em: https://mariofilho.com/precisao-recall-e-f1-score-em-machine-learning/. Acesso em: novembro de 2024.

GAVRILOVA, Yulia. Artificial Intelligence vs. Machine Learning vs. Deep Learning: Essentials. Disponível em: https://ai.plainenglish.io/data-science-vs-artificial-intelligencevs-machine-learning-vs-deep-learning-50d3718d51e5. Acesso em: maio 2024.

GHOSH, D.; VOGT, A. Outliers: An evaluation of methodologies. In: JOINT STATISTICAL MEETINGS (JSM), 2012, Boston. Proceedings of the Section on Survey Research Methods.

HARTMANN, Deborah; DYMOND, Robin. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: AGILE 2006 (AGILE'06). IEEE, 2006. p. 6 pp.-134.

HAYKIN, Simon. Redes Neurais, 2001.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. *IEEE Standard for Software Test Documentation (IEEE 829)*. New York: IEEE, 2008. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4578383. Acesso em: novembro de 2024.

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

LANGLEY, P. Selection of Relevant Features in Machine Learning. *AAAI Technical Report FS-94-02*. 1994.

LEAL, Stephanie. Desenvolvimento de um modelo de seleção e avaliação de métricas de software para contextos ágeis, 2023.

MAS, Antònia; MESQUIDA, Antoni-Lluís; PACHECO, Marcos. Supporting the deployment of ISO-based project management processes with agile metrics, 2020.

MENDES, Priscilla; MACHIAVELLI, Josiane; GUSMAO, Cristine. Avaliação de Tecnologias de Recomendação de Conteúdo baseadas em Inteligência Artificial Numa Visão Educacional, 2019.

MITCHELL, Thomas. Machine Learning, 1997.

MORANDINO, Matheus; RODRIGUES, Michelly. Algoritmos de recomendação: mais presente no dia a dia do que você pensa, 2020. Disponível em: https://www.each.usp.br/petsi/jornal/?p=2684. Acesso em: maio 2024

NGUYEN, Loc; AMER, Ali A. Advanced Cosine Measures for Collaborative Filtering. *Journal of Survey Research Methods*, v. 15, n. 3, p. 45-60, 2019.
PIO, Bruno Luiz de Assis. Uma Introdução às Redes Neurais Artificiais para Estudos em Ecologia. Brasil, Creative Commons, 2009.

PRESSMAN, Roger S. et al. A practitioner’s approach. Software Engineering, v. 2, p. 41-42, 2010.

RAM, Prabhat; RODRÍGUEZ, Pilar; OIVO, Markku. Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study, 2018.

RED HAT BLOG. Feature Engineering: What Is It and Why Is It Important? Disponível em: https://www.redhat.com/pt-br/topics/ai/what-is-machine-learning. Acesso em: novembro de 2024.

RIGATTI, Steven J. Random Forest. Journal of Insurance Medicine, v. 47, p. 31–39, 2017.

ROCHA, Gislene. Inteligência Artificial e Decisões: Percepção dos Usuários sobre os Sistemas de Recomendação, 2022.

RUSSELL, STUART J. (STUART JONATHAN), 1962- Inteligência artificial / Stuart Russell, Peter Norvig; tradução Regina Célia Simille. – Rio de Janeiro: Elsevier, 2013

SAMUEL, Arthur L. Machine learning. The Technology Review, v. 62, n. 1, 1959.

SANTANA. Marlesson. Deep Learning para Sistemas de Recomendação (Parte 1) — Introdução. Disponível em: https://medium.com/data-hackers/deep-learning-para-sistemas-de-recomenda%C3%A7%C3%A3o-parte-1-introdu%C3%A7%C3%A3o-b19a896c471e. Acesso em: maio 2024.

SANTOS, Elisama et al. Rede Neural Perceptron Aplicada ao Jogo da Nave, 2019.

SHU, J., SHEN, X., LIU, H., YI, B. and ZHANG, Z., 2017. A content-based recommendation algorithm for learning resources. Multimedia Systems, 24(2), pp.163-173.

SOMMERVILLE, Ian; Engenharia de Software, 10. ed. 2019.

SOFTEX; MPS.BR - Melhoria de Processo do Software Brasileiro. Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS-SW:2016. p. 46, p. 48), 2016.

SOFTEX, MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software, 2013

SOUZA, Gabriel Manhães de. Análise de Tráfego de Botnets com Machine Learning, 2022.

