Universidade Federal do Amazonas
Instituto de Ciências Exatas e Tecnologia
Projeto para Educação e Pesquisa

RELATÓRIO TÉCNICO CIENTÍFICO 3:
Projeto da Plataforma Educacional: Code Debt

Itacoatiara - AM
2024

RELATÓRIO TÉCNICO CIENTÍFICO 3:
Projeto da Plataforma Educacional: Code Debt

Referência	
Título do Projeto	FindDT: Uma Ferramenta com Técnicas de Aprendizado de Máquina para Identificar a Dívida Técnica Auto-Admitida em Código Fonte
Objetivo	Apresentar o Projeto da Plataforma FindDT
Professores Coordenadores	Odette Mestrinho Passos Rainer Xavier de Amorim
Alunos Pesquisadores	Lucas de Souza Pinheiro Maryse da Silva Pires
Metodologia	Modelo Incremental
Cronograma	Setembro de 2023 a Fevereiro de 2024
Palavras-Chave	Plataforma Educacional, Aprendizado de Maquina e Dívida Técnica
Instituição Financeira	SUPER

Itacoatiara – AM
2024

RESUMO

Na engenharia de software, a aplicação de técnicas de Machine Learning para identificar dívidas técnicas representa uma abordagem inovadora, automatizando a detecção de problemas de qualidade no código-fonte. Isso contribui para o desenvolvimento de software mais robusto e sustentável, permitindo que os computadores aprendam e melhorem com a experiência, sem necessidade de programação explícita. Nesse sentido, o objetivo deste trabalho é apresentar a Plataforma Code Debt, onde são disponibilizados os resultados do duas pesquisas científicas e de uma ferramenta desenvolvida para identificar a Dívida Técnica. O modelo de desenvolvimento de software adotado foi o modelo incremental. Como resultado tivemos a documentação desenvolvida na Linguagem Unificada de Modelagem (UML), a implementação foi feita nas linguagens HML e JavaScript, estando a plataforma disponível no link: https://projetosufam.com.br/codedebt.

Palavras-Chave: Plataforma Educacional, Aprendizado de Máquina e Dívida Técnica

SUMÁRIO

1. INTRODUÇÃO

A Dívida Técnica (DT) no desenvolvimento de software é o resultado de decisões anteriores que se tornam inadequadas ao longo do processo de desenvolvimento devido às mudanças no contexto externo. Ela pode ser causada por eventos como surgimento de novas tecnologias, decisões arquiteturais, falhas tecnológicas e outras decisões que prejudicam a manutenção e o desenvolvimento do sistema (Kruchten, 2012).

A DT pode ser introduzida consciente ou inconscientemente. Os gerentes de projeto podem optar por não realizar uma atividade técnica essencial ou inserir correções temporárias intencionalmente em um projeto de software devido a restrições de custo ou pressão do mercado. Quando essas decisões são documentadas em comentários ao longo do código-fonte para reconhecer explicitamente a DT acumulada ocorre a Dívida Técnica Auto-Admitida (DTAA) (Lima, 2021).

A identificação automatizada da DTAA representa uma abordagem inovadora no desenvolvimento de software. Ao utilizar técnicas de Aprendizado de Máquina (ML), podemos automatizar a detecção dessas dívidas técnicas diretamente a partir do código-fonte. O ML é uma área da Inteligência Artificial (IA) que usa algoritmos para coletar e analisar dados, aprender com eles e fazer previsões ou determinação, surgindo da necessidade de lidar com grandes quantidades de dados e extrair dados relevantes a partir deles (Mitchell, 1997). Isso contribui para o desenvolvimento de software mais robusto e sustentável, permitindo que os computadores aprendam e melhorem com a experiência, sem necessidade de programação explícita.

Nesse contexto, o objetivo deste relatório é descrever uma Plataforma Web chamada Code Debt, que disponibiliza dois Relatórios Técnicos Científicos resultantes de um Mapeamento Sistemático e uma Revisão Rápida, sobre IA e DT, respectivamente, além de apresentar a ferramenta FindDT, juntamente com um guia completo para acessar e utilizar a ferramenta.

Essa plataforma é produto do projeto de pesquisa intitulado "FindDT: Uma Ferramenta com Técnicas de Aprendizado de Máquina para Identificar a Dívida Técnica Auto-Admitida em Código-Fonte", conduzido pelos alunos que participam do projeto SUPER, no âmbito do Instituto de Ciências Exatas e Tecnologia, que visa estimular a capacitação e a pesquisa em cursos de graduação da Universidade Federal do Amazonas (UFAM).

2. VISÃO GERAL DO DOCUMENTO

Esta visão geral fornece as informações necessárias para fazer um bom uso deste documento, explicitando os objetivos e as convenções que foram adotadas. Nas subseções a seguir serão apresentadas as especificações da plataforma.

2.1. Convenções, Termos e Abreviações

A correta interpretação deste documento exige o conhecimento de algumas convenções etermos específicos, que são descritos a seguir:

ID: Identificador

RF: Requisito(s) Funcional(ais)

RNF: Requisito(s) Não Funcional(ais)

RN: Regra(s) de Negócio

UC: Use Case (Casos de Usos)

ML: Machine Learning (Aprendizado de Máquina)

DT: Dívida Técnica

DTAA: Dívida Técnica Auto-Admitida

2.2. Prioridades dos Requisitos

Para estabelecer a prioridade dos requisitos foram adotadas as denominações “essencial”,“importante” e “desejável”:

- **Essencial:** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis e que têm que ser implementados impreterivelmente.
- **Importante:** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável:** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempohábil para implementá-los na versão que está sendo especificada.

2.3. Descrição Geral da Plataforma

A Plataforma Code Debt apresentará os resultados do Projeto de Pesquisa "Uma Ferramenta com Técnicas de Aprendizado de Máquina para Identificar a Dívida Técnica Auto-Admitida em Código-Fonte", desenvolvido pelos alunos Lucas Pinheiro e Maryse Pires, sob coordenação da Professora Odette Passos. O projeto foi realizado no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER).

De maneira mais específica ela é considerada um website, que consiste em um conjunto de páginas acessíveis por meio de um navegador que são hospedadas em um servidor web. Dentro das páginas, podem existir diversos conteúdos em formatos distintos, como: textos, imagens, vídeos e som. Os websites, ou simplesmente, sites, são utilizados para diferentes finalidades, incluindo: compartilhamento de informações, comércio eletrônico, entretenimento e redes sociais.

No caso da Plataforma Code Debt, será compartilhado informações sobre a ferramenta FindDT, além de disponibilizar dois Relatórios Técnicos Científicos, frutos de um Mapeamento Sistemático (MS) e de uma Revisão Rápida (RR). Não obstante, é apresentado um guia de como acessar e utilizar a ferramenta, bem como um passo a passo de como fazer download do banco de dados e dos modelos pré-treinados utilizados no desenvolvimento.

