FUNDAÇÃO ESCOLA DE COMÉRCIO ÁLVARES PENTEADO – FECAP

CIÊNCIA DA COMPUTAÇÃO

**GUSTAVO MARCELLO CORREA DE ARAUJO - 23024729 LUCCA GIORDANO - 23024522**

**PEDRO HENRIQUE DANGELO DOS REIS - 23024777 VITOR UTIMURA LOCATELI - 23024638**

**Sistemas Operacionais e Computação em Nuvem: Entrega 2**

**São Paulo 2025**

**GUSTAVO MARCELLO CORREA DE ARAUJO - 23024729 LUCCA GIORDANO - 23024522**

**PEDRO HENRIQUE DANGELO DOS REIS - 23024777 VITOR UTIMURA LOCATELI - 23024638**

**Sistemas Operacionais e Computação em Nuvem: Entrega 2**

Relatório Técnico apresentado ao curso de Ciência da Computação, como parte dos re- quisitos da disciplina de Sistemas Operacionais e Computação em Nuvem, referente ao Projeto Interdisciplinar.

Orientador: Rodrigo da Rosa

São Paulo 2025

SUMÁRIO

[INTRODUÇÃO 4](#_Toc192250242)

[OBJETIVO 5](#_Toc192250243)

[MÉTODOS 6](#_Toc192250244)

[DESENVOLVIMENTO 7](#_Toc192250245)

[CONCLUSÃO 8](#_Toc192250246)

[REFERÊNCIAS 9](#_Toc192250247)

[ANEXOS 10](#_Toc192250248)

INTRODUÇÃO

Este relatório detalha a implementação de um sistema que combina duas aplicações de Inteligência Artificial (IA) e Aprendizado de Máquina (ML). O primeiro modelo é de Classificação Supervisionada, implementado através de um notebook Jupyter (Entrega.ipynb), que utiliza dados de alunos para prever a probabilidade de aprovação. O segundo sistema é de Monitoramento de Recursos com Detecção de Anomalias, que coleta métricas de desempenho (CPU, memória, disco) de uma máquina virtual (VM) Ubuntu Server durante a execução do notebook de classificação e aplica um modelo de IA não supervisionado (Isolation Forest) para identificar padrões de uso anômalos. O objetivo principal é atender aos requisitos da disciplina, demonstrando a aplicação prática de conceitos de IA/ML em um cenário real, envolvendo tanto aprendizado supervisionado quanto não supervisionado, além de técnicas de monitoramento e análise de desempenho.

OBJETIVO

O objetivo principal da atividade é implementar um sistema que combine dois tipos de Inteligência Artificial (IA) e Aprendizado de Máquina (ML). O primeiro componente envolve um modelo de classificação supervisionada, que utiliza dados de alunos para prever a probabilidade de aprovação. O segundo componente é um sistema de monitoramento de recursos que coleta métricas de desempenho de uma máquina virtual durante a execução do modelo de ML e identifica anomalias nos padrões de uso dos recursos computacionais. O projeto busca aplicar conceitos de IA e ML, tanto supervisionado quanto não supervisionado, além de técnicas de monitoramento e análise de desempenho, atendendo aos requisitos da disciplina de Sistemas Operacionais.

MÉTODOS

Os métodos utilizados no projeto incluem a implementação de um modelo de classificação supervisionada e a criação de um sistema de monitoramento de recursos com detecção de anomalias.

Para o modelo de classificação supervisionada, foram utilizados dados de alunos para prever a probabilidade de aprovação, com base em características demográficas e de desempenho. O pré-processamento envolveu a normalização das variáveis numéricas utilizando o StandardScaler do scikit-learn para garantir que todas as features tivessem média 0 e desvio padrão 1. Os dados foram divididos em conjuntos de treino (70%) e teste (30%) usando a função train\_test\_split. Foram aplicados quatro modelos de classificação: Regressão Logística, K-Nearest Neighbors (KNN), Árvore de Decisão e Random Forest. Para avaliar os modelos, foram utilizadas métricas como acurácia, relatório de classificação (que inclui precisão, recall e F1-score) e matriz de confusão, além de validação cruzada com 5 folds.

