Relatório: Implementação de Modelo de IA para Classificação e Monitoramento de Recursos com Detecção de Anomalias

Data: 22/04/2025

Disciplina: Sistemas Operacionais

1. Introdução

Este relatório detalha a implementação de um sistema que combina duas aplicações de Inteligência Artificial (IA) e Aprendizado de Máquina (ML):

* Modelo de Classificação Supervisionada: Implementado através de um notebook Jupyter (Entrega.ipynb), utiliza dados de alunos para prever a probabilidade de aprovação.
* Sistema de Monitoramento de Recursos com Detecção de Anomalias: Um sistema customizado 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.

Ferramentas Utilizadas:

Linguagem: Python 3

Bibliotecas Principais:

* Para ML (Notebook): pandas, numpy, seaborn, matplotlib, scikit-learn, ipywidgets, jupyter
* Para Monitoramento/Anomalia: psutil, pandas, scikit-learn, matplotlib
* Ambiente: VM Ubuntu Server, Ambiente Virtual Python (venv)
* Automação/Execução: subprocess, Python (para o loop de monitoramento), jupyter nbconvert (para execução do notebook)

2. Implementação do Modelo de IA (Notebook de Classificação)

Objetivo: Prever se um aluno será aprovado (1) ou reprovado (0) com base em características demográficas e de desempenho.

Fonte de Dados: O arquivo base\_aprovacao\_10mil.csv, contendo 10.000 registros de alunos.

Features (Variáveis Preditivas): idade, horas\_estudo, nota\_anterior, frequencia.

Variável Alvo: aprovado (variável binária).

Tipo de Aprendizado: Supervisionado (Classificação).

Pré-processamento:

* Os dados foram carregados usando pandas.
* As features numéricas foram normalizadas utilizando StandardScaler do scikit-learn para garantir que todas as features tivessem média 0 e desvio padrão 1, evitando que features com escalas maiores dominassem o modelo.
* Os dados foram divididos em conjuntos de treino (70%) e teste (30%) usando train\_test\_split.

Modelos de Classificação Testados:

* Regressão Logística (LogisticRegression): Um modelo linear para classificação binária.
* K-Nearest Neighbors (KNeighborsClassifier): Um modelo baseado em instância que classifica com base nos vizinhos mais próximos.
* Árvore de Decisão (DecisionTreeClassifier): Um modelo baseado em regras hierárquicas.
* Random Forest (RandomForestClassifier): Um modelo de ensemble que utiliza múltiplas árvores de decisão para melhorar a robustez e a precisão.

Avaliação dos Modelos:

* Acurácia (accuracy\_score): Percentual de previsões corretas.
* Relatório de Classificação (classification\_report): Inclui Precisão, Recall e F1-Score para cada classe (0 e 1), fornecendo uma visão mais detalhada do desempenho.
* Matriz de Confusão (confusion\_matrix): Mostra os verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.
* Validação Cruzada (cross\_val\_score): Utilizada com 5 folds para estimar a performance do modelo em diferentes subconjuntos dos dados, fornecendo uma medida mais robusta da generalização do modelo.

Interface Interativa (ipywidgets): O notebook inclui uma interface simples para que o usuário possa inserir manualmente os dados de um aluno e obter a predição de aprovação usando o modelo selecionado.

3. Implementação do Sistema de Monitoramento e Análise de Desempenho

Este sistema foi criado para observar o comportamento da VM enquanto o processamento computacionalmente intensivo (execução do notebook de ML) ocorria.

Coleta de Dados (monitor.py e run\_monitor\_loop.py):

O script monitor.py utiliza a biblioteca psutil para acessar informações do sistema operacional.

As métricas coletadas são:

cpu\_percent: Percentual de utilização da CPU.

memory\_percent: Percentual de utilização da memória RAM.

disk\_percent\_root: Percentual de utilização do disco na partição raiz (/).

Um timestamp ISO 8601 é registrado para cada coleta.

Os dados são anexados a um arquivo CSV (resource\_log.csv) a cada execução.