A plataforma contém uma página inicial (Home) onde primeiramente visualizamos um resumo do projeto, seus objetivos e introduzindo também sobre o Projeto Super. Em seguida, temos acesso as informações do menu: (1) Mapeamento Sistemático, (2) Revisão Rápida, (3) Ferramenta FindDT e (4) Sobre.

Ressalta-se que a metodologia adotada para desenvolver a plataforma foi a Modelo Incremental, que consiste em dividir o projeto em partes menores, ou incrementos , cada qual com uma funcionalidade completa e útil. Cada incremento é projetado, desenvolvido, testado e implementado de forma independente, e depois é agregado à plataforma final. No caso do Code Debt pode-se considerar que cada página do projeto é um incremento, juntamente com o desenvolvimento da ferramenta.

2.4. Descrição dos Usuários

Os usuários da plataforma estão descritos na Tabela 1:

Tabela 1 - Usuários da Plataforma

Usuários	Descrição
Público em Geral	Todos que possuem interesse em conhecer o projeto FindDT, ter acesso e utilizar a Ferramenta FindDT

Fonte: Autores (2024)

3. REQUISITOS GERAIS

3.1. Requisitos Funcionais

Os requisitos funcionais estão relacionados às funcionalidades que o software deve ter para atender as necessidades da empresa e/ou dos usuários, ou seja, do público-alvo. Eles ditam como a plataforma deve se comportar diante de determinadas situações que possam vir a ocorrer e podem dizer o que não deverá fazer. Com base no contexto da plataforma, foram identificados os requisitos funcionais mostrados na Tabela 2.

Tabela 2 - Requisitos Funcionais

ID	Descrição	Prioridade	Requisitos Relacionados
[RF001]	A plataforma deve permitir o download de arquivos	Essencial	[RN004]
[RF002]	A plataforma deve permitir a visualização do conteúdo	Essencial	[RN004]
[RF003]	A plataforma deve permitir o redirecionamento para ao “Kaggle”	Essencial	[RN003]

Fonte: Autores (2024)

3.2. Requisitos Não-Funcionais

Os requisitos não-funcionais estão relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenibilidade e tecnologias envolvidas. Eles descrevem as características mínimas de um software de qualidade, ficando a cargo do desenvolvedor optar por atender esses requisitos ou não. Foram identificados os requisitos não funcionais, conforme evidenciados na Tabela 3.

Tabela 3 - Requisitos Não-Funcionais

ID	Descrição	Categoria	Prioridade	Requisitos Relacionados
[RNF001]	Menu deverá ser fixo na parte superior da tela	Usabilidade	Desejável	[RF002]
[RNF002]	A plataforma deverá seguir a seguinte paleta de cores: ##2B4B80, #FF5C97, #FFFFFF e #F9F9F9	Usabilidade	Importante	[RF002]
[RNF003]	A plataforma deverá possuir as seguintes fontes de texto: Poppins e Sans-Serif	Usabilidade	Desejável	[RF002]
[RNF004]	O layout da plataforma deve ser responsivo	Usabilidade	Importante	[RF002]
[RNF005]	A plataforma deve estar disponível 24h por dia	Disponibilidade	Essencial	[RF001] e [RF002]

Fonte: Autores (2024)

3.3. Regras de Negócio

As regras de negócio definem a forma de fazer o negócio, refletindo a política interna, o processo definido e/ou as regras básicas de conduta, ou seja, é um conjunto de instruções que os usuários já seguem e que o sistema a ser desenvolvido deve contemplar. Na Tabela 4 estão as regras de negócio para realização das funcionalidades da Plataforma Code Debt.

Tabela 4 - Regras de Negócio

ID	Descrição	Prioridade	Requisitos Relacionados
[RN004]	O conteúdo da plataforma deve abordar sobre ML e DT e sobre o Projeto FindDT	Essencial	[RF001], [RF002] e [RF003]

Fonte: Autores (2024).

4. MODELAGEM DA PLATAFORMA CODE DEBT

Para a modelagem da plataforma, foram escolhidos dois dos diagramas da UML (*Unified Modeling Language*), que é uma linguagem de notação para uso em projetos de sistemas. O presente trabalho utilizou o Diagrama de Casos de Uso e Diagrama de Sequência.

4.1. Diagrama de Casos de Uso

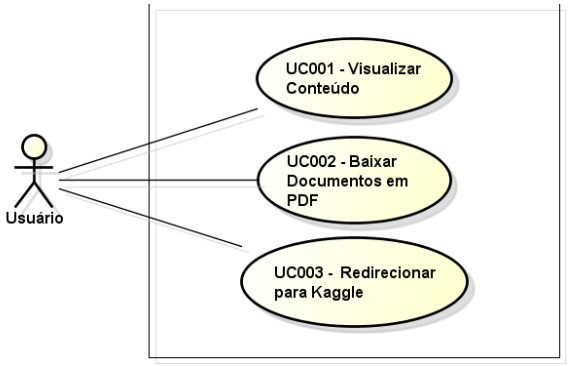
O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem nele. Ele é utilizado para capturar os requisitos da plataforma, ou seja, o que o software deve fazer a fim de atender as necessidades das partes interessadas. Os atores identificados no contexto deste projeto estão descritos na Tabela 5 e o diagrama de casos de uso da plataforma pode ser observado na Figura 1.

Tabela 5 - Atores

Ator	Descrição
Usuário	Vai acessar as informações do site

Fonte: Autores (2024).

Figura 1 - Diagrama de Caso de Uso

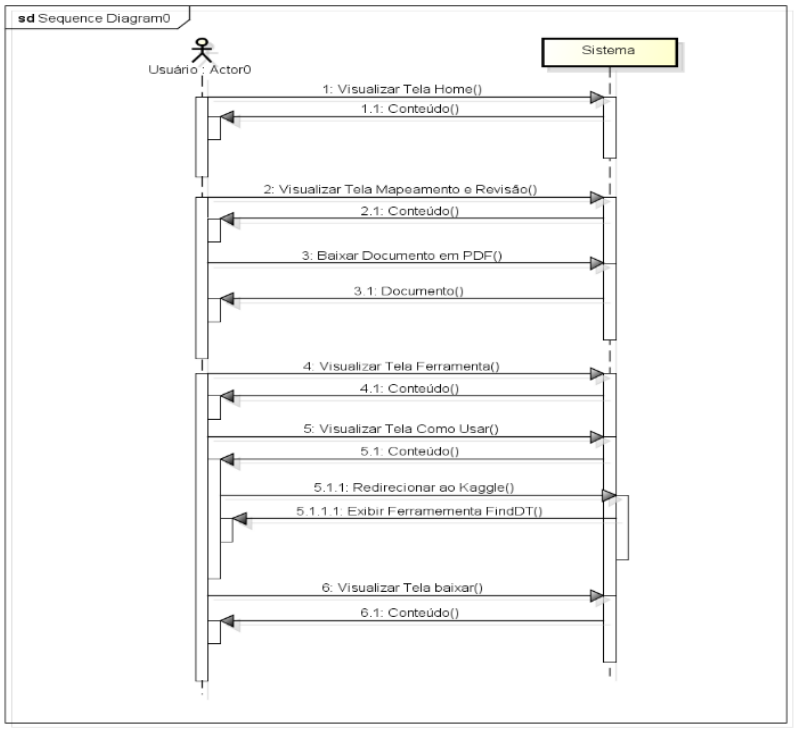


Fonte: Autores (2024)

4.2. Diagrama de Sequência

Um diagrama de sequência mostra a interação entre objetos ao longo do tempo, ou seja, exemplifica como os objetos se comunicam e trocam mensagens para realizar uma determinada funcionalidade em um sistema. Na Figura 2 é possível observar como ocorre a sequência de passospara que o usuário consiga baixar um arquivo.

