PMLDL. Assignment #2:
Movie Recommender System

Final Report

- GitHub Repository: https://github.com/KHiMAeRA05/PMLDL_movie_recommender_system
- Author: Voronova Aleksandra
- Email: a.voronova@innopolis.universiy
- Group Number: BS21-DS-02

Introduction

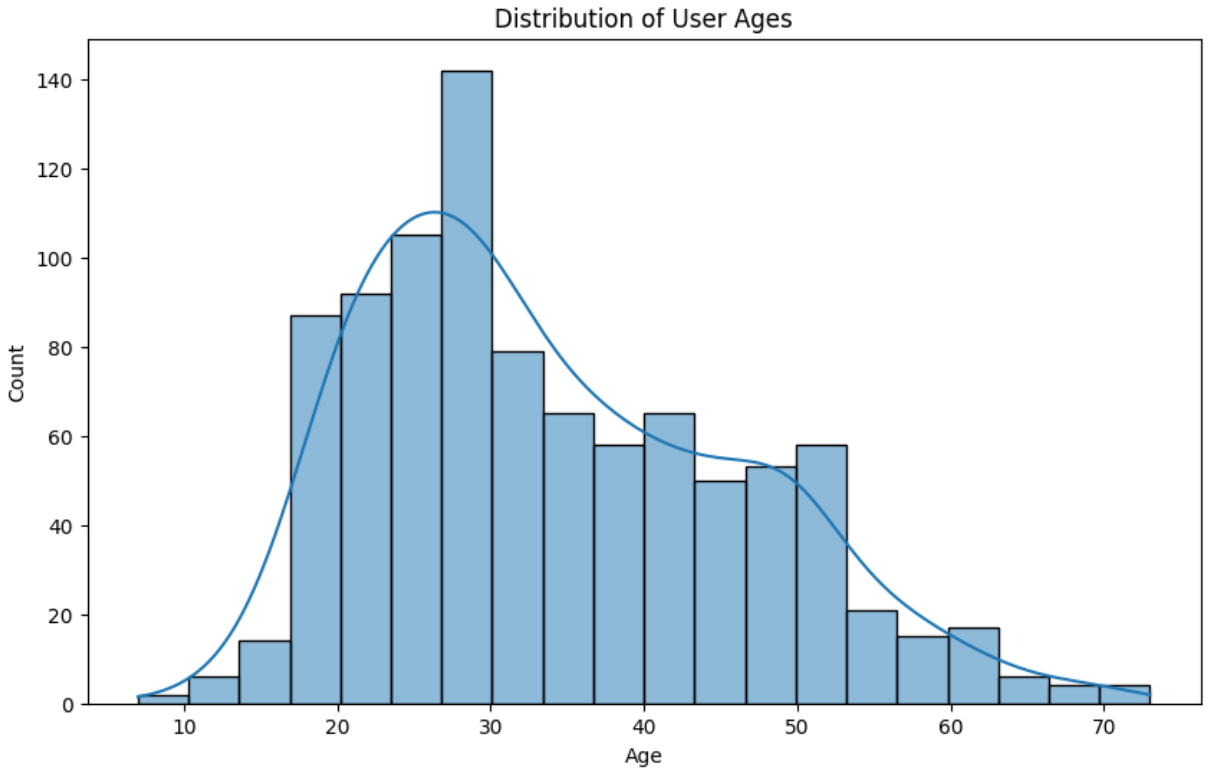
The task involves leveraging machine learning techniques to build a movie recommender system. The MovieLens 100K dataset was used. Users' demographic information, including age, gender, occupation, and zip code, along with a list of their favorite movies, forms the basis for generating personalized recommendations.