TEIXEIRA, Júlio Monteiro. Gestão Visual de Projetos: Utilizando a informação para inovar. Alta Books, 2018.

VAN VEEN, Fjodor. The Neural Network Zoo, 2019. Disponível em: https://www.asimovinstitute.org/neural-network-zoo/ . Acesso em: maio 2024.

VERSIONONE, Collabnet. 16th annual state of agile survey report (2022).

WINDER Ph.D., Phil. Reinforcement Learning. Estados Unidos: O'Reilly Media, 2020.

APÊNDICE I – Código Fonte

O código fonte do sistema está disponível em https://codigos.ufsc.br/gqs/sistema-recomendacao-de-metricas, na *branch* “tcc-2”.

APÊNDICE II – Artigo

Um modelo de recomendação de métricas ágeis para organizações de software

Vinicius Araldi Lunardi Farias

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC – Brasil

vinicius.lunardi@grad.ufsc.br

Abstract. *Defining metrics to assist agile project managers is challenging due to the variety and decentralization of metrics used by different teams. This work focuses on developing a recommendation model for agile metrics, utilizing a dataset created from academic research, interviews, and surveys with experts. The solution involves a web system where users answer questions about their organizational context and receive personalized metrics through a Machine Learning model. Initial tests indicate that the system meets the proposed requirements, showing potential to optimize the use of metrics in agile management.*

Resumo. *A definição de métricas para auxiliar gerentes de projetos ágeis é desafiadora devido à variedade e descentralização das métricas usadas por diferentes times. Este trabalho aborda o desenvolvimento de um modelo de recomendação de métricas ágeis, utilizando um dataset criado a partir de pesquisas acadêmicas, entrevistas e surveys com especialistas. A solução envolve um sistema web onde o usuário responde perguntas sobre seu contexto organizacional e recebe métricas personalizadas por meio de um modelo de Machine Learning. Os testes iniciais indicam que o sistema atende aos requisitos propostos, mostrando potencial para otimizar o uso de métricas no gerenciamento ágil.*

1 INTRODUÇÃO

As métricas de software são “características de um sistema de software, documentação de sistema ou processo de desenvolvimento que pode ser objetivamente medido” (SOMMERVILLE, 2019), auxiliando na qualidade e confiabilidade do desenvolvimento. Dividem-se em medidas básicas e derivadas, que geram indicadores para a tomada de decisão (SOFTEX; MPS.B, 2016). Devido à descentralização, métricas são customizadas, nem sempre atendendo às necessidades dos times (SOMMERVILLE, 2019). A IA e o Machine Learning, amplamente aplicados em sistemas de recomendação, sugerem itens que atendem às necessidades dos usuários (Brunialti et al., 2015; Mendes; Machiavelli; Gusmao, 2019). Este trabalho propõe recomendar métricas ágeis validadas, facilitando a análise de projetos, utilizando dois modelos de recomendação diferentes. Os modelos poderão ser utilizados através de um sistema web, para obter as recomendações de métricas de acordo com as necessidades apresentadas pelos usuários.

2 MÉTRICAS ÁGEIS

As métricas ágeis são projetadas para alinhar-se aos princípios ágeis, sendo fáceis de coletar e oferecer indicadores claros para suportar decisões estratégicas (Hartmann e Dymond, 2006). Elas medem valor e promovem entendimento aprofundado no desenvolvimento de software, com papeis cruciais no monitoramento e controle de projetos (MAS, A. et al., 2020). Fundamentadas em medições empíricas, devem emitir alertas, influenciar comportamentos, educar e ser recicladas conforme necessário (Mas, Mesquida e Pacheco, 2020). Exemplos incluem gráficos de "Burndown" e taxas de aprovação em testes (CHORAŚ et al., 2020).