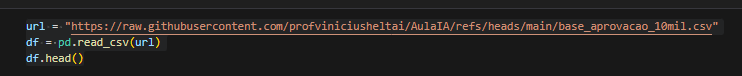
O sistema de monitoramento foi criado para coletar métricas de desempenho da máquina virtual durante a execução do notebook de ML. Utilizando a biblioteca psutil, o script monitor.py coleta dados sobre a utilização da CPU, memória RAM e disco da máquina. As métricas são registradas em um arquivo CSV, e o script run\_monitor\_loop.py executa o monitoramento a cada 5 segundos, enquanto o notebook é executado em outro terminal. A execução do notebook foi automatizada usando jupyter nbconvert, permitindo que todas as células do notebook fossem executadas de forma não interativa. O modelo de detecção de anomalias foi implementado utilizando Isolation Forest do scikit-learn, um algoritmo de aprendizado não supervisionado, para identificar padrões de uso anômalo nas métricas de CPU e memória durante a execução do notebook.

A avaliação das anomalias foi feita visualmente por meio de gráficos gerados com matplotlib, destacando os pontos identificados como anomalias. O parâmetro contamination do Isolation Forest foi ajustado para estimar a proporção esperada de anomalias nos dados.

DESENVOLVIMENTO

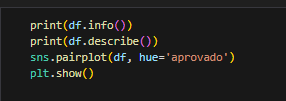
**Entrega.ipynb**

1.

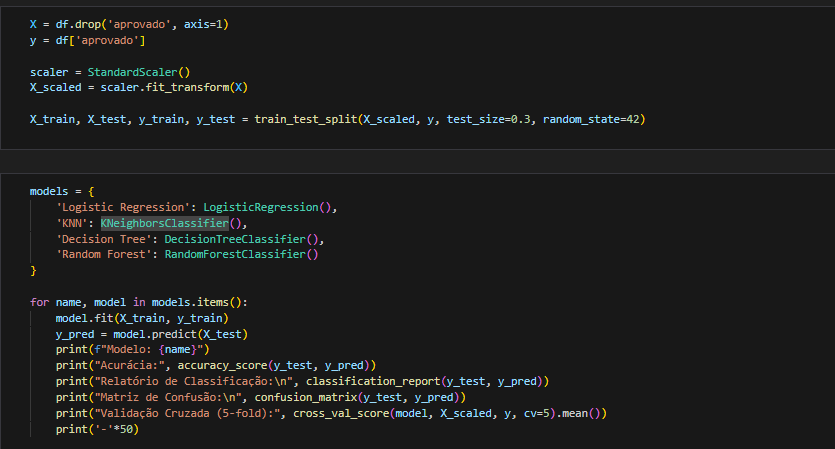


Aqui, o código está carregando um arquivo CSV diretamente da web, utilizandoa URL fornecida. O método **pd.read\_csv(url)** da biblioteca Pandas é responsável por ler o arquivo CSV e armazená-lo em um **DataFrame** chamado df. O DataFrame é uma estrutura de dados tabular, semelhante a uma planilha, que armazena dados de forma organizada.

2.

