FICHA TÉCNICA

Equipe responsável pela elaboração

****

Público alvo
**********.

Versão 1.0 - *****, Junho de 2024

Release 1.0

Sumário

1 Visão Geral do documento

Este documento está estruturado de forma a apresentar informações sobre o sistema em questão. Na **Seção 2**, é feita uma descrição geral do sistema, contemplando sua visão geral, necessidade e benefícios esperados. Na **Seção 3**, são apresentados todos os requisitos funcionais e não-funcionais do sistema, com descrição, processo, entradas, saídas e prioridade. A **Seção 4** é dedicada à descrição do pipeline da ferramenta, que consiste em uma sequência de etapas envolvendo pré-processamento, engenharia de

características, modelos de Machine Learning e otimização de hiperparâmetros. Cada etapa tem como objetivo preparar os dados para análise e construir um modelo que possa fazer previsões precisas. Na **Seção 5**, a arquitetura do sistema e a estrutura de diretórios são detalhadas. Na **Seção 6**, o diagrama de classe preliminar do sistema é apresentado, descrevendo suas classes e métodos. A **Seção 7** é dedicada à implementação do sistema. Na **Seção 8**, os testes realizados no sistema e um resumo dos resultados obtidos são apresentados. Por fim, na **Seção 9**, são descritos os requisitos mínimos para a instalação adequada da ferramenta.

2. Descrição Geral do Sistema

O Auto Machine Learning (AutoML) é uma técnica de automação de processos de machine learning que utiliza algoritmos para selecionar modelos, ajustar hiperparâmetros e pré-processar dados sem a necessidade de intervenção humana. Essa abordagem tem sido amplamente adotada em diversas áreas para aumentar a eficiência e reduzir a complexidade de tarefas de análise de dados, permitindo que usuários com pouca ou nenhuma experiência em machine learning possam criar modelos precisos de maneira mais rápida e eficiente.

O desenvolvimento do DroidAutoML foi motivado pela necessidade de um sistema de AutoML para o domínio de detecção de malwares android. Essa necessidade surgiu porque ferramentas populares de domínio específico, como TPOT, AutoGloun e AutoSklearn, muitas vezes não conseguem lidar com conjuntos de dados específicos do domínio android, como é o caso do dataset androcrawl.csv. Além disso, essas ferramentas apresentam limitações na interpretabilidade dos modelos gerados.

2.1 Benefícios Esperados

Nº	Descrição
1	**Economia de tempo:** a automação de tarefas de machine learning permite que os usuários criem modelos mais rapidamente, sem a necessidade de passar horas ajustando parâmetros e selecionando modelos.
2	**Aumento da eficiência:** a automatização de tarefas repetitivas e demoradas permite que os usuários concentrem seus esforços em tarefas mais importantes, como a análise e interpretação dos resultados.
3	**Redução de erros:** a automação de processos pode reduzir a probabilidade de erros humanos, melhorando a precisão dos modelos gerados.
4	**Maior acessibilidade:** uma ferramenta de AutoML pode permitir que usuários com pouco conhecimento em machine

	learning possam criar modelos de alta qualidade
5	**Melhoria na interpretabilidade:** a geração de modelos mais interpretáveis, pois a transparência e a explicabilidade são essenciais.
6	**Aumento da inovação:** explorar novas possibilidades de inovação e descoberta de insights.

3 Requisitos do sistema.

Pré-processamento de dados: O sistema deve permitir a realização de pré-processamento de dados para limpeza, normalização e tratamento de dados nulos e faltantes, balanceamento de dados e remoção de dados duplicados a fim de melhorar a qualidade dos dados de entrada para o processo de modelagem.

Engenharia de características: O sistema de AutoML deve permitir a seleção de características (ou features) que melhor representem os dados de entrada e permitam que o modelo aprenda a relação entre essas variáveis e a variável alvo. Reduzir a dimensionalidade dos dados pode afetar significativamente o desempenho do modelo e melhorar a interpretabilidade do modelo.

Seleção de modelos: O sistema de AutoML deve ter a capacidade de avaliar diferentes modelos de machine learning e selecionar automaticamente o modelo mais adequado com base nos requisitos e objetivos do usuário, bem como no desempenho histórico do modelo.

Otimização de modelos: O sistema de AutoML deve permitir a otimização automática de hiperparâmetros do modelo, a fim de melhorar a precisão e a eficiência do modelo gerado, além de reduzir o tempo de processamento necessário.

Esses requisitos do sistema são essenciais para garantir que o sistema de AutoML possa automatizar efetivamente tarefas complexas de machine learning, permitindo que usuários com pouca experiência em programação ou análise de dados possam criar modelos precisos e eficientes. Os requisitos serão discutidos em detalhes na seção de pipeline da ferramenta.

4 Pipeline da Ferramenta

A Figura 1 ilustra o pipeline de AutoML da MH-AutoML, com suas macro tarefas necessárias para a execução do fluxo de aprendizado de máquina: pré-processamento

dos dados, engenharia de características, seleção de modelo e ajuste de modelo. Nesta seção discutimos em detalhes cada etapa do pipeline da ferramenta e suas macro tarefas.

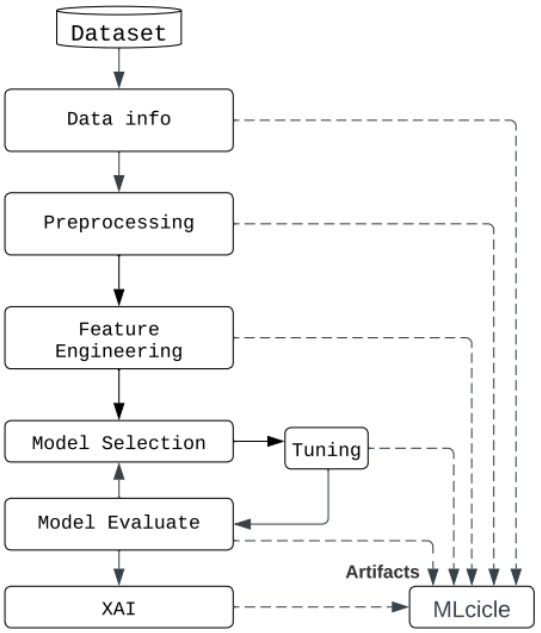


Figura 1. Fluxo do pipeline da MH-AutoML.