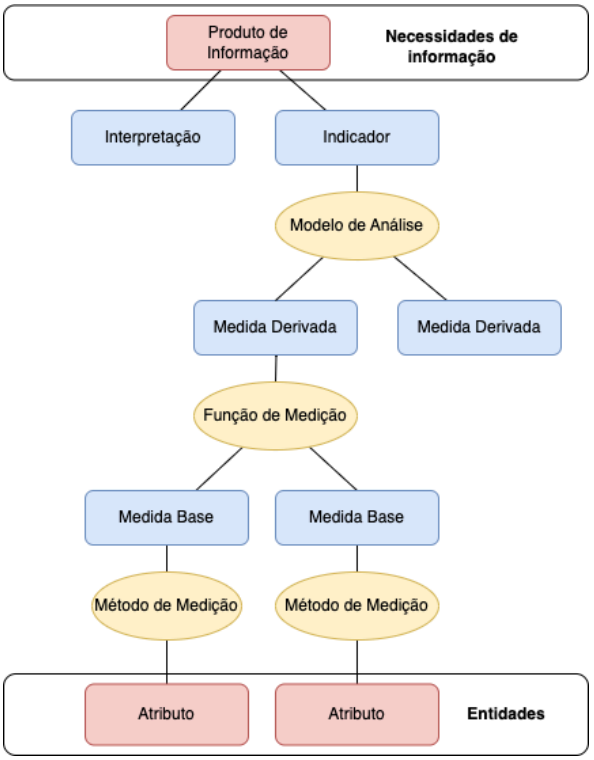


Figura 1: Formação das informações sobre o produto. Fonte: ISO/IEC/IEEE 15939:2017(E) (adaptado pelo Autor)

Segundo a ISO/IEC/IEEE 15939:2017(E), métricas são organizadas em medidas básicas, derivadas e indicadores, cada uma desempenhando funções específicas no processo de medição (Softex, 2016). A seleção de métricas deve ser baseada em relevância, simplicidade e capacidade de ação (CHORAŚ et al., 2020). Dados devem ser coletados de fontes confiáveis, automatizados quando possível, processados e visualizados de forma acessível em dashboards, garantindo iteração e ajustes contínuos para manter a relevância. Aplicadas ao desenvolvimento ágil, métricas suportam planejamento, monitoramento e controle, melhorando qualidade e eficiência (P. RAM et al., 2018).

3 SISTEMAS DE RECOMENDAÇÃO

Sistemas de recomendação utilizam inteligência artificial para filtrar e apresentar itens relevantes aos usuários, baseando-se em dados históricos e preferências previamente coletadas (SHAFFER, 2001; ABBAR et al., 2009). Filtragem colaborativa é uma técnica amplamente utilizada, que prevê itens de interesse com base em padrões de comportamento de usuários semelhantes (SANTANA, 2018).

Essa abordagem considera interações passadas, utilizando algoritmos como K-vizinhos mais próximos (KNN) e matrizes de interação usuário-item para calcular similaridades (Adomavicius e Tuzhilin, 2005; Bobadilla, 2013).

Já a filtragem baseada em conteúdo recomenda itens analisando as características dos produtos com os quais o usuário já interagiu, utilizando métodos como TF-IDF para avaliar similaridades em conteúdos textuais (PRIMO, 2013; ADOMAVICIUS e TUZHILIN, 2005). Esse tipo de sistema oferece recomendações personalizadas sem depender de dados de outros usuários, mas enfrenta desafios como o problema de cold start e superespecialização, que podem limitar a diversidade e a abrangência das recomendações (DAS et al., 2017; CARRER-NETO et al., 2012).

4 PREPARAÇÃO DOS DADOS

O dataset utilizado neste projeto foi desenvolvido em parceria com a mestranda Stéphanie da Silva Leal, por meio de um survey aprovado pelo Comitê de Ética em Pesquisa da UFSC, com o objetivo de investigar o uso e seleção de métricas ágeis em empresas de software no Brasil (LEAL, 2023). O survey seguiu os passos definidos por Kasunic (2005), como definição de objetivos, público-alvo, design do questionário, distribuição e análise de resultados. O questionário abordou aspectos como perfil dos respondentes, métodos ágeis utilizados, cerimônias aplicadas e principais métricas empregadas, totalizando 18 perguntas. A coleta de dados foi feita via Google Forms, com distribuição pelo LinkedIn para profissionais de gestão em métricas ágeis. Ao todo, 1043 profissionais foram convidados, resultando em 309 respostas e uma taxa de adesão de 29,6%. Os dados foram sanitizados e analisados utilizando a técnica de Open Coding, que organiza dados em categorias e identifica padrões para compreender melhor os resultados (CHARMAZ, 2014). A pesquisa forneceu uma base robusta para a análise do uso de métricas ágeis em diferentes contextos organizacionais no Brasil.

O método explode foi aplicado para separar métricas, criando múltiplas linhas por entrada. Dados nulos foram convertidos para strings, garantindo consistência. Outliers foram identificados em respostas do dataset que poderiam promover algum tipo de resultado indesejado nas recomendações. As figuras a seguir demonstraram que respostas com frequência menor que 2 foram removidas, garantindo maior confiabilidade.