Figura 2 - Diagrama de Sequência

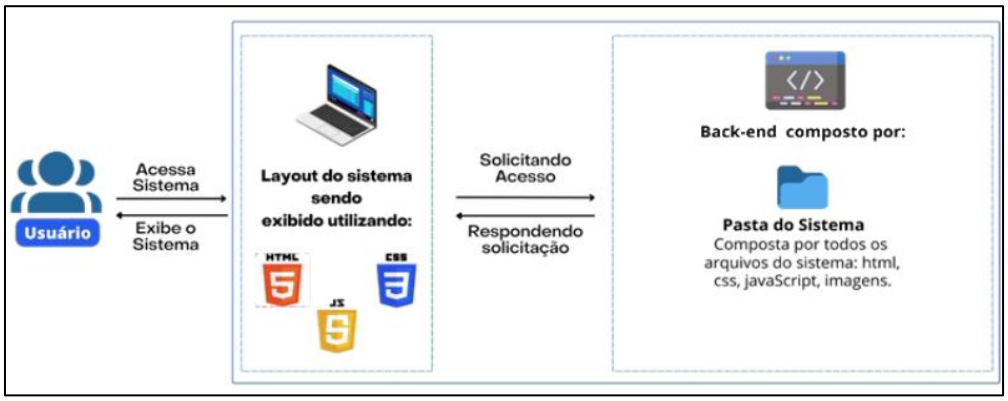


Fonte: Autores (2024)

5. ARQUITETURA DA PLATAFORMA CODE DEBT

A plataforma foi implementada utilizando HTML, CSS e JavaScript para criar da plataforma. Essas linguagens são ideais para o desenvolvimento web e permitem a construção de interfaces dinâmicas e responsivas. Focando na criação de uma experiência de usuário envolvente com recursos de HTML5 para estruturação, CSS3 para estilização e JavaScript para interatividade e animações. A arquitetura pode ser observada na Figura 3.

Figura 3 - Arquitetura



Fonte: Autores (2024)

6. IMPLEMENTAÇÃO DA PLATAFORMA CODE DEBT

A Plataforma Code Debt tem como objetivo disponibilizar os resultados de um projeto de pesquisa e está disponível no link: https://projetosufam.com.br/codedebt. Na página principal temos visão abrangente do conteúdo disponível, iniciando com uma mensagem introdutória de boas-vindas (Figura 4). Em seguida, é apresentado informações resumidas sobre os resultados das pesquisas realizadas, cada um acompanhado por um link para acesso direto (Figura 5). Logo após, são fornecidas informações sobre o Projeto de Pesquisa Super, incluindo um breve resumo e um link para a página oficial do projeto, caso o usuário deseje conhecer mais detalhes (Figura 6). Por fim, é apresentado a equipe de desenvolvimento do projeto, onde são mostrados alguns dados dos autores (Figura 7). Este layout foi projetado para garantir uma experiência intuitiva e informativa aos usuários da plataforma.

Na parte superior, temos o seguinte menu: (1) Home, (2) Mapeamento Sistemático, (3) Revisão Rápida e (4) Ferramenta FindDT. O item (1) se refere a página principal e os itens (2), (3) e (4) dão acesso rápido aos resultados científico das pesquisas, que também pode acessado conforme mostra a Figura 5, e que serão explicados com mais detalhes abaixo.

Figura 4 - Objetivo da Plataforma

Fonte: Autores (2024)

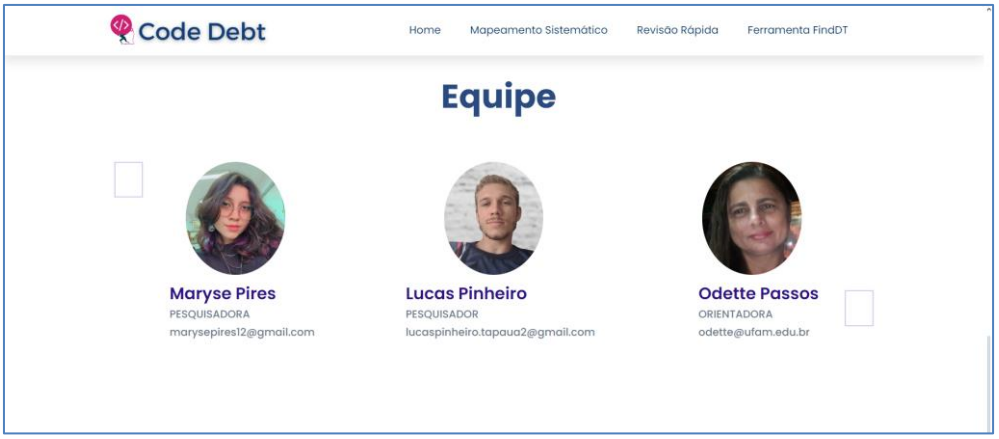
Figura 5 - Resultados da Pesquisa

Fonte: Autores (2024)

Figura 6 - Projeto Super

Fonte: Autores (2024)

Figura 7 - Sobre os Autores



Fonte: Autores (2024)

6.1. Item (2): Mapeamento Sistemático

No item “Mapeamento Sistemático” primeiramente é apresentado um resumo sobre a condução do MS (Figura 8). Em seguida, é apresentado o objetivo e as fontes de busca utilizadas (Figura 9) e as questões de pesquisa (Figura 10). Como complemento, foi disponibilizado a opção de "Baixar Documento" (Figura 11), oferecendo aos interessados o acesso ao relatório completo em formato PDF. Este documento contém desde o planejamento inicial da pesquisa até a exposição minuciosa dos resultados obtidos, proporcionando uma visão abrangente e detalhada do estudo realizado.

Figura 8 - Mapeamento Sistemático

Fonte: Autores (2024)

Figura 9 - Objetivos e Fontes de Pesquisa

Fonte: Autores (2024)

Figura 10 - Questões de Pesquisa

Fonte: Autores (2024)

Figura 11 - Baixar Documento

Fonte: Autores (2024)

6.2. Item (3): Revisão Rápida

No item “Revisão Rápida” (Figura 12) é apresentado um resumo contextualizando o tema de pesquisa e o que se busca pesquisar. Em seguida, é descrito o objetivo e a fonte de pesquisa utilizada (Figura 13), bem como as questões de pesquisa abordadas (Figura 14). Além disso, existe a opção de "Baixar Documento", onde é disponibilizado o documento completo da Revisão Rápida em formato PDF para acesso (Figura 15). Este documento contém desde o planejamento inicial da pesquisa até a exposição minuciosa dos resultados obtidos, proporcionando uma visão abrangente e detalhada do estudo realizado.