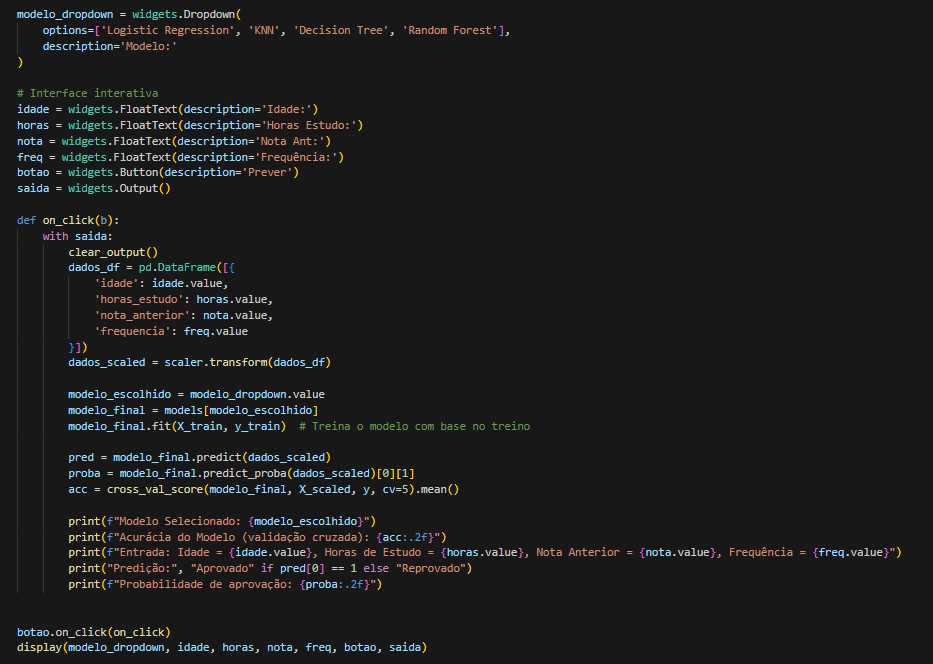


exibe informações sobre o DataFrame com **df.info()** (número de colunas, tipo de dados, valores não nulos) e estatísticas descritivas com **df.describe()** (média, desvio padrão, valores mínimo e máximo).

3.

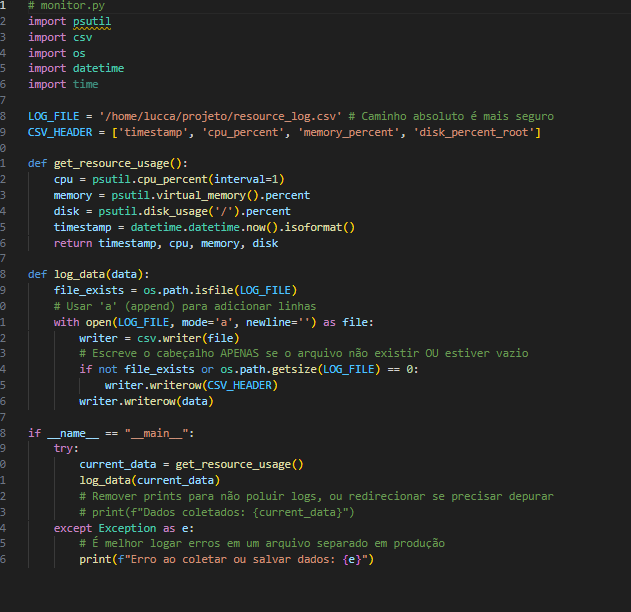
O código começa separando os dados em variáveis preditoras X e a variável alvo y (aprovado). Em seguida, os dados são normalizados usando StandardScaler para garantir que todas as variáveis tenham a mesma escala. Depois, os dados são divididos em conjuntos de treino e teste.

Quatro modelos de aprendizado de máquina são definidos: Regressão Logística, KNN, Árvore de Decisão e Random Forest. Para cada modelo, o código treina com os dados de treino e faz previsões com os dados de teste. O desempenho de cada modelo é avaliado em termos de acurácia, relatório de classificação, matriz de confusão e validação cruzada. Isso permite comparar a performance de cada modelo.

4.