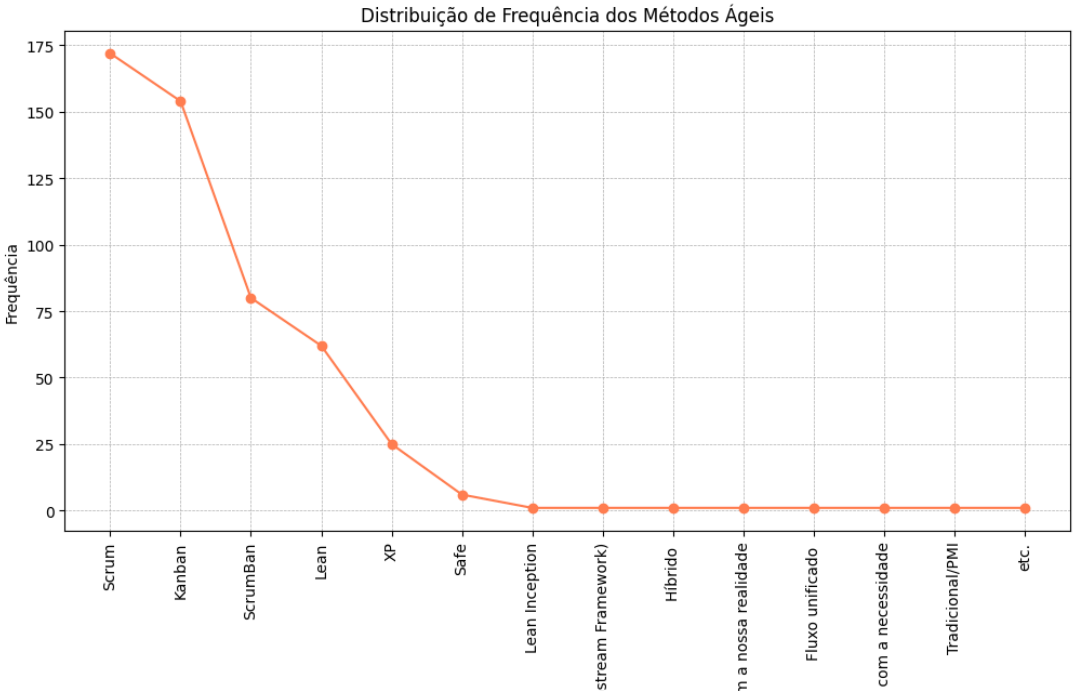


Figura 2: Gráfico da frequência de métodos ágeis. Elaborado pelo autor.

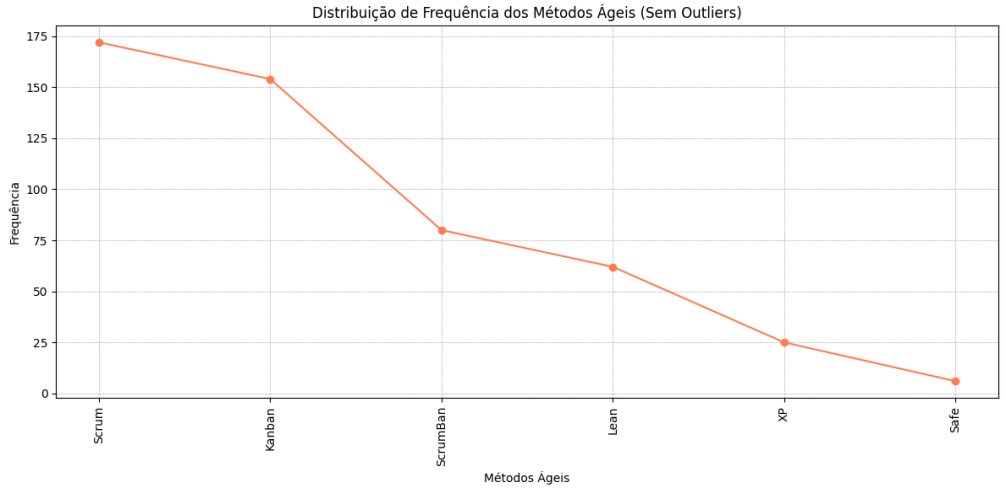


Figura 3: Gráfico da frequência de métodos ágeis sem outliers. Elaborado pelo autor.