O script run\_monitor\_loop.py foi utilizado para executar monitor.py repetidamente em um intervalo fixo (definido como 5 segundos no script) enquanto o notebook estava sendo executado em outro terminal. Isso garante a coleta contínua de dados durante a carga de trabalho.

Antes de iniciar um novo ciclo de monitoramento, o run\_monitor\_loop.py limpa o arquivo resource\_log.csv para garantir que os dados analisados correspondam apenas à execução atual.

Execução do Notebook (jupyter nbconvert):

O notebook Entrega.ipynb foi executado de forma não interativa usando o comando jupyter nbconvert --execute. Isso garante que todas as células sejam executadas sequencialmente, incluindo o treinamento e avaliação dos modelos de ML.

A saída da execução (incluindo gráficos e prints) foi salva no arquivo Entrega\_executado.ipynb.

Análise de Anomalias com IA (analyze.py):

Objetivo: Identificar momentos durante a execução do notebook em que o uso de recursos da VM se desviou significativamente do padrão usual.

Tipo de Aprendizado: Não Supervisionado (Detecção de Anomalias).

Algoritmo de IA: Foi utilizado o Isolation Forest (sklearn.ensemble.IsolationForest). Este algoritmo é eficaz na detecção de outliers (anomalias) pois funciona isolando observações anômalas, que geralmente requerem menos partições em árvores aleatórias para serem isoladas em comparação com pontos normais. É adequado para este cenário, pois não requer dados rotulados (não precisamos saber previamente o que é uma anomalia).

Features Utilizadas: O modelo de detecção de anomalias foi treinado usando as métricas cpu\_percent e memory\_percent como features de entrada, pois são indicadores comuns de carga do sistema. A métrica disk\_percent\_root foi coletada, mas não utilizada neste modelo específico, representando uma forma de seleção de features.

Parâmetro contamination: Este hiperparâmetro do Isolation Forest estima a proporção esperada de anomalias nos dados. Foi definido um valor [ex: 0.03 ou 'auto' - ajuste conforme usou], indicando que esperamos que cerca de [ex: 3%] dos pontos de dados sejam anômalos. Esse valor pode ser ajustado com base na análise dos resultados ou no conhecimento do domínio.

Visualização: O script utiliza matplotlib para gerar um gráfico de séries temporais (resource\_monitoring\_anomalies.png) mostrando o uso de CPU e Memória ao longo do tempo. Os pontos identificados como anomalias pelo Isolation Forest são destacados no gráfico (geralmente com marcadores vermelhos).

4. Conexão com Conceitos da Disciplina

Este projeto abrange diversos conceitos fundamentais de Inteligência Artificial e Aprendizado de Máquina:

Aprendizado Supervisionado vs. Não Supervisionado: Demonstra claramente a diferença entre os dois paradigmas. O notebook de classificação é um exemplo clássico de aprendizado supervisionado (aprender a partir de dados rotulados - aprovado/reprovado). O monitoramento com Isolation Forest é um exemplo de aprendizado não supervisionado (encontrar padrões/anomalias sem rótulos prévios).

Classificação: O notebook aborda um problema de classificação binária, utilizando e comparando diferentes algoritmos (Regressão Logística, KNN, Árvore de Decisão, Random Forest).

Detecção de Anomalias (Outlier Detection): A aplicação do Isolation Forest no monitoramento é uma técnica específica de IA para identificar dados que não se conformam com o comportamento esperado, essencial para monitoramento de sistemas e detecção de fraudes.

Pré-processamento de Dados: Inclui a normalização de features (StandardScaler) no notebook, crucial para algoritmos sensíveis à escala (como KNN e Regressão Logística), e a coleta/formatação de dados de séries temporais no sistema de monitoramento.

Engenharia e Seleção de Features: A escolha das variáveis preditivas no notebook (idade, horas\_estudo, etc.) e a seleção das métricas (cpu\_percent, memory\_percent) para a detecção de anomalias são exemplos de seleção de features relevantes para cada problema.