Figura 12 - Revisão Rápida

Fonte: Autores (2024)

Figura 13 - Objetivos e Fontes de Pesquisa

Fonte: Autores (2024)

Figura 14 - Questões de Pesquisa

Fonte: Autores (2024)

Figura 15 - Baixar Documento

Fonte: Autores (2024)

6.3. Item (4): Ferramenta FindDT

Em “Ferramenta FindDT” é apresentado um resumo sobre a problemática, o objetivo da ferramenta e alguns resultados (Figura 16). Em seguida, descrevemos suas principais funcionalidades (Figura 17), bem como foi realizado todo o processo de implementação e treinamento dos modelos de ML (Figura 18). Além disso, foi disponibilizada a alternativa de “Baixar Documento” (Figura 19), proporcionando aos interessados o acesso ao relatório integral em PDF da plataforma Code Debt e da Ferramenta FindDT. Por fim, o usuário é convidado a acessar os itens “Como Usar” e “Como Baixar” o FindDT (Figura 19).

Ao selecionar a opção “Como Usar” apresentamos um guia abrangente para auxiliar os usuários na utilização da ferramenta FindDT (Figura 20), disponível na plataforma Kaggle. No “Como Baixar” disponibilizamos um tutorial detalhado sobre como baixar o banco de dados utilizado e os modelos de ML pré-treinados (Figura 21).

Figura 16 - Ferramenta FindDT

Fonte: Autores (2024)

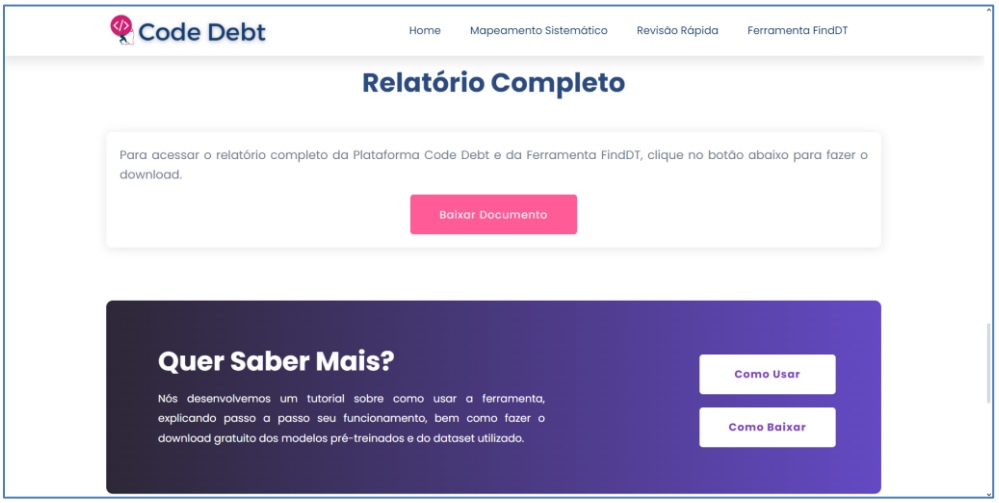
Figura 17 - Funcionalidade do FindDT

Fonte: Autores (2024)

Figura 18 - Implementação e Treinamento

Fonte: Autores (2022)

Figura 19 - Quer Saber Mais?

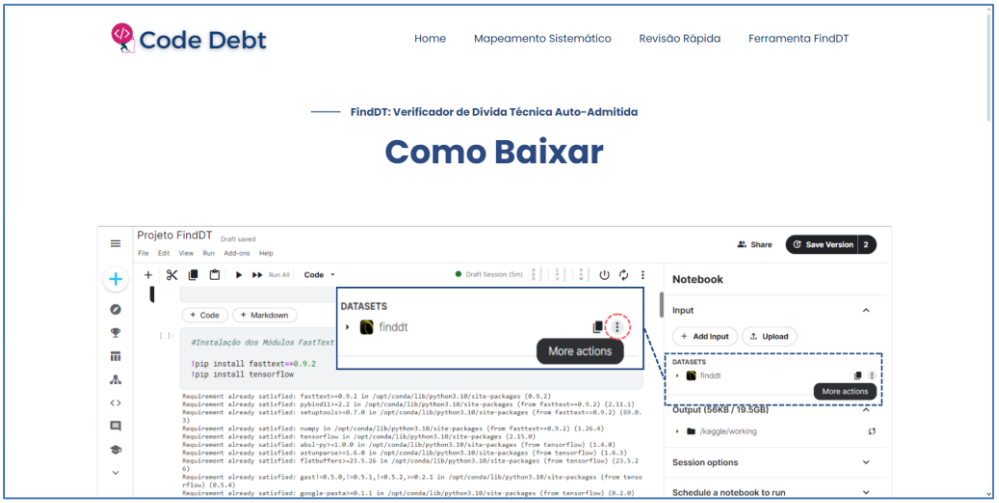


Fonte: Autores (2022)

Figura 20 - Como Usar

Fonte: Autores (2024)

Figura 21 - Como Baixar



Fonte: Autores (2024)

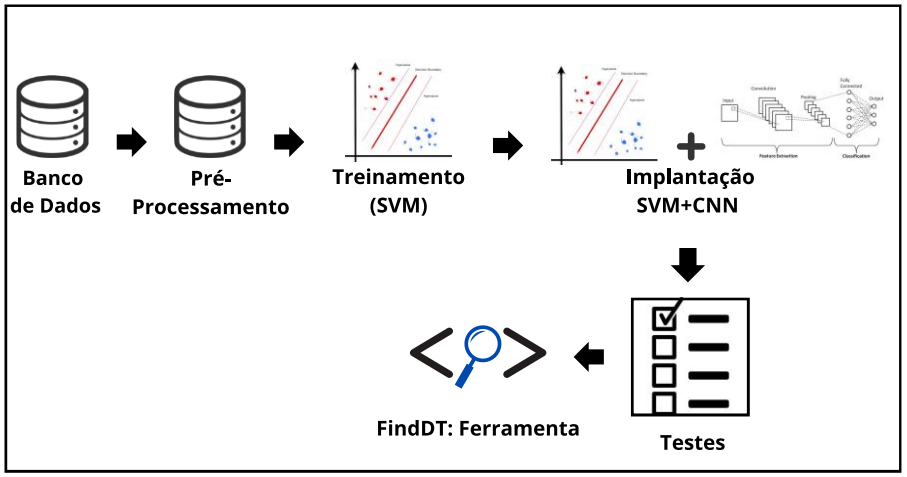
6.3.1 Metodologia da Ferramenta FindDT

Identificar a DTAA no código-fonte é um dos desafios presentes no desenvolvimento de software contemporâneo. Essa dívida reflete compromissos na qualidade do código, frequentemente decorrentes de decisões de design ou implementação tomadas rapidamente e de forma temporária. Para a aquisição de conhecimento realizamos uma Revisão Rápida voltada a DT e um Mapeamento Sistemático voltado a área de ML, onde identificamos o algoritmo que melhor se encaixava a nossa problemática.