O código cria uma interface onde o usuário pode escolher um modelo de aprendizado de máquina (como Regressão Logística ou KNN), inserir dados como idade, horas de estudo, nota anterior e frequência, e clicar no botão "Prever" para obter uma previsão. Quando o botão é clicado, os dados inseridos são usados para fazer uma previsão de aprovação usando o modelo escolhido. O modelo é treinado com os dados de treino e a acurácia é calculada. Em seguida, o código exibe o modelo selecionado, a acurácia, os dados inseridos e a previsão de aprovação, além da probabilidade de aprovação.

**monitor.py**

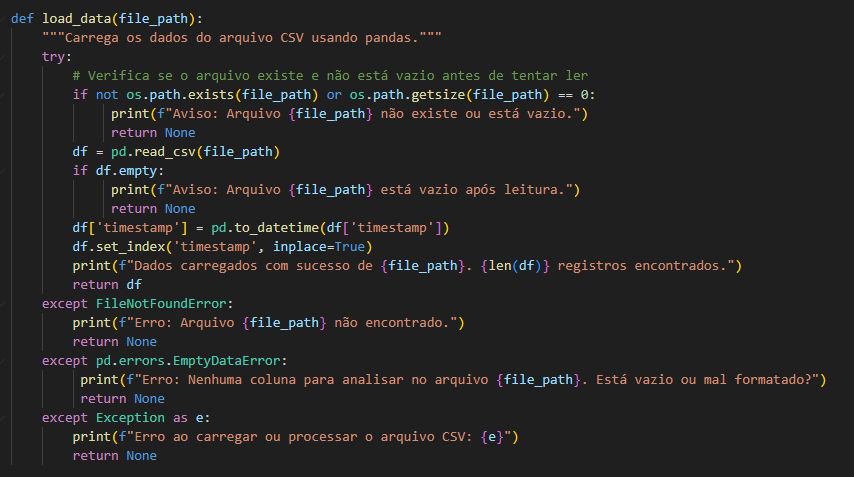
1.

O código coleta informações sobre o uso de recursos do sistema, como CPU, memória e disco, e grava esses dados em um arquivo CSV. Ele usa a biblioteca psutil para obter as informações de uso de CPU, memória e disco, e a biblioteca csv para escrever os dados no arquivo. A função get\_resource\_usage() coleta o uso da CPU, memória e disco, e a data e hora atual. Em seguida, a função log\_data() verifica se o arquivo de log já existe e, se necessário, escreve o cabeçalho e os dados coletados no arquivo CSV. O bloco principal chama essas funções para coletar e registrar os dados, tratando erros com um bloco try-except para garantir que qualquer falha seja registrada de forma apropriada.

**Analyze.py**

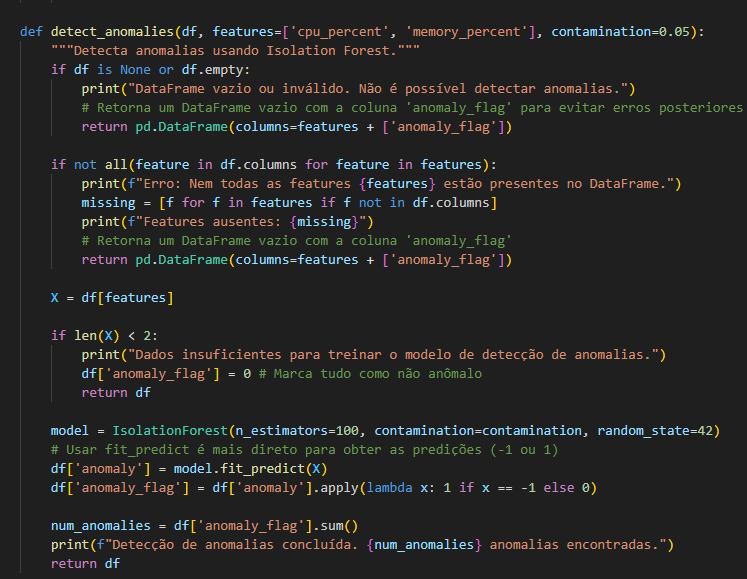
1.