4.1 Pré-processamento

O pré-processamento de dados é uma etapa importante em Machine Learning, pois ele permite que os dados sejam preparados para que possam ser usados de forma mais eficaz pelos modelos. O pré-processamento envolve várias etapas, como a limpeza dos dados, que envolve a identificação e correção de erros e inconsistências nos dados, garantindo assim que os dados estejam prontos para serem usados pelos modelos de Machine Learning.

O livro "Data Mining: Concepts and Techniques" de Jiawei Han, Micheline Kamber e Jian Pei. O capítulo 3 aborda as técnicas mais comuns de pré-processamento de dados em mineração de dados, incluindo a remoção de dados duplicados, o tratamento de dados ausentes, o tratamento de valores discrepantes, a padronização e normalização de dados e a codificação de variáveis categóricas. Já o paper "Data Preprocessing Techniques for Classification without Discrimination" de Saikat Basu e Rina Ghose, revisa as principais técnicas de pré-processamento de dados para classificação, incluindo a remoção de dados duplicados, o tratamento de dados ausentes, o tratamento de valores discrepantes e a padronização e normalização de dados.

- Identificar e remover dados duplicados: dados duplicados podem afetar negativamente a precisão do modelo, pois eles inflam o tamanho do conjunto de dados, introduzindo um viés em favor de uma determinada classe. Portanto, é importante identificar e remover dados duplicados antes de usá-los para treinar o modelo. O paper "Data Preprocessing Techniques for Classification without Discrimination" de Saikat Basu e Rina Ghose, explora diferentes técnicas de pré-processamento, incluindo a remoção de dados duplicados.

- Identificar e remover valores ausentes: valores ausentes podem prejudicar a precisão do modelo, pois eles introduzem um viés no conjunto de dados e podem fazer com que o modelo perca informações importantes. É importante identificar e remover valores ausentes antes de usar o conjunto de dados para treinar o modelo. O paper "Dealing with Missing Data: Key Assumptions and Methods for Applied Analysis" de Michael J. Stern e outros, fornece uma revisão abrangente das técnicas mais comuns para lidar com dados ausentes.

- Identificar e remover valores discrepantes: valores discrepantes são valores que se desviam significativamente da distribuição normal do conjunto de dados e podem ser causados por erros de medição, falhas de registro ou outras razões. A presença de valores discrepantes pode afetar negativamente a precisão do modelo, tornando-o menos capaz de generalizar para dados não vistos. Portanto, é importante identificar e remover valores discrepantes antes de usar o conjunto de dados para treinar o modelo. O paper "Outlier Detection Techniques" de Charu Aggarwal, fornece uma revisão das principais técnicas de detecção de outliers e suas aplicações em diferentes áreas.

- Padronizar e normalizar os dados: a padronização e normalização dos dados é um processo importante para garantir que todos os recursos do conjunto de dados tenham uma escala comparável. Isso é importante porque muitos algoritmos de Machine Learning supõem que os dados estejam em uma escala comparável, o que pode não ser o caso se os recursos tiverem unidades ou escalas diferentes. O livro "Python Machine Learning" de Sebastian Raschka e Vahid Mirjalili, dedica um capítulo inteiro (Capítulo 4) para a pré-processamento de dados, incluindo a padronização e normalização dos dados.

- Identificar se os dados estão balanceados: o desbalanceamento no conjunto de dados pode levar a um modelo que favorece a classe majoritária, enquanto a classe minoritária é ignorada. Portanto, é importante identificar se os dados estão balanceados para garantir que o modelo seja capaz de aprender com todas as classes. Se os dados não estiverem balanceados, é importante usar técnicas de balanceamento, como oversampling ou undersampling, para garantir que o modelo possa aprender com todas as classes. Isso pode envolver a geração de dados sintéticos para a classe minoritária ou a remoção de dados da classe majoritária, dependendo da técnica de balanceamento escolhida. O paper "Learning from Imbalanced Data" de Haibo He e outros, que revisa os principais desafios enfrentados ao trabalhar com conjuntos de dados desbalanceados e as principais técnicas para abordar esse problema. Já o livro "Applied Predictive Modeling" de Max Kuhn e Kjell Johnson, que dedica um capítulo inteiro (Capítulo 16) para o balanceamento de classes em conjuntos de dados desbalanceados.

4.2 Engenharia de características

Engenharia de características: nesta etapa, a tarefa principal é identificar aquelas características mais relevantes para o modelo de aprendizagem. Por exemplo, na predição de malwares Android, utiliza-se características como permissões. Tipicamente, permissões como READ SMS e SEND SMS são relevantes, enquanto outras, como VIBRATE e SET TIME, podem ser descartadas.

A redução de características é tipicamente realizada através de métodos como PCA (Análise de Componentes Principais) ou algoritmos de seleção de características, como SelectKBest e RFE (Recursive Feature Elimination). Em domínios específicos, como detecção de malwares Android, temos também uma diversidade de métodos sofisticados e especializados que observam diferentes aspectos particulares do domínio, como o impacto de diferentes tipos de características dos APKs de acordo com sua relevância [Sun et al., 2016; Cai et al., 2021; Moutaz, 2020].

Por exemplo, enquanto métodos mais generalistas (e.g., PCA) consideram características como a permissão INTERNET, devido ao seu grau elevado de recorrência, métodos específicos ao domínio, como SigPID [Sun et al., 2016], excluem essa característica por gerar ambiguidade ao modelo, uma vez que a característica é tipicamente utilizada tanto por aplicativos malignos quanto benignos. Neste sentido, um dos principais objetivos da DroidAutoML é incorporar métodos de seleção de características específicas ao domínio, potencializando a construção de modelos menos ambíguos. Na versão atual, ela incorpora três métodos específicos do domínio: SigPID, RFG e JOWNDroid.

4.3 Modelos de Machine Learning

Modelos de Machine Learning: nesta etapa, são treinados diferentes modelos de Machine Learning para prever a variável de interesse. Existem muitos modelos diferentes que podem ser usados, no entanto, incorporamos os modelos mais comuns em ferramentas de AutoML como TPOT, AutoGloun e AutoSklearn, que implementam modelos como: ensemble (RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier,ExtraTreesClassifier, BaggingClassifier) DecisionTreeClassifier, KNeighborsClassifier e SVC.

4.4 Otimização de hiperparâmetros

Otimização de hiperparâmetros: nesta etapa, são ajustados os hiperparâmetros do modelo para obter o melhor desempenho possível. Os hiperparâmetros são os parâmetros do modelo que não são aprendidos diretamente durante o treinamento. Atualmente a

DroidAutoML implementa a biblioteca Optuna para a otimização de hiperparâmetros para modelos de machine learning.