FindDT é uma ferramenta de ML desenvolvida para identificar a DTAA em código-fonte. Baseando-se em um trabalho prévio de Li, Soliman e Avgeriou (2022), disponível no GitHub, a ferramenta utiliza um modelo pré-treinado e uma base de dados extraída desse trabalho como ponto de partida. Além disso, foi desenvolvido um modelo adicional utilizando SVM (Support Vector Machine) para identificação dos diferentes tipos de DT.

A metodologia utilizada para o desenvolvimento da FindDT foi a Prototipação, um processo iterativo de desenvolvimento que envolve a criação de modelos iniciais ou protótipos para validar conceitos, testar funcionalidades e obter feedback dos usuários (Oliveira, Fontoura e Medina, 2020). Nesse contexto foi seguido as etapas identificadas no trabalho de Li, Soliman e Avgeriou (2022), com foco na identificação de DTAA em código-fonte. Os principais passos estão descritos na Figura 22.

Figura 22 - Metodologia do FindDT

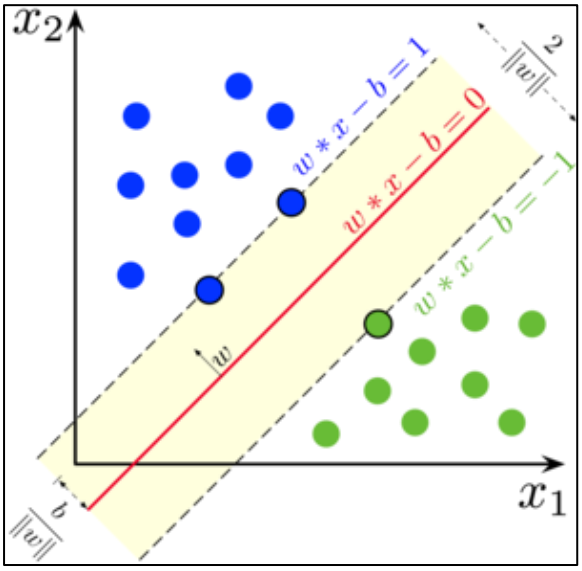


Fonte: Autores (2024)

6.3.2 Algoritmos SVM e CNN

O classificador SVM (Support Vector Machine), conhecido como Máquina de Vetores de Suporte, é amplamente empregado em várias tarefas de classificação. Este algoritmo supervisionado é capaz de separar dados em diferentes classes ao construir um hiperplano (ou conjunto de hiperplanos) em um espaço dimensional elevado (Figura 23). Embora ideal para problemas de classificação binária, o SVM também é aplicável em cenários multi-classe. Sua robustez permite lidar com dados de treinamento ruidosos, e sua eficiência computacional é destacável (Nissila, 2023).

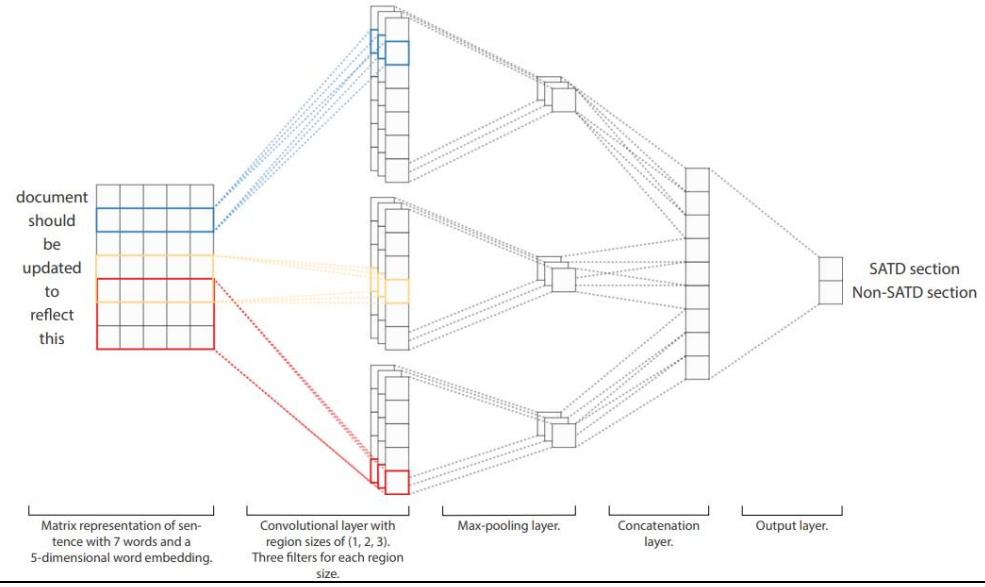
Figura 23 - Modelo SVM



Fonte: Máquina de Vetores de Suporte (2022)

Uma CNN (Convolutional Neural Network) é um tipo de rede neural, frequentemente utilizada em tarefas de visão computacional. O principal componente de uma CNN são as camadas convolucionais, que aplicam filtros para extrair características importantes das entradas, como pixels de uma imagem (Figura 24). Essas características são então passadas por camadas de pooling para reduzir a dimensionalidade e finalmente conectadas a camadas totalmente conectadas para realizar a classificação ou regressão. As CNNs são eficazes em aprender representações hierárquicas de dados e são amplamente empregadas em aplicações onde a localidade espacial das características é relevante, como em problemas de processamento de imagens e sequências (Kattenborn, 2021).

Figura 24 - Modelo CNN

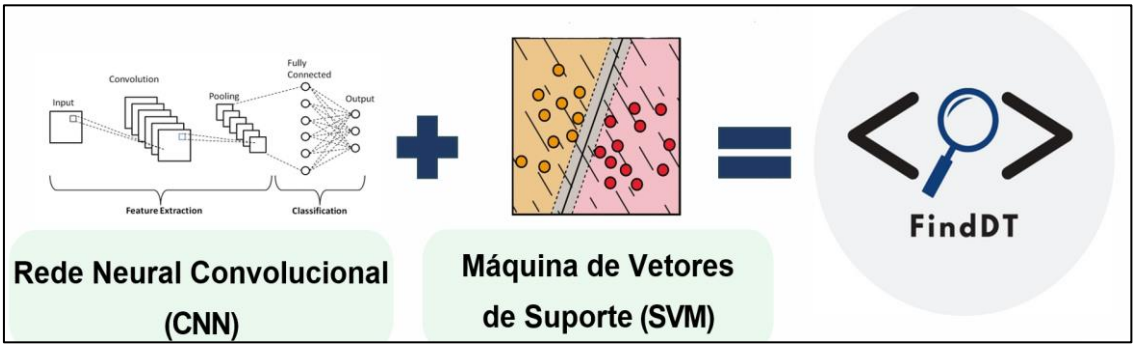


Fonte: Li, Soliman e Avgeriou (2022)

6.3.3 Arquitetura do FindDT

Nossa ferramenta foi desenvolvida com base em uma arquitetura composta por dois modelos principais: um modelo baseado em CNN (Rede Neural Convolucional) e um modelo SVM (Máquina de Vetores de Suporte), conforme ilustra a Figura 25. O modelo CNN foi extraído do trabalho de Li, Soliman e Avgeriou (2022).

