□ Como Recriar o Ambiente Virtual

```
# Crie o ambiente dentro da raiz do projeto
python -m venv venv

# Ative o ambiente
# Windows:
venv\Scripts\activate

# Linux/macOS:
source venv/bin/activate

# Instale os pacotes
pip install -r requirements.txt
```

Análise de Avaliações de Filmes - MovieLens 100k

Este projeto realiza uma análise estatística do dataset `u.data` do repositório MovieLens 100k (https://grouplens.org/datasets/movielens/100k/), utilizando a biblioteca `pandas` para obter estatísticas por filme e por usuário.

□ Explicação Detalhada do Código `u.data.py`

```
import pandas as pd
```

Importa a biblioteca `pandas`, usada para manipulação de dados tabulares.

```
import os
```

Importa a biblioteca `os`, usada para lidar com caminhos de arquivos de forma portátil.

```
current_dir = os.path.dirname(os.path.abspath(__file__))
```

Obtém o diretório atual onde o script está sendo executado.

```
file_path = os.path.join(current_dir, '..', 'ml-100k', 'u.data')
```

Cria o caminho para o arquivo `u.data`, assumindo que está em uma pasta acima (`..`) na estrutura de diretórios.

```
columns = ['user_id', 'movie_id', 'rating', 'timestamp']
```

Define os nomes das colunas do dataset, que não possui cabeçalho.

```
data_frame_movies = pd.read_csv(file_path, sep='\t', names=columns)
```

Lê o arquivo `u.data` separando as colunas por tabulação (`\t`), aplicando os nomes definidos.

```
print(data_frame_movies['movie_id'].nunique())
```

Imprime o número de filmes distintos avaliados no dataset.

```
movie_statistics = data_frame_movies.groupby(['movie_id'])['rating'].agg([
    ('movie_rating_mean', 'mean'),
    ('movie_rating_standard_deviation', 'std'),
    ('movie_rating_variance', 'var')
]).reset_index()
```

Agrupa as avaliações por filme (`movie_id`) e calcula a média, o desvio padrão e a variância das avaliações de cada filme.

```
print(movie_statistics)
```

Exibe o DataFrame contendo as estatísticas por filme.

```
data_frame_users = pd.read_csv(file_path, sep='\t', names=columns)
```

Lê novamente o mesmo dataset, agora com foco nos usuários.

```
user_statistics = data_frame_users.groupby('user_id')['rating'].agg([
    ('user_rating_mean', 'mean'),
    ('user_rating_standard_deviation', 'std'),
    ('user_rating_variance', 'var')
]).reset_index()
```

Agrupa as avaliações por usuário (`user_id`) e calcula as estatísticas para cada um.

```
data_frame_users_and_users_statistics = pd.merge(data_frame_users, user_statistics, on='user_id')
```

Une o DataFrame original com as estatísticas calculadas, ligando pelo campo `user_id`.

```
print(data_frame_users_and_users_statistics)
```

Exibe o DataFrame completo com os dados das avaliações e estatísticas de cada usuário.

```
uniform_rating_users = user_statistics.sort_values(by='user_rating_standard_deviation')
```

Ordena os usuários pelo desvio padrão das suas avaliações — os mais "uniformes" vêm primeiro.

```
unique_uniform_rating_users = uniform_rating_users.drop_duplicates(subset='user_id')
```

Remove possíveis duplicatas de usuários.

```
print(unique_uniform_rating_users.loc[:, ['user_id', 'user_rating_standard_deviation']].head(10))
```

Imprime os 10 usuários com menor desvio padrão, ou seja, os mais consistentes nas suas avaliações.

▶ Como Executar o Script

Estrutura de diretórios recomendada:

```
projeto/
├── ml-100k/
│   └── u.item
├── 2-movies-data/
│   └── u.data.py
│   └── README.md
└── requirements.txt
```

Para executar o script:

```
python 1-rating-data/u.data.py
```

Análise de Gêneros de Filmes - MovieLens 100k

Este projeto realiza uma análise estatística do arquivo `u.item` do repositório MovieLens 100k (https://grouplens.org/datasets/movielens/100k/), extraindo informações sobre os gêneros e dados dos filmes.