5 Arquitetura

5.1 MVC (Model-View-Controller)

A DroidAutoML seguiu a arquitetura MVC proposta pela primeira vez em 1979 por Trygve Reenskaug. Dentre as vantagem do MVC destacam-se:

- Separação de responsabilidades: O modelo é responsável pelos dados e lógica de negócios, a visão é responsável pela apresentação da interface do usuário e o controlador é responsável pela comunicação entre o modelo e a visão. Essa separação de responsabilidades torna o código mais fácil de entender e modificar.

- Reutilização de código: Os componentes do MVC são independentes uns dos outros e podem ser reutilizados em outras partes do aplicativo ou em outros aplicativos.

- Facilidade de teste: Como os componentes do MVC são independentes uns dos outros, é mais fácil testar cada componente separadamente.

- Manutenção mais fácil: A separação de responsabilidades e a modularidade dos componentes tornam a manutenção do código mais fácil e menos propensa a erros.

- Flexibilidade: A arquitetura MVC permite a adição de novos recursos e funcionalidades sem afetar o resto do aplicativo.

- Design escalável: O MVC é adequado para projetos grandes e complexos, pois permite a escalabilidade do design e a distribuição do trabalho em equipe.

O MVC possui três componentes principais, sendo:

1. **Model:** é a camada responsável pela representação dos dados e regras de negócio do sistema. É aqui que ocorre o processamento e validação dos dados.
2. **View:** é a camada responsável pela apresentação dos dados ao usuário final. Ela representa a interface gráfica do sistema, como telas, botões e campos de entrada.
3. **Controller:** é a camada responsável por controlar a interação entre a camada Model e a camada View. Ele recebe as entradas do usuário, chama as funcionalidades correspondentes na camada Model e atualiza a camada View com as alterações necessárias.

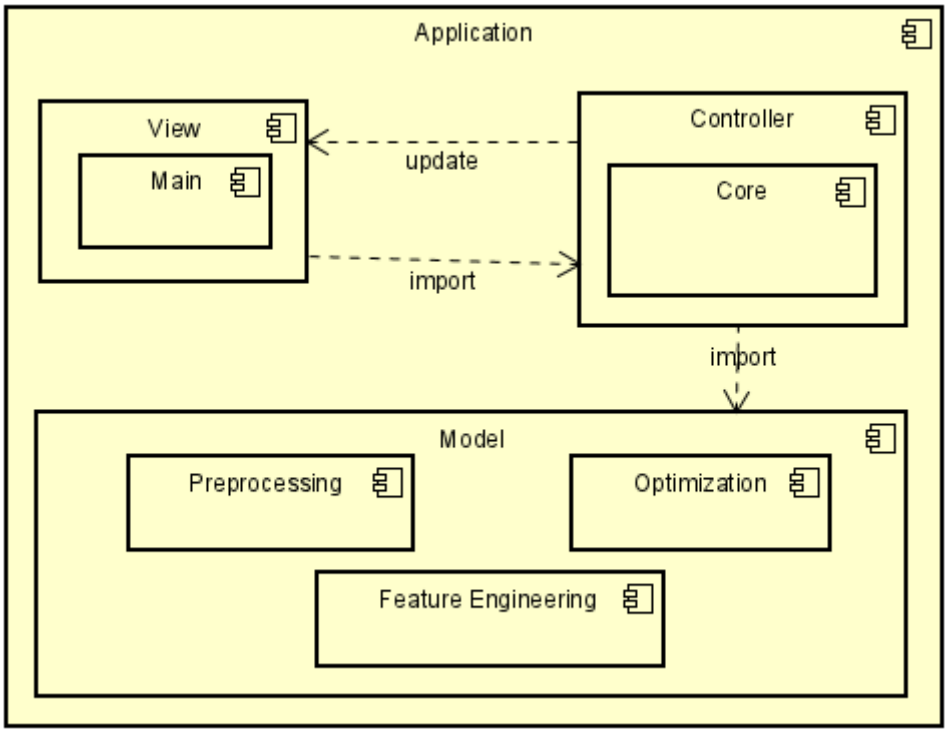


Figura 2. Arquitetura da DroidAutoML.

5.2 Estrutura de diretórios.

A tabela 2 detalha a estrutura de diretórios utilizada pela ferramenta. A estrutura é bastante simples e intuitiva. Como mencionado anteriormente, podemos notar três principais pacotes denominados (Model, View, Controller).

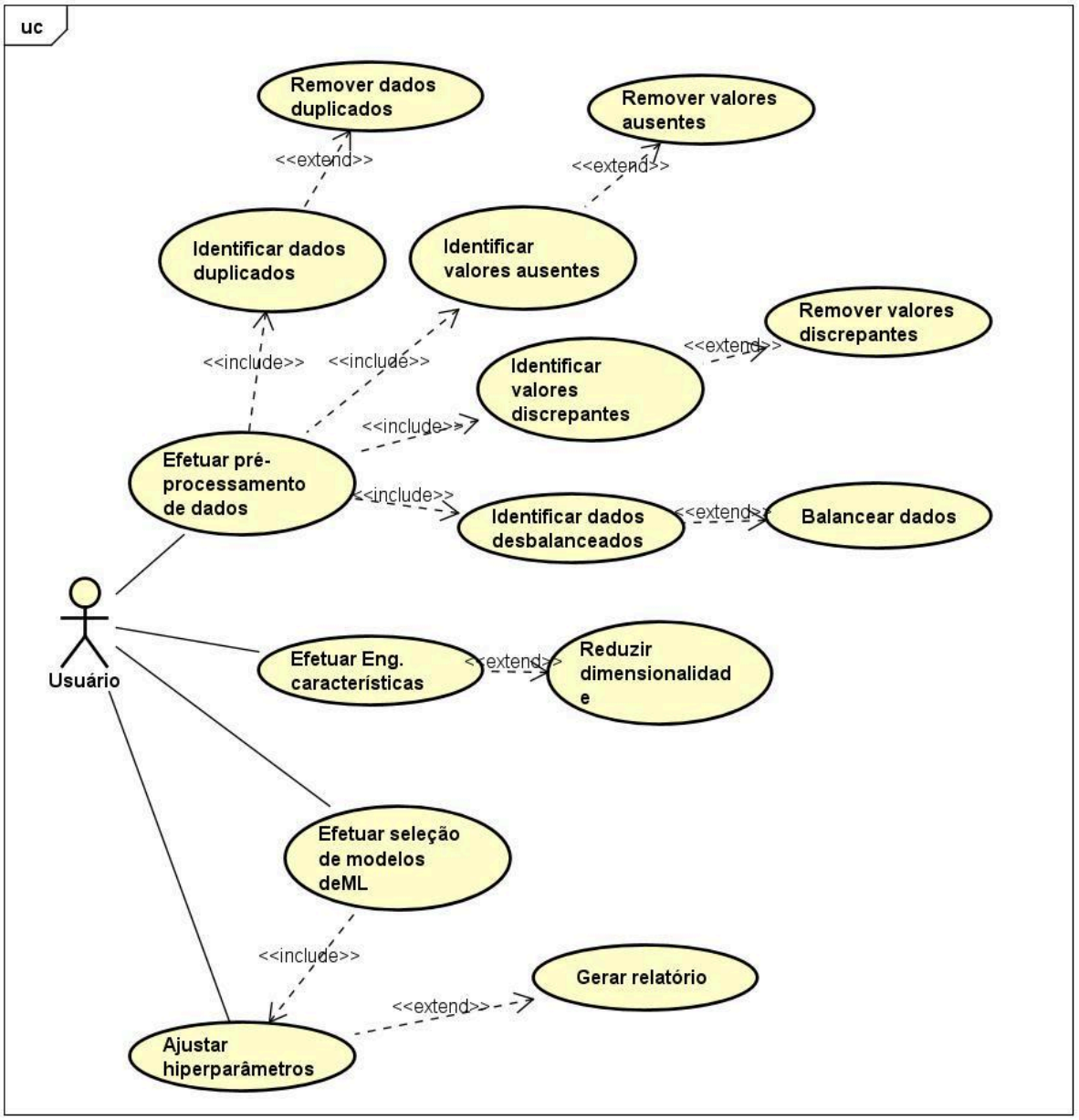
Tabela 2. Estrutura de diretórios utilizada pela ferramenta.