Figura 25 - Arquitetura do FindDT



Fonte: Autores (2024)

Após realização do Mapeamento Sistemático, identificamos alguns pontos que destacam a vantagem do modelo SVM para a classificação de DT em código-fonte. De acordo com nossa análise, o SVM é uma escolha promissora devido à sua capacidade de lidar com dados complexos e de alta dimensionalidade, comum na análise de código-fonte. Sua robustez contra ruídos nos dados também se mostrou relevante, permitindo distinguir padrões significativos daqueles que são irrelevantes ou ruidosos. Portanto, com base nesses resultados, concluímos que o modelo SVM é uma escolha sólida e promissora para a classificação de DT em código-fonte, oferecendo uma combinação de desempenho, robustez e eficiência computacional.

Além disso, utilizamos a Plataforma Kaggle para a implementação e treinamento da ferramenta. Ela é reconhecida como uma plataforma popular que oferece oportunidades para os usuários participarem de competições de ML explorarem conjuntos de dados e publicarem seus trabalhos, além de acessarem treinamentos. É um ambiente propício para interação, conexão e colaboração entre Cientistas de Dados visando a construção de modelos de ML de qualidade (Matos, 2020).

Durante nossa implementação na Plataforma Kaggle, utilizamos a linguagem de programação Python juntamente com os frameworks como TensorFlow, FastText e Joblib. Essa linguagem é amplamente reconhecida na comunidade de Ciência de Dados e ML por sua versatilidade e eficácia no desenvolvimento de modelos e análise de dados.

6.3.4 Banco de Dados e Pré-Processamento

A base de dados utilizada neste estudo foi originada do trabalho de Li, Soliman e Avgeriou (2022), contendo um total de 5.556 registros de dados relacionados a diferentes tipos de DT. Esses dados abrangem 8 categorias de DT, nomeadamente: Arquitetura, Código, Design, Documentação, Defeitos, Construção, Requisitos e Testes, contendo as colunas: Projeto, Chave, Sessão, Tipo, Indicador e Texto.

Após uma revisão rápida aprofundada, identificamos que dentre os tipos de DT originalmente listados, cinco deles podem ser identificados a partir do código-fonte. Os detalhes sobre esses tipos e a quantidade de registro de dados estão disponíveis na Tabela 6.

Concluindo a análise, realizou-se um pré-processamento dos dados, que incluiu a limpeza de registros vazios e a remoção de duplicatas. Esse procedimento garantiu que a base de dados estivesse completamente utilizável, com um total de 2.759 registros de dados para uso e treinamento, sem registros desnecessários ou duplicados, preparando-a adequadamente para a análise e classificação dos diferentes tipos de DT identificáveis a partir do código-fonte.

Tabela 6 - Tipos de DT

Tipos de DT	Registros de Dados
Design	933
Arquitetura	87
Código	1242
Requisitos	95
Testes	338
Total	**2.759**

Fonte: Autores (2024)

6.3.5 Treinamento

No tópico de Treinamento, conduzimos nosso estudo na Plataforma Kaggle com o objetivo de identificar a DTAA (Figura 26). Dividimos nossa base de dados em conjuntos de treinamento e teste, utilizando 80% dos dados para treinar o modelo e reservando os 20% restantes para avaliação. O modelo SVM (Support Vector Machine) foi empregado nesse contexto.

Figura 26 - Treinamento

Fonte: Autores (2024)

6.3.6 Implantação

O primeiro modelo, utilizando CNN, é responsável por identificar se há ou não a presença de DT em determinados trechos de código ou documentos. Essa abordagem de aprendizado profundo permite uma análise mais sofisticada e eficiente, capturando padrões complexos nos dados de entrada e fazendo previsões precisas sobre a presença da DT.

O segundo modelo, implementado com SVM, é encarregado de identificar o tipo específico de DT quando detectada pelo modelo CNN. Neste contexto, o SVM é treinado para reconhecer padrões associados a diferentes tipos de DT, como má prática de código, dependências desatualizadas, falta de documentação, entre outros.

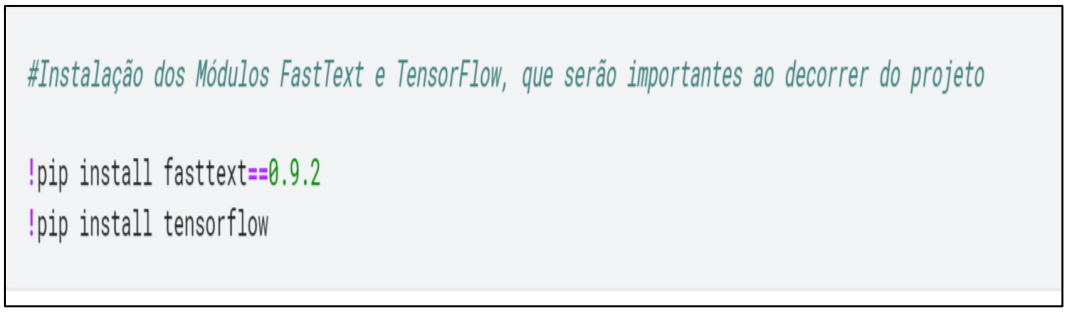
A combinação desses dois modelos permite uma abordagem abrangente e eficaz na identificação e classificação da DT, fornecendo insights valiosos para os desenvolvedores e equipes de gerenciamento de software. Essa estrutura modular e multifacetada torna nossa ferramenta versátil e adaptável a uma variedade de contextos e tipos de aplicação, contribuindo significativamente para o processo de desenvolvimento e manutenção de software.

6.3.7 Resultados e Testes

FindDT utiliza um modelo CNN pré-treinado para analisar os comentários e identificar possíveis áreas de DT no código-fonte. Em seguida, aplica-se um modelo SVM para classificar os diferentes tipos de DT encontrados, tais como código complexo, falta de documentação, entre outros.

Na Plataforma Kaggle, FindDT está disponível juntamente com o código de treinamento utilizado para desenvolver o modelo SVM. Para que seja possível utilizá-lo é preciso seguir alguns passos. Dentro da Plataforma Kaggle, o primeiro passo consiste em iniciar a sessão para carregar os dados do conjunto de dados intitulado “Debt Code”. Em seguida, deve ser feita a instalação dos módulos de fastText e tensorFlow, localizados na segunda célula de Código. Esses módulos são importantes para a execução da ferramenta, conforme visto na Figura 27.