□ Explicação Detalhada do Código `u.item.py`

```
import pandas as pd
```

Importa a biblioteca `pandas` para manipulação de dados.

```
import os
```

Importa a biblioteca `os` para manipular caminhos de arquivos de forma portátil.

```
current_dir = os.path.dirname(os.path.abspath(__file__))
```

Determina o diretório onde o script atual está localizado.

```
file_path = os.path.join(current_dir, '..', 'ml-100k', 'u.item')
```

Constrói o caminho para o arquivo `u.item`, assumindo que ele está na pasta `ml-100k`, um nível acima.

```
columns = ['movie_id', 'title', 'release_date', 'video_release_date', 'IMDb_URL', 'unknown', 'Action', 'Adventure', 'Animation', "Ch
```

Define manualmente os nomes das colunas, já que o arquivo não possui cabeçalho.

```
movies_data_frame = pd.read_csv(file_path, sep='|', names=columns, encoding='latin-1', engine='python')
```

Lê o arquivo `u.item` utilizando `|` como separador, com codificação `latin-1` e usando o engine `python` (necessário por causa da codificação).

```
number_of_movies_by_gender = movies_data_frame.iloc[:, 5:].sum()
```

Soma as colunas correspondentes aos gêneros de filmes (a partir da 6ª coluna em diante) para contar quantos filmes existem por gênero.

```
gender_with_more_movies = number_of_movies_by_gender.idxmax()
```

Identifica o gênero com a maior quantidade de filmes.

```
print(f"The gender with more movies is {gender_with_more_movies}")
```

Exibe no console qual gênero tem mais filmes.

```
amount_of_missing_data = movies_data_frame.isnull().sum().sum()
```

Calcula a quantidade total de dados ausentes no DataFrame.

```
print(amount_of_missing_data)
```

Exibe a quantidade de dados faltantes (se houver).

▶ Como Executar o Script

Estrutura de diretórios recomendada:

```
projeto/
├── ml-100k/
│   └── u.item
├── 2-movies-data/
│   └── u.item.py
│   └── README.md
└── requirements.txt
```

Para executar o script:

```
python 2-movies-data/u.item.py
```

□ Análise de Filmes com Estatísticas e Normalização - MovieLens 100k

Este projeto trabalha com os arquivos `u.item` e `u.data` do dataset MovieLens 100k para extrair informações sobre os filmes, calcular estatísticas e aplicar diferentes técnicas de **normalização** — uma etapa fundamental em projetos de Machine Learning.

□ Explicação do Código `new_dat_frame_movies.py`

□ 1. Importação de bibliotecas

```
import pandas as pd
import os
```

- `pandas`: permite manipular dados em formato de tabela.
- `os`: permite montar caminhos de arquivos automaticamente.

□ 2. Definindo os caminhos dos arquivos

```
current_dir = os.path.dirname(os.path.abspath(__file__))
uitem_file_path = os.path.join(current_dir, '..', 'ml-100k', 'u.item')
udata_file_path = os.path.join(current_dir, '..', 'ml-100k', 'u.data')
```

- Localiza automaticamente onde está o script e monta os caminhos para os arquivos `u.item` e `u.data`.

□ 3. Criando um DataFrame com os gêneros dos filmes

```
columns = [...]
movies_data_frame = pd.read_csv(...)
genre_columns = movies_data_frame.columns[5:]
```

- Define os nomes das colunas e carrega os dados do arquivo `u.item`.
- Pega apenas as colunas relacionadas aos gêneros dos filmes.

```
movies_genre = []
for index, row in movies_data_frame.iterrows():
    ...
```

- Cria uma nova coluna chamada `genre`, contendo uma string com os gêneros de cada filme, como "`Action, Comedy`".

```
new_movies_data_frame = movies_data_frame.drop(columns=genre_columns)
new_movies_data_frame['genre'] = movies_genre
```

- Remove as colunas de gênero em forma binária e adiciona a versão em texto.

□ 4. Calculando estatísticas de avaliação dos filmes

```
columns = ['user_id', 'movie_id', 'rating', 'timestamp']
data_frame_ratings = pd.read_csv(...)
movies_statistics = data_frame_ratings.groupby('movie_id')['rating'].agg(...)
```

- Carrega o arquivo `u.data`, que contém as avaliações dos usuários.
- Agrupa por `movie_id` e calcula:
  - total de avaliações (`count`)
  - soma das notas (`sum`)
  - média (`mean`)
  - maior e menor nota (`max`, `min`)
  - desvio padrão e variância (`std`, `var`)

```
new_movies_statistics_data_frame = pd.merge(...)
```

- Junta os dados de filmes com os dados de avaliação por `movie_id`.

□ 5. Filmes mais e menos avaliados

```
most_evaluated_movies = ...
least_evaluated_movies = ...
```

- Ordena os filmes pelo número de avaliações e exibe os 10 com mais e os 10 com menos avaliações.