A função load\_data() é responsável por carregar os dados de um arquivo CSV usando a biblioteca Pandas. Ela começa verificando se o arquivo existe e se não está vazio. Caso contrário, retorna None. Se o arquivo for válido, ele é lido com pd.read\_csv(), e a coluna timestamp é convertida para o formato datetime usando pd.to\_datetime(). Essa coluna é então definida como o índice do DataFrame. Caso o arquivo tenha algum erro, como estar vazio ou mal formatado, a função captura essas exceções e retorna mensagens de erro apropriadas.



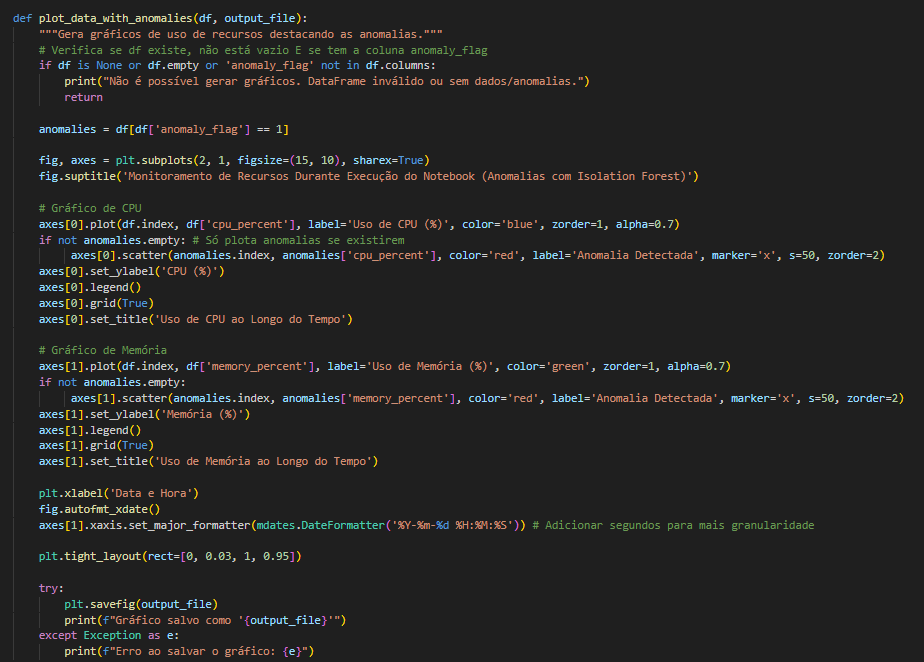
2.

A função detect\_anomalies() usa o modelo **Isolation Forest** para detectar anomalias nos dados. Ela começa verificando se o DataFrame é válido e se as colunas especificadas (como cpu\_percent e memory\_percent) estão presentes. Se algum dado estiver ausente ou inválido, a função retorna um DataFrame vazio. Quando os dados estão prontos, a função cria um modelo **Isolation Forest** e o treina com as features selecionadas. As anomalias são então detectadas, onde a função marca as linhas como anômalas com 1 e as normais com 0, e retorna o DataFrame atualizado com a nova coluna anomaly\_flag.