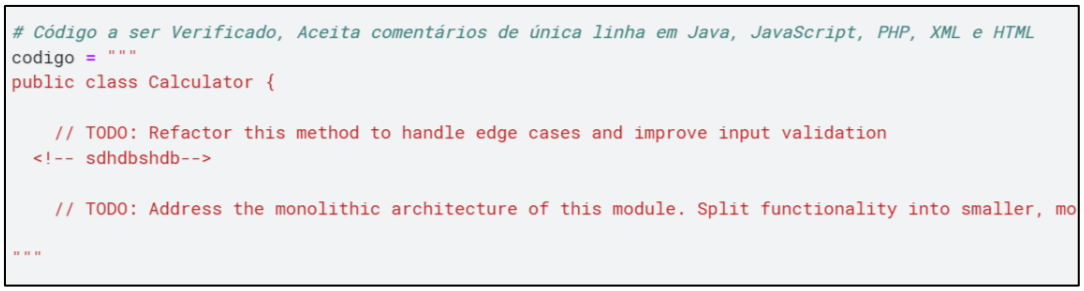
Figura 27 - Módulos Impostastes



Fonte: Autores (2024)

A ferramenta conta com uma seção dedicada à inserção do código-fonte a ser testado, que está localizada abaixo da classe “Find”. Nessa seção, o usuário pode inserir o código desejado, para assim obter a verificação e os resultados detalhados da predição, conforme ilustrado na Figura 28.

Figura 28 - Área de Código



Fonte: Autores (2024)

FindDT extrai os comentários identificados no código e os armazena para predição, como mostrado na Figura 29. Além disso, é capaz de identificar comentários em diversas linguagens de programação, incluindo Java, JavaScript, PHP, XML e HTML. A ferramenta utiliza a biblioteca finditer, que é responsável pela extração dos comentários de diferentes linguagens de programação, oferecendo uma solução versátil e eficaz para uma variedade de situações.

Ao executar a ferramenta, a classe denominada “Find” é inicializada, prepara o modelo CNN com seus pesos e um tokenizador, além de definir as configurações de rótulo que denominam se existe DT ou não. A Figura 30 ilustra essa ação.

Figura 29 - Função de Extração de Comentários

```
    def extrair_comentarios(self, codigo):
        comentarios = []

        # Extraindo comentários de uma linha em Java, JavaScript e PHP
        comentarios_linha_unico_java = re.finditer(r"//(.+)", codigo)
        for correspondencia in comentarios_linha_unico_java:
            comentario = correspondencia.group(1).strip()
            idx_inicio = correspondencia.start()

            # Indentifica o número da linha onde o comentário está
            numero_linha = codigo.count('\n', 0, idx_inicio) + 1
            # Adiciona Comentário e Linha na lista
            comentarios.append((comentario, numero_linha))

        # Extraindo comentários de uma linha XML e HTML
        comentarios_xml = re.finditer(r"<!--(.*?)-->", codigo, re.DOTALL)
        # Indentifica o número da linha onde o comentário está
        for correspondencia in comentarios_xml:
            linhas = correspondencia.group(1).strip().split('\n')
           # Adiciona Comentário e Linha na lista
            for linha in linhas:
                comentarios.append((linha.strip(), codigo[:codigo.index(linha)].count('\n') + 1))

        # Retorna a lista de Comentários e a Linhas
        return comentarios
```

Fonte: Autores (2024)

Figura 30 - Modelo CNN

```
def __init__(self, arquivo_pesos, arquivo_embutimento_palavras):

    # Carregar o modelo e seus pesos
    print('Carregando modelo {}...'.format(arquivo_pesos))
    self._modelo = load_model(arquivo_pesos)
    self._tamanho_entrada = self._modelo.layers[0].get_output_at(0).get_shape()[1]

    # Carregar os embeddings de palavras FastText
    self._embutimento_palavras = fasttext.load_model(arquivo_embutimento_palavras)
    self._cache_embutimento_palavras = {}

    # Inicializar o tokenizador
    self._tokenizador_palavras = nltk.TweetTokenizer()

    # Configurações de rótulo
    self._rotulos = ['SATD', 'non-SATD']
    self._preenchimento = '<pad>'
```

Fonte: Autores (2024)

Em seguida a função “classificar comentário” é chamada, recebendo como entrada um comentário e o número da linha. Essa função, utilizando o modelo CNN, é encarregada de realizar a previsão do comentário e exibir o resultado correspondente. Para isso, ela utiliza a função "tokenizador", que recebe o comentário como entrada e realiza a tokenização em palavras. Esse procedimento envolve a divisão do comentário em palavras, preservando palavras compostas. Dessa forma, os comentários encontrados são preparados para predição. A Figura 31 mostra a função de tokenização. Se o comentário for classificado como DTAA, o modelo SVM é então carregado para classificar seu tipo, culminando na exibição do resultado, conforme ilustrado na Figura 32.

Figura 31 - Tokenizador

```
def tokenizar_comentario(self, comentario):
    # tokens_frases é uma lista de listas de tokens, onde cada lista interna representa uma frase
    # A linha abaixo quebra o comentário em frases
    tokens_frases = [self._tokenizador_palavras.tokenize(frase) for frase in
    nltk.sent_tokenize(comentario)]

    # A linha abaixo as frases em Palavras, armazenandos em uma só lista
    tokens = [palavra for frase in tokens_frases for palavra in frase]

    # Retorna a lista de tokens resultante
    return tokens
```

Fonte: Autores (2024)

Figura 32 - Classificar Comentário

```
def classificar_comentario(self, comentario, linha):

    # Preparar o comentário para classificação
    entrada_x = self.preparar_comentarios(comentario)

    # Fazer previsões usando o modelo
    previsao_y = self._modelo.predict(entrada_x)
    previsao_booleana_y = np.argmax(previsao_y, axis=1)
    result = self._rotulos[previsao_booleana_y[0]]

    # Imprimir os resultados da previsão
    print('Linha: {}'.format(linha))
    print('Texto: {}'.format(comentario)+'"')
    print('Previsão: {}'.format(self._rotulos[previsao_booleana_y[0]]))

    # Se for SADT prever o tipo e imprime
    if result == "SATD":
        query_result = v.consultar_tipo(comentario)
        print('Tipo: ' + query_result[0])
        print("\n")
    else:
        print("\n")
```

Fonte: Autores (2024)

Após a identificação feita pelo modelo CNN, o modelo SVM utilizando a biblioteca Joblib e realiza a predição do tipo de DT. Essa biblioteca é responsável apenas pelo carregamento do modelo e seu vetorizador. A Figura 33 mostra o carregamento do modelo SVM e o vetorizador.