□ 6. Normalização dos dados

A normalização é usada para ajustar os valores para uma escala comum, essencial em algoritmos de Machine Learning.

□ Min-Max

```
from sklearn.preprocessing import MinMaxScaler
```

- Transforma os valores para um intervalo entre 0 e 1.

□ Normalização pela média

```
def mean_normalization(X):
    return (X - X.mean()) / (X.max() - X.min())
```

- Centraliza os dados em torno da média.

□ Z-Score

```
from sklearn.preprocessing import StandardScaler
```

- Transforma os dados com média 0 e desvio padrão 1 (padrão estatístico comum).

Em todos os casos, os dados normalizados são mesclados de volta com os dados dos filmes.

▶ Como Executar o Script

Estrutura esperada:

```
projeto/
├── ml-100k/
│   ├── u.item
│   └── u.data
├── 3-new-data-set/
│   └── new_dat_frame_movies.py
│   └── README.md
└── requirements.txt
```

Passos no terminal

1. Ativar o ambiente virtual:

```
venv\Scripts\activate  # Windows
source venv/bin/activate  # Linux/macOS
```

2. Executar o script:

```
python 3-new-data-set/new_dat_frame_movies.py
```

□ Explicações Complementares (Etapas detalhadas)

□ Construção da coluna "genre" (texto descritivo dos gêneros do filme)

```
movies_genre = []

for index, row in movies_data_frame.iterrows():
    genre_by_movie = []
    for genre in genre_columns:
        if row[genre] == 1:
            genre_by_movie.append(genre)
    genre_str = ', '.join(genre_by_movie)
    movies_genre.append(genre_str)
```

- Esse trecho percorre **linha por linha** do DataFrame de filmes.
- Para cada filme (`row`), ele verifica quais colunas de gênero possuem valor `1` (indicando que o filme pertence àquele gênero).
- Cria uma **lista de strings** com os nomes dos gêneros e depois transforma em uma única string separada por vírgulas.
- Exemplo: um filme com valores 1 nas colunas `Action` e `Thriller` terá como saída: `"Action, Thriller"`.

```
new_movies_data_frame = movies_data_frame.drop(columns=genre_columns)
new_movies_data_frame['genre'] = movies_genre
```

- Remove as colunas binárias e substitui por uma única coluna de texto contendo os gêneros.

□ Estatísticas por filme: explicação de cada métrica

```
movies_statistics = data_frame_ratings.groupby('movie_id')['rating'].agg(
    rating_total='count',
    rating_sum='sum',
    rating_mean='mean',
    rating_max_value='max',
    rating_min_value='min',
    rating_standard_deviation='std',
    rating_variance='var'
).reset_index()
```

Para cada filme (`movie_id`), calcula:

- `rating_total`: número de vezes que o filme foi avaliado.
- `rating_sum`: soma total das notas recebidas.

- `rating_mean`: média das notas.
- `rating_max_value`: maior nota recebida.
- `rating_min_value`: menor nota recebida.
- `rating_standard_deviation`: mede o quanto as notas variam da média.
- `rating_variance`: a dispersão dos dados (quadrado do desvio padrão).

Essas métricas ajudam a entender **a popularidade, consistência e variabilidade** das avaliações de cada filme.

□ Normalizações com mais detalhes

□ Min-Max

```
scaler = MinMaxScaler()
normalized_data_min_max = scaler.fit_transform(...)
```

- Transforma os valores para uma escala de **0 a 1** com base no valor mínimo e máximo de cada coluna.
- Útil quando você quer manter a proporção dos dados, mas padronizar a escala.

□ Normalização pela média

```
def mean_normalization(X):
    return (X - X.mean()) / (X.max() - X.min())
```

- Centraliza os dados em torno de zero considerando a **média e o intervalo (máx - mín)**.
- Isso ajuda a evitar que os valores com maiores magnitudes dominem os algoritmos de aprendizado.

□ Z-score

```
StandardScaler()
```

- Calcula: `z = (x - média) / desvio padrão`
- Resultado: dados com **média 0 e desvio padrão 1**.
- Útil para algoritmos que assumem distribuição normal (como regressão logística, SVM, etc).

□ Junção final dos dados normalizados

Em todas as normalizações (MinMax, Média, Z-score), o resultado é um novo DataFrame que:

- Contém os valores normalizados (`rating_total`, `rating_sum` em nova escala)
- É unido de volta ao conjunto original usando `pd.merge(..., on='movie_id')` para preservar a identidade de cada filme.

Isso garante que cada filme tenha todas as suas métricas e normalizações combinadas em um só lugar.