5 DESENVOLVIMENTO DE MODELO DE RECOMENDAÇÃO DE FILTRAGEM COLABORATIVA

O modelo de recomendação de filtragem colaborativa desenvolvido neste trabalho utiliza similaridade de cossenos para calcular a proximidade entre perfis com base em features extraídas, como métodos ágeis, categorias de métricas e dados

organizacionais. A técnica TF-IDF foi empregada para vetorização de dados textuais, enquanto colunas binárias foram diretamente convertidas em matrizes numéricas. O modelo foi implementado com Scikit-learn, utilizando TF-IDF para transformar dados textuais e calcular similaridades. O sistema recomenda métricas utilizadas pelos perfis mais similares. O modelo foi salvo com joblib, armazenando TF-IDF, pesos, dataset e vetorizadores para reutilização sem necessidade de reprocessamento.

6 DESENVOLVIMENTO DE MODELO DE RECOMENDAÇÃO DE CLASSIFICAÇÃO MULTI-LABEL

A segunda abordagem desenvolvida foi a Classificação Multi-label. Essa técnica permite que cada instância seja associada a múltiplas categorias de métricas, como "Cronograma e Progresso", "Produto" e "Processo". A estratégia inicial, baseada em conteúdo, foi descartada devido à falta de metadados detalhados no dataset. A primeira tentativa de desenvolvimento do modelo utilizou o Random Forest com MultiOutputClassifier, que tenta prever todas as categorias simultaneamente. No entanto, os resultados foram inconsistentes, com baixo desempenho em categorias desbalanceadas.

Categoria	Precisão (Precision)	Recall	F1-Score
Cliente	56%	31%	40%
Cronograma e Progresso	80%	90%	85%
Pessoas	89%	40%	55%
Processo	40%	33%	36%
Produto	70%	97%	81%
Tecnologia	41%	50%	45%

Quadro1: resultados obtidos com a implementação do modelo 1.

A segunda abordagem tratou cada categoria como problema binário, refinando o pré-processamento para utilizar colunas estruturais do modelo de recomendação de filtro colaborativo, tratando todas as colunas como representações binárias utilizando One Hot Encoding. O Random Forest Classifier foi utilizado para criar um modelo individual de classificação para cada categoria. O dataset foi dividido em 80% treinamento, 10% validação e 10% teste. Isso resultou em (Quadro 2):

Categoria	Precisão (Precision)	Recall	F1-Score
Cronograma e Progresso	95%	100%	97%
Produto	94%	97%	96%
Processo	81%	83%	82%
Tecnologia	83%	88%	86%
Pessoas	91%	91%	90%
Cliente	86%	88%	87%

Quadro 2: resultados obtidos com a implementação do modelo 2.

Com base nas classificações, o sistema utiliza funções probabilísticas para calcular afinidades por categoria. Cada métrica é associada às categorias identificadas, e todas as métricas de cada categoria prevista são recomendadas. O usuário pode definir um limiar de similaridade para personalizar as recomendações. Os modelos foram salvos com joblib, permitindo sua reutilização em novas recomendações sem reprocessamento.

7 DESENVOLVIMENTO DE SISTEMA WEB

O sistema web foi desenvolvido como uma Single Page Application, utilizando React, @mui/material para estilização e Flask no backend, garantindo integração com modelos de aprendizado de máquina via joblib. Ele permite a seleção entre os modelos de recomendação de Classificação Multi-label e Filtro Colaborativo, fornecendo formulários para preenchimento de dados do perfil, ajuste de similaridade e exibição de métricas recomendadas em tabelas organizadas por afinidade, categoria e descrição. Os requisitos incluem interface responsiva, validação de dados e APIs para consumir cada modelo. O fluxo de interação abrange seleção do modelo, envio de informações pelo usuário, processamento no backend e apresentação dos resultados.

Figura 4: página inicial de seleção de modelo.

8 ANÁLISE COMPARATIVA DOS MODELOS

Os testes realizados avaliaram a aplicabilidade dos modelos de recomendação em diferentes cenários de uso, com perfis variados que combinam características organizacionais e práticas ágeis. Foram definidos casos de teste para cobrir uma ampla gama de situações, desde perfis bem definidos até intermediários e genéricos, alinhados aos requisitos funcionais e não funcionais do sistema.