Figura 33 - Modelo SVM

```
def consultar_tipo(self, texto):
    # Carregar o modelo SVM e o vetorizador TF-IDF a partir dos arquivos salvos
    modelo_svm = joblib.load("/kaggle/input/finddt/modelo_svm_pre_treinado_tipo.joblib")
    vetorizador_tfidf = joblib.load("/kaggle/input/finddt/modelo_tfidf_pre_treinado_tipo.joblib")

    # Vetorizar o texto usando o vetorizador TF-IDF, podendo assim ser usado como entrada para o Mo
    texto_tfidf = vetorizador_tfidf.transform([texto])

    # Fazer previsão usando o modelo SVM
    previsao = modelo_svm.predict(texto_tfidf)

    # Retornar a string indicando o tipo de SADT
    return previsao
```

Fonte: Autores (2024)

Com base nas análises feitas pelos modelos, FindDT gera relatórios detalhados sobre a localização, natureza e gravidade da DT identificada. A Figura 34 apresenta um exemplo de execução da ferramenta, onde é apresentado o relatório contendo a predição positiva para DTAA. Se a predição for negativa para DTAA, o tipo de DT não será apresentado no relatório da ferramenta, conforme pode ser observado na Figura 35.

Esses relatórios são essenciais para facilitar o planejamento e a priorização das atividades de refatoração, permitindo que as equipes de desenvolvimento foquem nos pontos mais críticos e urgentes, visando aprimorar a qualidade do código-fonte.

Figura 34 - Exemplo de Execução Positiva

```
1/1 [==============================] - 0s 247ms/step
Linha: 4
Texto: TODO: Refactor this method to handle edge cases and improve input validation"
Previsão: SATD
Tipo: code_debt
```

Fonte: Autores (2024)

Figura 35 - Exemplo de Execução Negativa

```
1/1 [==============================] - 0s 28ms/step
Linha: 7
Texto: TODO: Address the monolithic architecture of this module. Split functionality into smaller,
nts to improve modularity and maintainability."
Previsão: non-SATD
```

Fonte: Autores (2024)

Os resultados obtidos em relação a tarefa de classificação do tipo de DTAA foi de uma precisão (acurácia) de 65% e um F1-Score de 64%, conforme mostra a Figura 36, além dos testes realizados utilizando os dois modelos em conjunto, que atingiram um F1-Score de 68%. Fazendo um paralelo com o estudo de Li, Soliman e Avgeriou (2022), que alcançou um F1-Score de 68% somente na identificação da DTAA, sem incluir seu tipo. Essas métricas são indicadores essenciais do desempenho do modelo na identificação correta das classes de DT.

Figura 36 - Teste SVM

	precision	recall	f1-score	support
architecture_debt	1.00	0.16	0.28	25
build_debt	0.43	0.50	0.46	6
code_debt	0.64	0.71	0.67	250
design_debt	0.63	0.63	0.63	187
requirement_debt	1.00	0.50	0.67	18
test_debt	0.70	0.71	0.71	66
accuracy			0.65	552
macro avg	0.73	0.53	0.57	552
weighted avg	0.67	0.65	0.64	552

Fonte: Autores (2024)

A avaliação realizada nos proporcionou insights valiosos sobre o desempenho do SVM, especificamente para essa tarefa de identificação de DTAA. Os resultados da acurácia nos permitiram ajustar e otimizar a abordagem de ML, visando alcançar maior precisão e eficácia na identificação, e classificação da DT em conjuntos de dados na Plataforma Kaggle.

7. CONSIDERAÇÕES FINAIS

Este projeto resultou na criação da plataforma educacional denominada Code Debt, desenvolvida como parte de um projeto de pesquisa no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER). O Code Debt tem como objetivo central compartilhar informações sobre uma ferramenta inovadora voltada para identificação de Dívida Técnica Auto-Admitida em código-fonte.

A plataforma está disponível e pode ser acessada por meio do link https://projetosufam.com.br/codedebt. Ela contém informações sobre o projeto, compartilhando informações relevantes sobre a ferramenta desenvolvida, incluindo dois Relatórios Técnicos Científicos gerados a partir de um Mapeamento Sistemático e de uma Revisão Rápida. Além disso, a plataforma disponibiliza um Guia detalhado sobre como acessar e utilizar a ferramenta, incluindo instruções para download do banco de dados e modelos pré-treinados utilizados durante o desenvolvimento.

O FindDT representa não apenas uma ferramenta, mas também um resultado significativo de pesquisa e desenvolvimento na área da identificação automatizada da DT, contribuindo para o avanço do conhecimento e compartilhamento de recursos na comunidade acadêmica e profissional.

AGRADECIMENTOS

Esta pesquisa, realizada no âmbito do Projeto Samsung-UFAM de Ensino e Pesquisa (SUPER), de acordo com o Artigo 39 do Decreto n° 10.521/2020, foi financiada pela Samsung Eletrônica da Amazônia Ltda, nos termos da Lei Federal n° 8.387/1991, através do convênio 001/2020 firmado com a UFAM e FAEPI, Brasil.

REFERÊNCIAS

Kattenborn, T.; Leitloff, J.; Schiefer, F. e Hinz, S. (2021). **Review on Convolutional Neural Networks (CNN) in Vegetation Remote Sensing**. ISPRS Journal of Photogrammetry and Remote Sensing, v. 173, p. 24-49.

Kruchten, P.; Nord, R. e Ozkaya, I. (2012)**. Technical Debt: From Metaphor to Theory and Practice.** IEEE Software, v. 29, n. 6, p. 18-21.

Li, Y.; Soliman, M. e Avgeriou, P. (2022). **Identifying Self-Admitted Technical Debt in Issue Tracking Systems Using Machine Learning**. Empirical Software Engineering, v. 27, n. 131.

Lima, B. (2021)**. Uma Abordagem de Priorização para Apoiar o Pagamento de Dívida Técnica Auto-Admitida em Código-Fonte**.

Nissila, L. (2023). **Análise Comparativa de Métodos Computacionais para a Classificação de Estoque: Redes Neurais, kNN e SVM**. 68 f. Monografia (Graduação em Engenharia de Produção) - Universidade Federal de Ouro Preto

Máquina de Vetores de Suporte. (2022). In: Wikipédia, a Enciclopédia Livre. Flórida: Wikimedia Foundation. Disponível em: <https://pt.wikipedia.org/w/index.php?title=M%C3%A1quina_de_vetores_de_suporte&oldid=64412706>. Acesso em: 29 mar. 2024.

Matos, D. (2020). **O Kaggle é Realmente Válido Para Aprender Data Science?**. Disponível em: <https://www.cienciaedados.com/o-kaggle-e-realmente-valido-para-aprender-data-science/>. Acesso em: 01 abr. 2024.

Oliveira, P; Fontoura, L e Medina, D. (2020). **Metodologias Usadas no Desenvolvimento de Jogos Eletrônicos Educacionais: Uma Revisão da Literatura**. Anais do XXXI Simpósio Brasileiro de Informática na Educação, p. 542-551.