model	Pacote de modelo que representa as etapas do pipeline da ferramenta.
preprocessing	Pacote contendo as classes responsáveis pelo pré-processamento dos dados.
data_cleaning.py	Classe responsável pela limpeza dos dados.
data_info.py	Classe responsável pela coleta de informações dos dados.
data_transformation.py	Classe responsável pela transformação dos dados.
feature_engineering	Pacote contendo as classes especializadas em engenharia de características.
sigpid.py	Classe especializada em características de permissões.
rfg.py	Classe especializada em características de chamadas de APIs.
jowmdroid.py	Classe especializada em múltiplas características.
optimization	Pacote contendo as classes que aplicam a otimização dos modelos de machine learning.
hyperparameters_methods.py	Classe que implementa modelos de machine learning e otimiza seus hiperparâmetros.
view	Pacote contendo as classes que representam a interação com o usuário.
main.py	Classe que interage com o usuário.
controller	Pacote contendo as classes que controlam o sistema.
core.py	Classe que controla as demais classes.

6 Diagramas do Sistema.

6.1 Diagrama de caso de uso

Escrevendo….



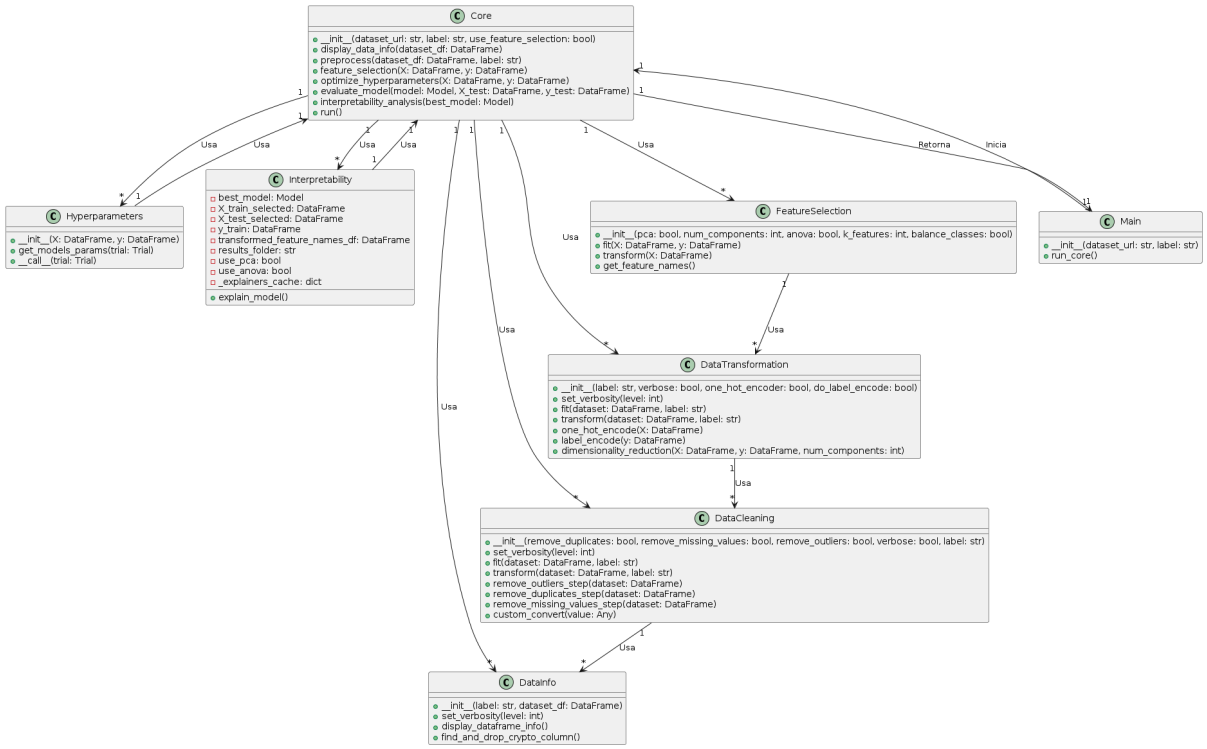


Figura 4 Diagrama de classes MH-AutoML.

