UNIVEM

Centro Universitário Eurípides de Marília

Aprendizado de Máquina - Trabalho Final

Gabriel Shimbo - 627917
João Renato - 626856
João Rozante - 626155
Lucas Colombo - 625221
Pedro Renato - 626864

Professor: Luís Hilário Tobler Garcia

Sumario:

1. Objetivo do Projeto

O objetivo deste projeto foi desenvolver um modelo de **Machine Learning** capaz de resolver um problema de **classificação binária**. A tarefa consistia em prever quais passageiros da nave *Spaceship Titanic* foram transportados para uma dimensão alternativa (um valor `True` ou `False`), com base em seus dados pessoais e registros de bordo.

A métrica de avaliação definida pelo Kaggle foi a **Acurácia** (Accuracy), que mede a porcentagem de previsões corretas.

2. Arquitetura e Design do Projeto

O projeto foi desenhado como um **pipeline de processamento de dados linear**, onde cada etapa prepara os dados para a seguinte. O design foi focado em duas prioridades:

1. **Engenharia de Atributos (Feature Engineering):** A maior parte do esforço foi focada em extrair o máximo de informação possível dos dados brutos (como `Cabin` e `PassengerId`).
2. **Simplicidade do Modelo:** Foi escolhido o algoritmo mais simples e interpretável (`DecisionTreeClassifier`) como ponto de partida (baseline), conforme solicitado.

O fluxo de trabalho foi estruturado da seguinte forma:

1. **Carregamento dos Dados:** Leitura dos arquivos `train.csv` e `test.csv`.
2. **Engenharia de Atributos:** Criação de novas colunas (features) a partir de colunas existentes.
3. **Tratamento de Dados Faltantes (Imputação):** Preenchimento de todos os valores `NaN` (nulos).
4. **Codificação Categórica (Encoding):** Transformação de dados de texto e booleanos em números.
5. **Treinamento do Modelo:** Uso dos dados limpos para treinar o algoritmo.
6. **Geração da Submissão:** Formatação das previsões para o padrão do Kaggle.

3. Detalhamento das Etapas (O que cada coisa faz)

Abaixo está a descrição funcional de cada bloco de código principal.

3.1. Carregamento e Preparação Inicial

- **O que faz:** Utiliza a biblioteca **Pandas** (`pd.read_csv`) para carregar os arquivos `train.csv` (dados de treino) e `test.csv` (dados de teste) na memória.
- **Função:** Disponibiliza os dados em DataFrames. Imediatamente, o `PassengerId` do `test_csv` é salvo em uma variável separada (`passenger_ids_teste`), pois ele é necessário para o arquivo final de submissão, mas não deve ser usado no treinamento.

3.2. Engenharia de Atributos

- **O que faz:** Esta etapa cria novas colunas informativas a partir de colunas "sujas" ou complexas.
- **Função:**
    - **`PassengerId` (formato `gggg_pp`):** Foi dividido em duas novas colunas, `Group` e `GroupNumber`, para identificar pessoas que viajavam juntas.
    - **`Cabin` (formato `deck/num/side`):** Esta foi a transformação mais importante. A coluna foi dividida em três: `Deck` (local da nave), `CabinNum` (número) e `Side` (P - Port ou S - Starboard). Isso transforma uma única informação de texto em três potenciais indicadores de previsão.
    - **`TotalSpend`:** Foi criada uma nova coluna somando todos os gastos individuais (`RoomService`, `FoodCourt`, `ShoppingMall`, `Spa`, `VRDeck`). Isso resume a atividade financeira do passageiro.

3.3. Tratamento de Dados Faltantes (Imputação)

- **O que faz:** Preenche todos os valores `NaN` (nulos) do dataset. Modelos de machine learning não conseguem processar valores nulos.
- **Função (Estratégias):**
    - **Colunas de Gastos:** Valores `NaN` foram preenchidos com `0.0`, partindo da premissa de que um registro nulo significa que o passageiro não gastou.
    - **`CryoSleep` (Lógica Inteligente):** Usamos a `TotalSpend`. Se um passageiro com `CryoSleep` nulo tinha `TotalSpend > 0`, seu `CryoSleep` foi preenchido como `False` (não estava dormindo). Se `TotalSpend == 0`, foi preenchido como `True`.
    - **`Age` (Numérico):** `NaN` preenchido com a **mediana** das idades. A mediana é usada (em vez da média) por ser menos sensível a idades extremas (outliers).
    - **`HomePlanet`, `Destination`, `VIP` (Categóricos):** `NaN` preenchidos com a **moda** (o valor mais frequente) de cada coluna.

3.4. Codificação e Preparação (Encoding)

- **O que faz:** Converte todas as colunas que não são numéricas (texto e booleanos) em números, que é o único formato que o modelo entende.
- **Função:**
    - **Booleanos (`CryoSleep`, `VIP`, `Transported`):** Convertidos de `True`/`False` para 1/`0`.
    - **Categóricos (`HomePlanet`, `Deck`, `Side`, `Destination`):** Foi aplicada a técnica de **One-Hot Encoding** (`pd.get_dummies`). Isso transforma uma coluna (ex: `HomePlanet` com 3 planetas) em 3 novas colunas (ex: `HomePlanet_Earth`, `HomePlanet_Mars`, `HomePlanet_Europa`), onde o valor é 1 se o passageiro for daquele local e `0` se não for.
    - **Limpeza Final:** Colunas originais que não são mais úteis (`PassengerId`, `Name`, `Cabin`, etc.) foram removidas.

3.5. Modelagem (Treinamento)

- **O que faz:** Esta é a etapa de inteligência artificial.
- **Função:**
    - **Separação:** Os dados de treino foram divididos em `X_train` (as features, ou seja, todas as colunas de dados) e `y_train` (o alvo, ou seja, apenas a coluna `Transported`).
    - **Algoritmo:** Foi escolhida a **Árvore de Decisão** (`DecisionTreeClassifier`) da biblioteca **Scikit-learn**.
    - **Treinamento:** O comando `model.fit(X_train, y_train)` "ensina" o modelo a encontrar padrões nos dados `X_train` que levem às respostas em `y_train`.

3.6. Geração da Submissão

- **O que faz:** Utiliza o modelo treinado para gerar o arquivo final.
- **Função:**
    - **Previsão:** O comando `model.predict(X_test)` usa o modelo treinado para adivinhar o resultado (1 ou `0`) para todos os passageiros do dataset de teste.
    - **Formatação:** As previsões (1/`0`) são convertidas de volta para o formato exigido pelo Kaggle (`True`/`False`).
    - **Criação do CSV:** Um novo DataFrame é criado contendo os `passenger_ids_teste` (guardados no início) e as previsões (`Transported`).
    - **Salvamento:** O DataFrame é salvo como `submission.csv`, pronto para ser enviado ao Kaggle.

4. Conclusão

O projeto foi projetado como um pipeline completo, capaz de receber dados brutos e ruidosos e transformá-los em um produto final (um arquivo de submissão). O design priorizou a limpeza e a engenharia de atributos (como a separação da `Cabin` e a imputação inteligente do `CryoSleep`) como os passos mais críticos para o sucesso, utilizando um modelo de Árvore de Decisão como classificador base.