Data Analysis

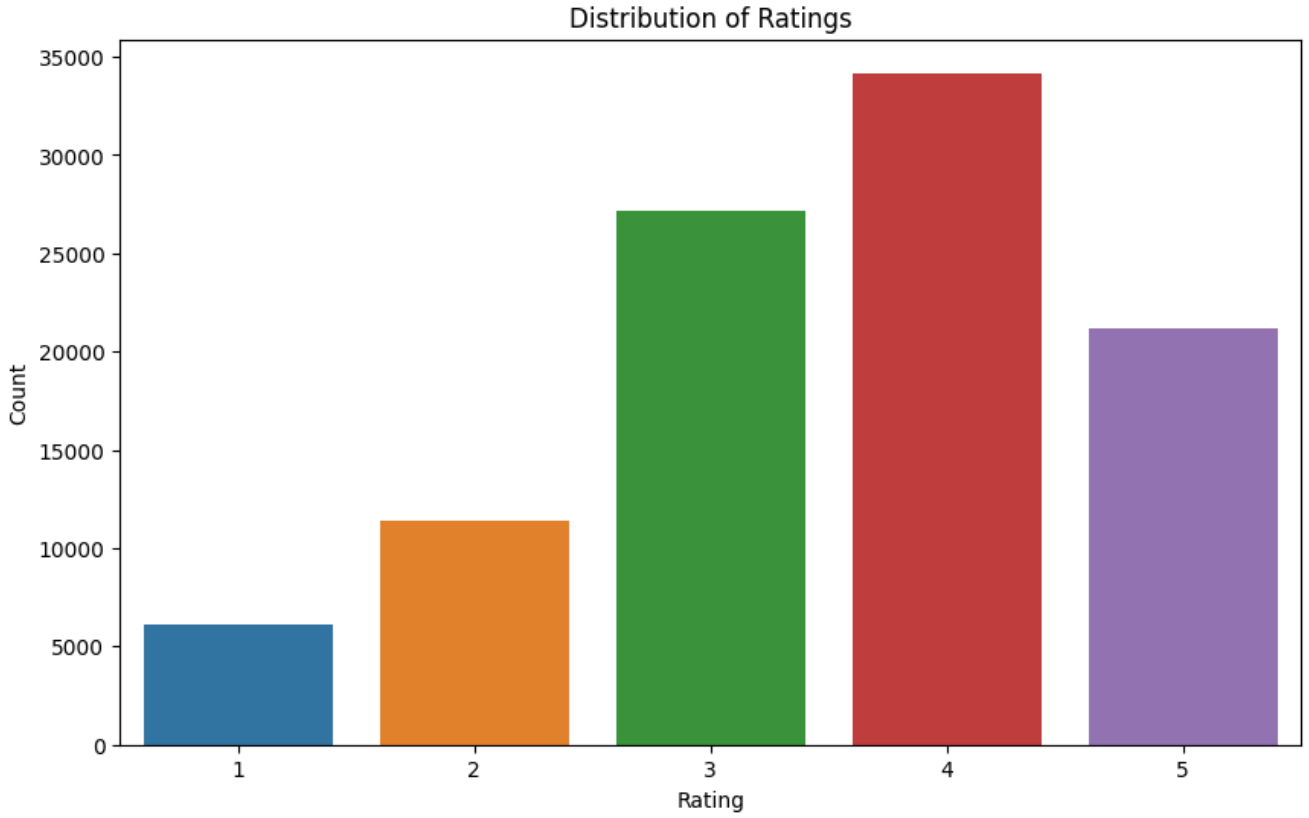
Dataset Overview

The MovieLens 100K dataset consists of 100,000 ratings from 943 users on 1682 movies. Before delving into the model implementation, an exploration of the dataset was conducted to gain insights into its structure and characteristics.