6.2 Descrição das Classes

DataCleaning: Classe que representa o processo de limpeza de dados dentro do pacote model/preprocessing. Essa classe possui os métodos:

- **remove_missing_values:** método que recebe como parâmetro um dataset e retorna o dataset sem valores nulos ou faltantes. Esse método é usado para remover todas as linhas com valores *NULL* ou NaN do dataset por meio da função *dropna()* da biblioteca Pandas.
- **remove_duplicates:** método que recebe como parâmetro um dataset e retorna o dataset sem valores duplicados. Esse método é usado para remover as linhas duplicadas, ou seja, cujo valor seja exatamente o mesmo por meio da função *drop_duplicates()* da biblioteca pandas.
- **remove_outliers:** método que recebe como parâmetro um dataset e retorna o dataset sem valores anormais. Um outlier é um valor que foge da normalidade e que pode causar anomalias nos resultados obtidos. Para tratar os outlier utilizamos coeficiente de correlação de Matthews (MCC). O MCC é um coeficiente de correlação projetado para trabalhar com dados binários e é definido como:
  MCC = (TP x TN - FP x FN) / sqrt((TP + FP) (TP + FN) (TN + FP) (TN + FN))
  Onde TP (True Positives) é o número de casos positivos corretamente identificados, TN (True Negatives) é o número de casos negativos corretamente identificados, FP (False Positives) é o número de casos negativos incorretamente identificados como positivos, e FN (False Negatives) é o número de casos positivos incorretamente identificados como negativos. O MCC varia de -1 a 1, onde -1 indica uma correlação inversa perfeita, 0 indica nenhuma correlação e 1 indica uma correlação perfeita.

- **run_data_cleaner:** método chamado para executar os métodos (remove_missing_values, remove_duplicates e remove_outliers)

DataInfo: Classe que apresenta informações básicas de um determinado conjunto de dados ela pertence ao pacote model/preprocessing. Essa classe possui os métodos:

- **display_dataframe_info:** método que apresenta para o usuário informações quanto ao tamanho do conjunto de dados com relação ao número de (features e samples), se o conjunto de dados possuir valores duplicados, se o conjunto de dados está desbalanceado e qual a proporção, se possui valores nulos e faltantes e os tipos de dados que compõem o dataset (e.g., int,float,object,string).
- **is_balanced_dataset:** método que verifica se o dataset está balanceado retornando verdadeiro ou falso caso os dados não estejam balanceados. Em resumo, esse método verifica se um conjunto de dados é balanceado comparando o número de exemplos da classe minoritária com o número de exemplos da classe majoritária e retornando True se a proporção entre eles for maior ou igual a 0,5.

DataTransformation: Classe que trabalha na transformação de dados que é o processo de mudança de dados de um formato para outro. Ela pertence ao pacote model/preprocessing. Essa classe possui os métodos:

- **one_hot_encoder:** método que codifica variáveis categóricas em uma matriz numérica única, recebe uma string label que representa uma coluna de classificação no conjunto de dados e um dataset retornando os valores tratados de X e y. Esse método implementa as funções *LabelEncoder* que codifica rótulos de destino com valor entre 0 e n_classes -1 e a função *OneHotEncoder* que codifica características categóricas em numérica.
- **standardize_data:** método que realiza a padronização dos dados em uma variável denominada "self.dataset" usando a técnica de normalização Z-score. A padronização é feita utilizando a biblioteca Scikit-learn e o objeto *StandardScaler*. O objetivo da padronização é transformar os dados para terem média zero e desvio padrão igual a 1, o que é uma técnica comum de pré-processamento de dados que ajuda a melhorar o desempenho de muitos algoritmos de aprendizado de máquina. A função retorna a variável "self.dataset" com os dados padronizados.
- **balancing_dataset:** o método começa contando o número de instâncias em cada classe no conjunto de dados. A partir desses contadores, é identificada a classe minoritária e a classe majoritária. A função então seleciona todas as instâncias da classe minoritária e um subconjunto aleatório das instâncias da classe majoritária, de forma que o número de instâncias da classe minoritária seja igual ao número de instâncias da classe majoritária. Essa seleção aleatória é realizada usando o método "sample()" do Pandas. Por fim, retorna um novo conjunto de dados que contém as instâncias da classe minoritária e o subconjunto aleatório das instâncias da classe majoritária.

Hyperparameters: Esta classe é usada para ajustar os hiperparâmetros de modelos de aprendizado de máquina usando a biblioteca Optuna. Possui um método chamado " get_models_params " que cria um classificador de votação e o método " call " que realiza validação cruzada nos dados para obter uma pontuação, que é então retornada.

- **get_models_params:** método utilizado para selecionar e instanciar um classificador de aprendizado de máquina a partir de uma lista de opções de classificadores,

incluindo "RandomForestClassifier", "DecisionTreeClassifier", "ExtraTreesClassifier" e "SVC". A seleção do classificador é feita usando a sugestão "trial.suggest da biblioteca Optuna. Finalmente, o código instancia um classificador de votação "VotingClassifier" usando o classificador selecionado e retorna a instância do classificador de votação e retorna um ranking dos modelos e seus hiperparâmetros ajustados.

- **__call__:** método que realiza validação cruzada "cross_val_score" nos dados para obter uma pontuação, que é então retornada.

SigPID: Escrever….
RFG: Escrever….
jowmdroid: Escrever….

Core: Esta é uma classe controladora que pertence ao pacote *controller* tem o objetivo de executar as etapas de pré-processamento, seleção de recursos, otimização de hiperparâmetros e avaliação do modelo de aprendizado de máquina.

- **preprocess:** esse método é responsável pelo controle do pré-processamento, onde a biblioteca Pandas é usada para carregar os dados em um *DataFrame* e executar a limpeza dos dados e padronização dos dados, como remover duplicatas e valores ausentes. Também é verificado se o conjunto de dados está balanceado e se há linhas categóricas, e essas informações são usadas para executar a transformação dos dados, como a codificação one-hot.
- **feature_selection:** A etapa de seleção de recursos, atualmente não esta integrada com os métodos SigPID, RFG e jowmdroid.
- **optimize_hyperparameters:** o método utiliza a biblioteca Optuna para executar a otimização dos hiperparâmetros do modelo de aprendizado de máquina.
- **evaluate_model:** esse método aplica a avaliação do modelo, por meio de métricas como acurácia, precisão, recall, f1-score e relatório de classificação(ranking) para avaliar o modelo de aprendizado de máquina.
- **run:** método executor.

Main: a classe pertence ao pacote de visualização (view) que interage com o usuário onde recebe dois parâmetros de entrada: dataset_url e label.
O método **run_core** instancia um objeto Core com esses parâmetros e chama o método run dessa classe.

O script utiliza o módulo argparse para fazer o parsing de argumentos da linha de comando. Ele define dois argumentos obrigatórios (--dataset e --label) que serão usados como entrada para a instância da classe Main.