Avaliação de Modelos: O notebook utiliza métricas quantitativas (Acurácia, Precisão, Recall, F1-Score, Matriz de Confusão) e Validação Cruzada para avaliar rigorosamente os modelos supervisionados. A avaliação do detector de anomalias é mais qualitativa neste caso, baseada na análise visual do gráfico e na interpretação das anomalias detectadas no contexto da execução do notebook.

Ensemble Methods: A utilização do Random Forest no notebook demonstra o poder dos métodos de ensemble, que combinam múltiplos modelos para obter melhor performance e robustez.

5. Resultados

[Nesta seção, você descreverá os resultados obtidos após executar os scripts. Substitua os colchetes por suas observações.]

Desempenho do Modelo de Classificação (do Entrega\_executado.ipynb):

Após a execução e avaliação dos quatro modelos de classificação no notebook, o modelo [Nome do Melhor Modelo, ex: Random Forest] apresentou o melhor desempenho geral.

A acurácia média obtida na validação cruzada de 5 folds para este modelo foi de aproximadamente [Valor da Acurácia CV, ex: 0.93].

No conjunto de teste, o modelo alcançou uma acurácia de [Valor da Acurácia no Teste, ex: 0.94].

Analisando o relatório de classificação e a matriz de confusão para o melhor modelo no conjunto de teste, observou-se:

Precisão para a classe "Aprovado" (1): [Valor]

Recall para a classe "Aprovado" (1): [Valor]

F1-Score para a classe "Aprovado" (1): [Valor]

Precisão para a classe "Reprovado" (0): [Valor]

Recall para a classe "Reprovado" (0): [Valor]

F1-Score para a classe "Reprovado" (0): [Valor]

[Descrever brevemente a matriz de confusão: Quantos foram classificados corretamente/incorretamente para cada classe.]

[Se houver, mencionar alguma observação do gráfico pairplot, como correlações visuais entre features e a variável alvo.]

Monitoramento de Recursos e Detecção de Anomalias (do resource\_monitoring\_anomalies.png e saída do analyze.py):

O gráfico resource\_monitoring\_anomalies.png mostra o uso de CPU e memória durante o período de execução do notebook Entrega.ipynb.

Observou-se um padrão de uso de recursos caracterizado por [Descrever o padrão geral: ex: uso baixo no início, picos durante certas fases, estabilização].

Picos notáveis no uso de CPU e/ou memória foram observados em torno de [Descrever os momentos dos picos, ex: durante o treinamento dos modelos mais complexos como Random Forest]. Isso é esperado, dado o custo computacional dessas tarefas.

O algoritmo Isolation Forest identificou [Número] pontos de dados como anomalias, destacados em vermelho no gráfico.

Analisando os pontos anômalos: [Descrever onde ocorreram as anomalias - coincidem com os picos esperados? Há algum comportamento inesperado? Ex: Um pico súbito e curto de CPU pode ser normal, mas um uso de memória que cresce continuamente e não diminui pode indicar um vazamento, embora improvável neste script específico].

O uso de disco na partição raiz permaneceu relativamente [Estável/Variável?] durante a execução.

6. 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/ML.

7. Trabalhos Futuros

Aprimorar o Monitoramento: Incluir métricas adicionais (rede I/O, processos específicos).

Sofisticar a Detecção de Anomalias: Testar outros algoritmos (ex: One-Class SVM, Autoencoders) ou abordagens baseadas em previsão de séries temporais (ex: LSTM).

Análise Causal: Investigar mais a fundo as causas das anomalias detectadas.

Sistema de Alerta: Implementar um mecanismo para notificar quando anomalias críticas forem detectadas.

Dashboard: Criar uma interface de visualização em tempo real para o monitoramento.

Otimização do Modelo de ML: Realizar ajuste fino de hiperparâmetros no modelo de classificação para potencialmente melhorar ainda mais a performance.

Análise da Métrica de Disco: Incorporar a análise de disco na detecção de anomalias ou em outra análise (ex: previsão de esgotamento de espaço).