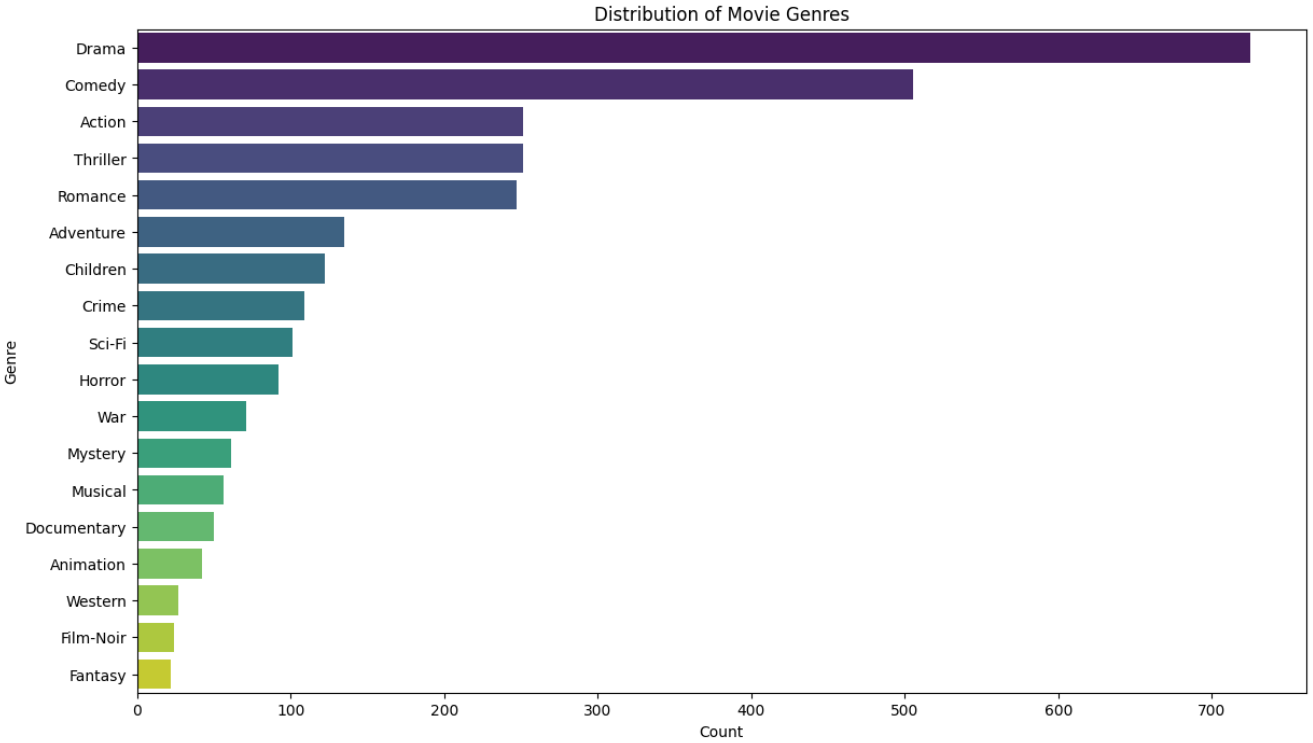
Age Distribution



Ratings distribution



Genres distribution



Top-10 Movie Titles

	Movie Title	Average Rating
6	Aiqing wansui (1994)	5.0
9	Entertaining Angels: The Dorothy Day Story (1996)	5.0
5	Great Day in Harlem, A (1994)	5.0
1	Marlene Dietrich: Shadow and Light (1996)	5.0
8	Prefontaine (1997)	5.0
2	Saint of Fort Washington, The (1993)	5.0
7	Santa with Muscles (1996)	5.0
3	Someone Else's America (1995)	5.0
4	Star Kid (1997)	5.0
0	They Made Me a Criminal (1939)	5.0

Key Observations

1. No empty values
2. No missing values

Main Insight:

not data preprocessing pipeline is required.

Model Implementation

Now, when I saw my data, made some visualizations and insights, it's time to start implementing my model. Since the main goal of the assignment is to create a movie recommender system I decided to choose a popular python library: Surprise.