Ao executar o script diretamente, ou seja, quando __name__ == "__main__", o módulo argparse é acionado para obter os argumentos da linha de comando, que são usados para criar uma instância de Main e chamar o método run_core.

6.3 Diagrama de sequência execução padrão

Escrever….

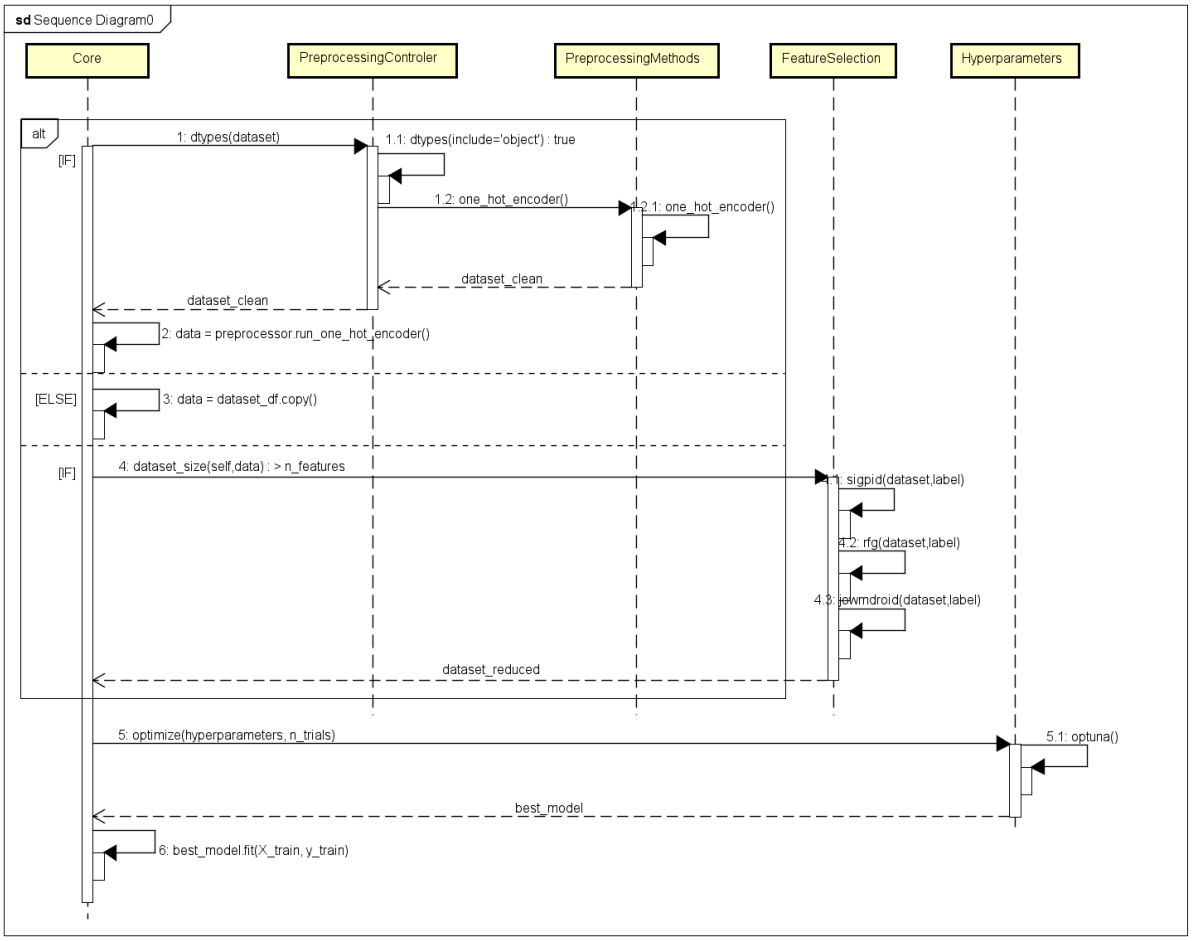


Figura 5. Diagrama de sequência execução padrão DroidAutoML.

7 Implementação.

A implementação da ferramenta foi realizada utilizando a linguagem de programação Python na versão 3.9.7 e bibliotecas de código aberto amplamente utilizadas em modelos de análise de dados e aprendizado de máquina. Na tabela 3, é possível verificar as dependências da ferramenta, juntamente com suas respectivas versões e objetivos específicos. Para uma visão mais completa, as classes e métodos utilizados no sistema podem ser encontrados no anexo deste relatório.

Tabela 3. bibliotecas utilizadas na implementação da ferramenta DroidAutoML.

Pacote	Versão	Descrição

pandas	1.3.3	biblioteca para análise e manipulação de dados.
numpy	1.20.1	biblioteca de computação científica que fornece suporte para arrays e matrizes multidimensionais, além de funções matemáticas para operações nesses objetos.
scipy	1.7.1	biblioteca que oferece diversas funções para resolução de problemas matemáticos, estatísticos e científicos em geral.
tqdm	4.62.3	biblioteca que oferece uma barra de progresso para iterações em loops de Python, útil para monitorar o progresso de operações que podem levar algum tempo.
joblib	1.0.1	biblioteca que oferece ferramentas para execução de funções paralelamente em diferentes processadores, útil para acelerar a execução de operações intensivas.
threadpoolctl	2.2.0	biblioteca que oferece ferramentas para controle de threads em pools de execução, permitindo que as threads sejam compartilhadas entre diferentes processos.
scikit-learn	1.1.1	biblioteca de aprendizado de máquina para tarefas de classificação, regressão e clustering.
optuna	2.10.1	biblioteca de otimização de hiperparâmetros para aprendizado de máquina
logging	0.4.9.6	biblioteca para geração de logs em um aplicativo
argparse	1.4.0	biblioteca para criação de interfaces de linha de comando, que permite definir facilmente argumentos e opções para um programa.
termcolor	1.1.0	biblioteca que permite imprimir texto colorido no terminal para destacar informações importantes.
halo	0.0.31	biblioteca para criar indicadores de atividade e progresso em um aplicativo.
mlxtend	0.20.0	biblioteca de aprendizado de máquina que oferece ferramentas para análise, visualização e construção de modelos, incluindo implementações de algoritmos comuns e ferramentas de avaliação e visualização de modelos e dados.
spinners	0.0.24	biblioteca que oferece animações de carregamento para programas que necessitam de indicações visuais de progresso.