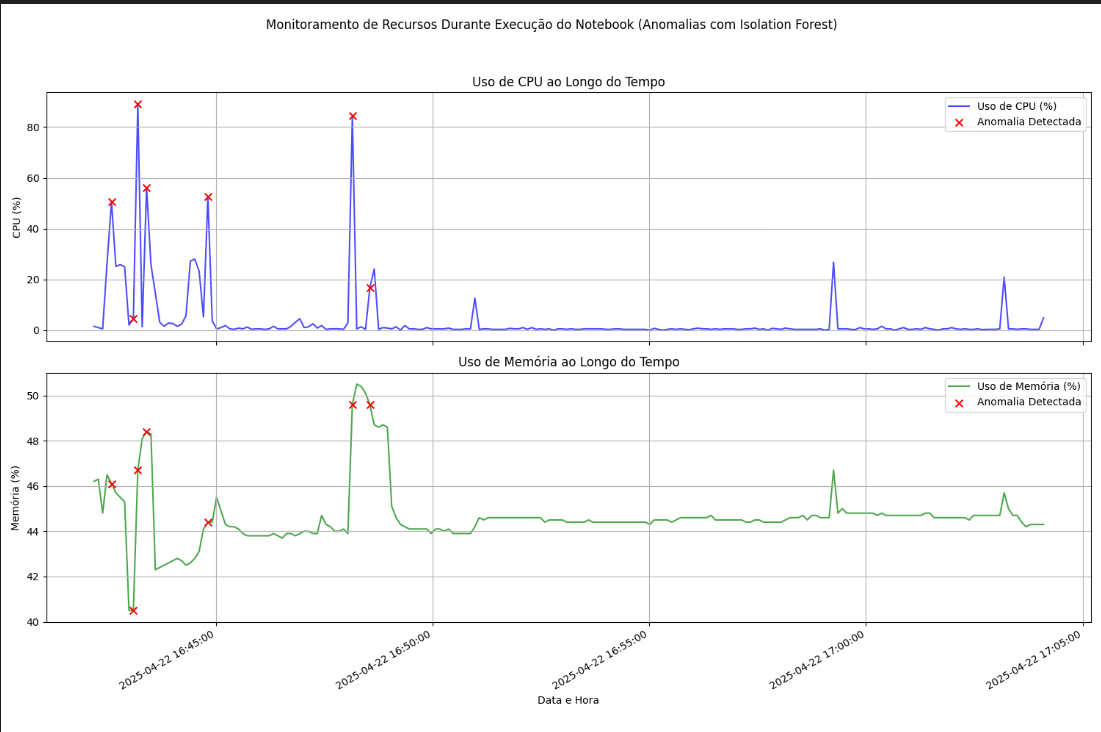


3.

A função plot\_data\_with\_anomalies() gera gráficos para visualizar o uso de CPU e memória ao longo do tempo, destacando as anomalias detectadas. Ela começa verificando se o DataFrame é válido e se a coluna anomaly\_flag está presente. Em seguida, cria dois gráficos: um para o uso de **CPU** e outro para o uso de **memória**. Se existirem anomalias, essas são destacadas nos gráficos com pontos vermelhos. O gráfico é salvo como uma imagem PNG, e a função também garante que o eixo de tempo seja formatado corretamente com a data e hora.



**Gráfico de resultado**

****

O gráfico mostra o monitoramento do uso de CPU e memória ao longo do tempo durante a execução de um notebook. A linha azul representa o uso de CPU, e a linha verde mostra o uso de memória. As marcações vermelhas, representadas por 'X', indicam as anomalias detectadas pelo algoritmo **Isolation Forest**. É possível observar picos de uso de CPU e memória, com algumas flutuações mais intensas que foram identificadas como anômalas. O gráfico ilustra como os recursos do sistema variam durante a execução e destaca os momentos em que o uso de CPU e memória é incomum.

CONCLUSÃO

O projeto integrou com sucesso um modelo de aprendizado supervisionado para classificação com um sistema de monitoramento de recursos baseado em aprendizado não supervisionado para detecção de anomalias. O modelo [Nome do Melhor Modelo] demonstrou alta capacidade preditiva para a aprovação de alunos, conforme avaliado por múltiplas métricas. O sistema de monitoramento conseguiu capturar a dinâmica de uso de recursos da VM durante a execução da tarefa de ML, e o Isolation Forest foi capaz de identificar momentos de comportamento atípico, que [Coincidiram/Não coincidiram] em grande parte com as fases computacionalmente mais intensivas do notebook. Este trabalho ilustra a aplicação prática e complementar de diferentes técnicas de IA