Choice of Model: Surprise Library

Advantages:

1. **Ease of Use:** Surprise provides a high-level, user-friendly API, simplifying the implementation of complex recommendation algorithms.
2. **Versatility:** The library includes a range of built-in collaborative filtering algorithms, such as Singular Value Decomposition (SVD), k-Nearest Neighbors (k-NN), and Non-negative Matrix Factorization (NMF), offering flexibility in choosing the most appropriate algorithm.
3. **Integration with Common Datasets:** Surprise seamlessly integrates with various common recommendation datasets, streamlining the data loading and preprocessing steps.
4. **Evaluation Metrics:** The library offers convenient functions for evaluating model performance using standard metrics like Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

But since I utilize Singular Value Decomposition (SVD), surprise have some disadvantages regarding the *collaborative filtering algorithm*.

Disadvantages:

1. **Cold start:** Collaborative filtering methods, including Surprise, often struggle with the cold start problem. This occurs when there is insufficient user or item interaction data, making it challenging to provide accurate recommendations for new users or items.
2. **Data Sparsity:** Collaborative filtering methods become less effective when dealing with sparse datasets. If users have rated only a small subset of items, it may lead to a lack of sufficient information for accurate recommendations.
3. **Popularity Bias:** Collaborative filtering methods, including Surprise, are prone to popularity bias. They tend to recommend popular items more frequently, as these items have more user interactions.

Training Process

Data Preprocessing

Before training the model, it was imperative to prepare the MovieLens 100K dataset for compatibility with the Surprise library. This involved creating a Surprise Dataset object and specifying the rating scale.
To evaluate the model effectively, I split the dataset into training and testing sets using Surprise's *train_test_split function*. This ensures that the model's performance is assessed on unseen data.

```
reader = Reader(rating_scale=(1, 5))
dataset = Dataset.load_from_df(df[['user_id', 'item_id', 'rating']], reader)

trainset, testset = train_test_split(dataset, test_size=0.2)
```

Model Training

I selected the Singular Value Decomposition (SVD) algorithm from the Surprise library for this movie recommender system. SVD is a matrix factorization technique commonly used in collaborative filtering.

```
model = SVD()
model.fit(trainset)
```

Evaluation

To assess the performance of the implemented movie recommender system, I employed standard evaluation metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). These metrics provide valuable insights into the accuracy and precision of the recommendations made by the system.

Data Preparation and Model Loading

The dataset was preprocessed and converted into a Surprise-compatible format using the Surprise library. The dataset included user-item interactions with corresponding ratings, and a reader was defined with a rating scale ranging from 1 to 5.
The pre-trained recommender model was loaded from the specified file path.

```
def main():
    # File paths for the trained model and testing dataset
    model_file = '/kaggle/working/PMLDL_movie_recommender_system/models/model.pkl'
    data_file = '/kaggle/working/PMLDL_movie_recommender_system/data/interim/preprocessed_data.csv'

    data = load_data(data_file)

    trained_model = load_surprise_model(model_file)['algo']

    surprise_test_data = load_surprise_data(data)
```

Evaluation Metrics

The model's performance was evaluated using cross-validation, a robust technique to ensure reliable results. I employed a 5-fold cross-validation strategy, where the dataset was divided into five subsets, and the model was trained and evaluated five times, with a different subset used as the test set in each iteration.

```
def evaluate(model, data):
    results = cross_validate(model, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
    return results
```

Results

The evaluation results are summarized below:

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9339  0.9342  0.9400  0.9384  0.9379  0.9369  0.0024
MAE (testset)     0.7374  0.7377  0.7420  0.7375  0.7402  0.7390  0.0019
Fit time          1.39    1.65    1.30    1.57    1.55    1.49    0.13
Test time         0.22    0.22    0.20    0.33    0.20    0.23    0.05
```

Conclusion

The SVD algorithm demonstrated promising performance with a mean RMSE of 0.9369 and a mean MAE of 0.7390 across the 5-fold cross-validation. Also, the model exhibited reasonable fit and test times, ensuring efficient recommendations.

In summary, the implemented movie recommender system demonstrates promising performance, with the chosen collaborative filtering algorithm delivering reliable results on the MovieLens 100K dataset.

For more details on the code, please check my GitHub repository (https://github.com/KHiMAeRA05/PMLDL_movie_recommender_system).