A análise comparativa dos modelos de recomendação evidenciou diferenças fundamentais nas abordagens, resultados e contextos de uso, com cada modelo apresentando vantagens e limitações complementares. A principal distinção está na granularidade. O modelo de classificação multi-label, baseado em aprendizado supervisionado, categoriza o perfil do usuário em uma ou mais classes de métricas e atribui uma afinidade percentual a cada categoria. Ele oferece recomendações abrangentes, abrangendo todas as métricas associadas às categorias identificadas, sendo ideal para cenários em que os perfis se encaixam claramente em categorias predefinidas. Já o modelo de filtro colaborativo, fundamentado na similaridade entre perfis, utiliza características textuais e numéricas ponderadas para calcular afinidades, gerando recomendações mais específicas e personalizadas. As principais vantagens do modelo multi-label incluem sua abrangência e adequação a cenários diversos, especialmente para usuários iniciantes ou perfis pouco detalhados. Em contrapartida,

o modelo colaborativo se destaca pela alta personalização, fornecendo recomendações precisas em perfis avançados e bem definidos. Sua limitação, no entanto, reside na dependência da densidade e qualidade do dataset, tornando-o menos eficiente para perfis genéricos ou pouco representados.

9 CONCLUSÃO

Este trabalho desenvolveu dois modelos de recomendação de métricas ágeis para apoiar gerentes de projetos e equipes em contextos organizacionais variados. O modelo multi-label mostrou-se promissor em fornecer uma visão ampla de categorias de métricas, enquanto o modelo de filtro colaborativo se destacou por recomendações altamente personalizadas para perfis bem definidos. A análise comparativa evidenciou a complementaridade entre os modelos, sugerindo que o multi-label é ideal para cenários iniciais e genéricos, enquanto o filtro colaborativo é mais eficiente em perfis específicos.

Como contribuição científica, o trabalho combinou aprendizado supervisionado e técnicas colaborativas, estruturando um dataset refinado e categorizado. Trabalhos futuros podem explorar um sistema híbrido que combine ambos os modelos, além de aprimorar a coleta de dados e o pré-processamento para lidar melhor com outliers e lacunas no dataset. O desenvolvimento de uma interface mais clara para usuários leigos e a aplicação de técnicas como oversampling e ajuste automático de pesos podem potencializar a aplicabilidade dos modelos em contextos reais.

REFERÊNCIAS

ADOMAVICIUS, G; TUZHILIN A. Context-Aware Recommender Systems. 2010.

BOBADILLA, J. Recommender systems survey, 2013.

BRUNIALTI, Lucas et al. Aprendizado de Máquina em Sistemas de Recomendação Baseados em Conteúdo Textual: Uma Revisão Sistemática, 2015.

CARDOZO, Luiz Fernando. Um guia para aplicação de métricas ágeis de gerenciamento de projetos para organizações de desenvolvimento de software, 2020.

CARRER-NETO, W. et al. Social knowledge-based recommender system. Application to the movies domain 2012.

CHARMAZ, Kathy. Constructing grounded theory. sage, 2014.

CHORAŚ, Michał et al. Measuring and Improving Agile Processes in a Small-Size Software Development Company, 2020.

DAS, D. et al., 2017. A Survey on Recommendation System. International Journal of Computer Applications, 160(7), pp.6-10.

HARTMANN, Deborah; DYMOND, Robin. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: AGILE 2006 (AGILE'06). IEEE, 2006. p. 6 pp.-134.

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. *IEEE Standard for Software Test Documentation (IEEE 829)*. New York: IEEE, 2008. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4578383. Acesso em: novembro de 2024.

LEAL, Stephanie. Desenvolvimento de um modelo de seleção e avaliação de métricas de software para contextos ágeis, 2023.

MAS, Antònia; MESQUIDA, Antoni-Lluís; PACHECO, Marcos. Supporting the deployment of ISO-based project management processes with agile metrics, 2020.

MENDES, Priscilla; MACHIAVELLI, Josiane; GUSMAO, Cristine. Avaliação de Tecnologias de Recomendação de Conteúdo baseadas em Inteligência Artificial Numa Visão Educacional, 2019.

RAM, Prabhat; RODRÍGUEZ, Pilar; OIVO, Markku. Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study, 2018.

SANTANA. Marlesson. Deep Learning para Sistemas de Recomendação (Parte 1) — Introdução. Disponível em: https://medium.com/data-hackers/deep-learning-para-sistemas-de-recomenda%C3%A7%C3%A3o-parte-1-introdu%C3%A7%C3%A3o-b19a896c471e. Acesso em: maio 2024.

SOMMERVILLE, Ian; Engenharia de Software, 10. ed. 2019.

SOFTEX; MPS.BR - Melhoria de Processo do Software Brasileiro. Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS-SW:2016. p. 46, p. 48), 2016.

SOFTEX, MPS.BR - Melhoria de Processo do Software Brasileiro - Guia Geral MPS de Software, 2013