7.1 Backlog

A DroidAutoML utilizou-se da técnica de desenvolvimento guiado por backlog, também conhecida como backlog-driven development, é uma metodologia de desenvolvimento ágil que se concentra na criação e gerenciamento de um backlog de tarefas priorizadas e bem definidas para orientar o processo de desenvolvimento.

O backlog é uma lista de itens que precisam ser implementados no projeto, como funcionalidades, correções de bugs e melhorias. Essa lista é mantida e gerenciada pelo product owner, que é responsável por definir as prioridades e requisitos de cada item. O time de desenvolvimento trabalha de acordo com a ordem de prioridade definida no backlog, sempre se concentrando em concluir os itens mais importantes primeiro.

A técnica de desenvolvimento guiado por backlog é baseada em ciclos iterativos de desenvolvimento, geralmente de duas a quatro semanas de duração, chamados sprints. No início de cada sprint, o time de desenvolvimento seleciona os itens mais importantes do backlog que serão implementados durante o sprint. Durante o sprint, o time de desenvolvimento trabalha para implementar esses itens e garantir que eles atendam aos requisitos definidos pelo product owner. Ao final do sprint, o time de desenvolvimento realiza uma revisão e retrospectiva para avaliar o que foi concluído e definir as próximas prioridades do backlog.

7.2 Prioridades do backlog

Para o desenvolvimento foram adotadas as seguintes denominações:

- **Essencial:** tarefa sem o qual o sistema não funcionará. Essas tarefas são essenciais e devem estar no topo da prioridade de implementação.
- **Importante:** tarefa sem o qual o sistema funcionará, porém de forma não satisfatória. Tarefas importantes devem ser implementadas, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável:** tarefas que complementam o sistema, porém não são de suma importância para o funcionamento do mesmo. Devem ser implementados caso não prejudiquem no tempo de implementação de requisitos que possuam prioridade superior.

Tabela 1. Backlog do projeto

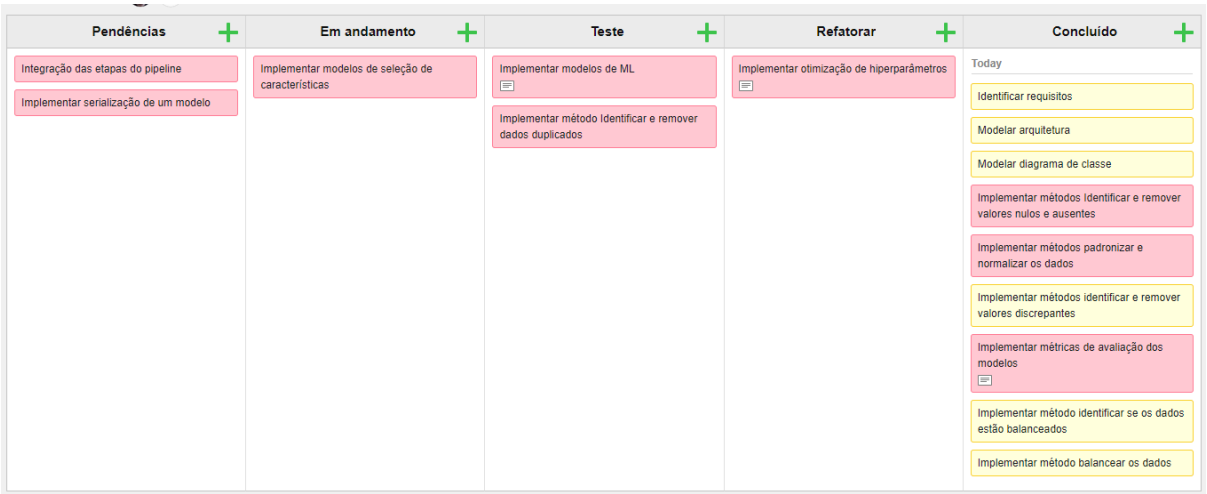
Backlog DroidAutoML	
Pré-processamento	
Identificar e remover dados duplicados	Importante
Identificar e remover valores nulos e ausentes	Essencial
Identificar e remover valores discrepantes	Importante
Padronizar e normalizar os dados	Essencial
Identificar se os dados estão balanceados	Importante

Balancear os dados	Importante
Engenharia de características	
ANOVA	Essencial
PCA	Essencial
LASSO	Essencial
	Desejável
Outros	Desejável
Modelos ML	
RandomForestClassifier	Essencial
DecisionTreeClassifier	Essencial
ExtraTreesClassifier	Essencial
KNNClassifier	Essencial
LightGBMClassifier	Essencial
CatBoostClassifier	Essencial
VotingClassifier	Essencial
	Importante
Otimização de Hiperparâmetros	
Optuna	Essencial
GridSearchCV	Desejável
RandomizedSearchCV	Desejável
BayesSearchCV	Desejável
HalvingGridSearchCV	Desejável
HalvingRandomSearchCV	Desejável
Avaliação dos modelos	
Acurácia	Essencial
Precisão	Importante
Recall	Essencial
F1-Score	Importante
AUC-ROC	Desejável
RMSE	Desejável
MAE	Desejável
Serialização de um modelo	Essencial

7.3. Gestão de fluxo de trabalho

O fluxo de trabalho é gerenciado pela ferramenta kanbanflow. O Kanban é amplamente utilizado para definir, gerenciar e melhorar serviços que envolvem o trabalho do

conhecimento. Esse método permite a visualização do trabalho em quadros Kanban, o que ajuda a maximizar a eficiência e aprimorar continuamente o processo. Com o Kanban, é possível otimizar a entrega de trabalho em várias equipes e lidar com projetos complexos em um único ambiente. Assim também possibilitando analisar o status report das atividades.



7.4.Codificação e controle de versão

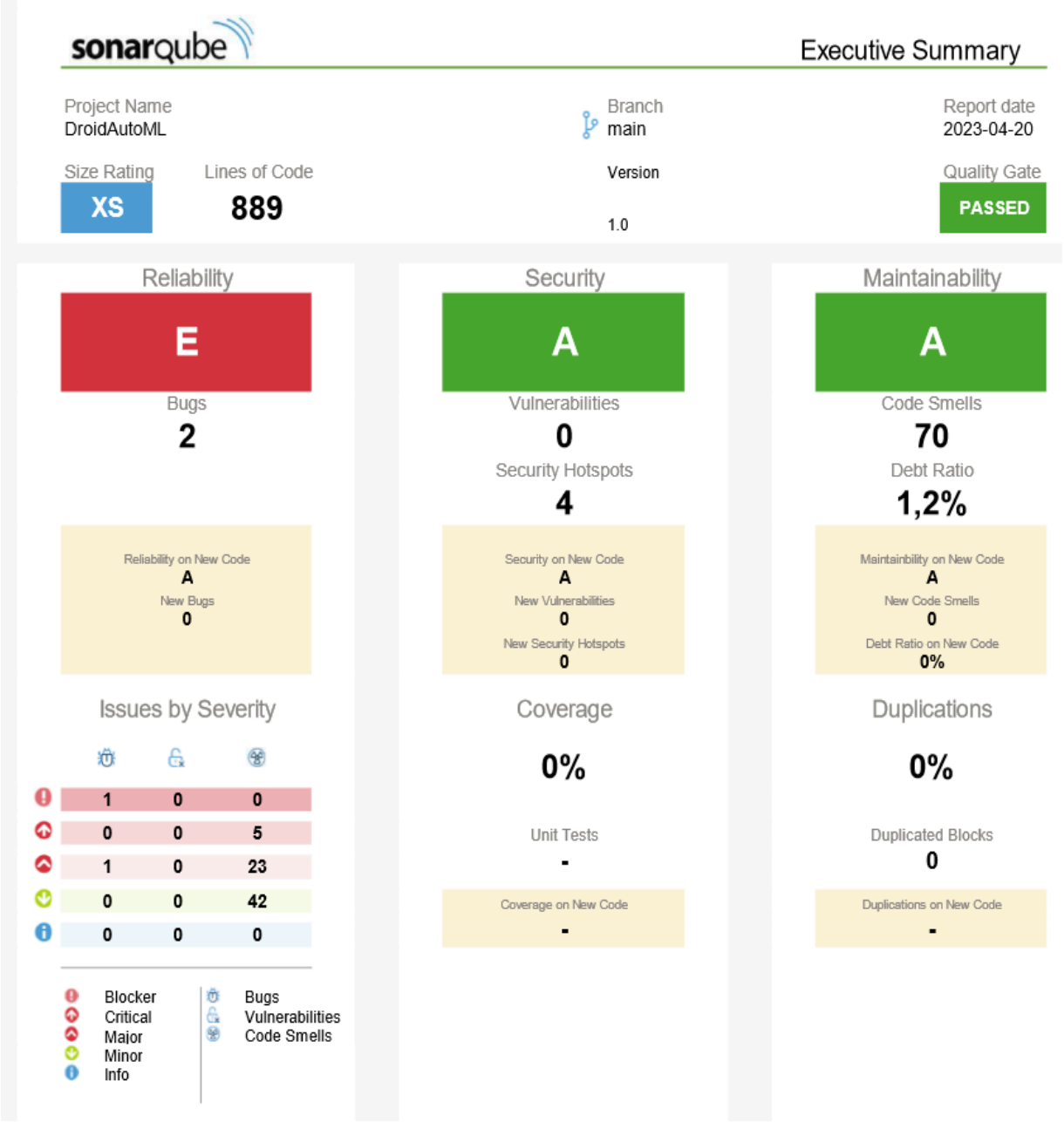
Escreverndo…..
O controle de versão é uma prática fundamental para o desenvolvimento de software, e o Git é uma das ferramentas mais populares para o controle de versão de código fonte. Com o Git, desenvolvedores podem gerenciar diversas versões do mesmo projeto e manter um histórico completo das alterações realizadas ao longo do tempo. Essa ferramenta permite que múltiplos colaboradores trabalhem simultaneamente em um mesmo projeto, sem conflitos de código ou problemas de integração.

Uma das principais vantagens do Git é a sua capacidade de criar ramificações (branches) de um projeto, permitindo que diferentes equipes ou desenvolvedores trabalhem em recursos ou funcionalidades distintas sem afetar o código principal. Além disso, o Git oferece um sistema de revisão de código (code review), no qual os desenvolvedores podem revisar e comentar as alterações realizadas por outros membros da equipe antes de integrá-las ao código principal. Isso ajuda a garantir a qualidade do código e a reduzir erros e bugs.

7.5 Qualidade do código

Para o gerenciamento contínuo da qualidade do código foi utilizado a ferramenta SonarQube que é uma ferramenta popular de controle de qualidade de código que ajuda a garantir que o código-fonte de um projeto atenda aos padrões de qualidade e segurança. Ele realiza uma análise estática do código-fonte para identificar possíveis problemas, como vulnerabilidades de segurança, bugs, duplicações de código, complexidade excessiva e outras questões de manutenção de código.

Além disso, o SonarQube fornece métricas úteis para avaliar a qualidade do código, como cobertura de testes, complexidade ciclomática, dívida técnica e outras. Essas métricas ajudam os desenvolvedores e gerentes de projeto a entender a qualidade do código e a priorizar a correção de problemas críticos. A integração do SonarQube com as ferramentas de integração contínua também ajuda a garantir que o código seja avaliado automaticamente a cada push, proporcionando feedback imediato sobre a qualidade do código.



	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
DroidAutoML							
└ controller	100	0	0	7	0	0.0%	0.0%
└ model	757	2	0	62	4	0.0%	0.0%
└ view	16	0	0	1	0	0.0%	0.0%
└ setup.py	16	0	0	0	0	0.0%	0.0%

Project Size

Lines of code	889
Files	17
Functions	59
Statements	728

Issues Breakdown

Severity	Rule	Type	Language	Issues
BLOCKER	The number and name of arguments passed to a function should match its parameters	BUG	Python	1
CRITICAL	Cognitive Complexity of functions should not be too high	CODE_SMELL	Python	4
CRITICAL	String literals should not be duplicated	CODE_SMELL	Python	1
MAJOR	Sections of code should not be commented out	CODE_SMELL	Python	13

MAJOR	Function names should comply with a naming convention	CODE_SMELL	Python	6
MAJOR	Functions, methods and lambdas should not have too many parameters	CODE_SMELL	Python	1
MAJOR	Builtins should not be shadowed by local variables	CODE_SMELL	Python	1
MAJOR	Unused function parameters should be removed	CODE_SMELL	Python	1
MAJOR	Unused assignments should be removed	CODE_SMELL	Python	1
MAJOR	All code should be reachable	BUG	Python	1
MINOR	Local variable and function parameter names should comply with a naming convention	CODE_SMELL	Python	36
MINOR	Unused local variables should be removed	CODE_SMELL	Python	6

8 Testes.

9 instalação.

10 Anexo.

Análise preliminar do código das classes e métodos