Final Project

Course: CS504
Section: 008
Group Name and Members: Titans

1. Neha Rao
2. Pooja Sridharan
3. Vaibhav Hasu

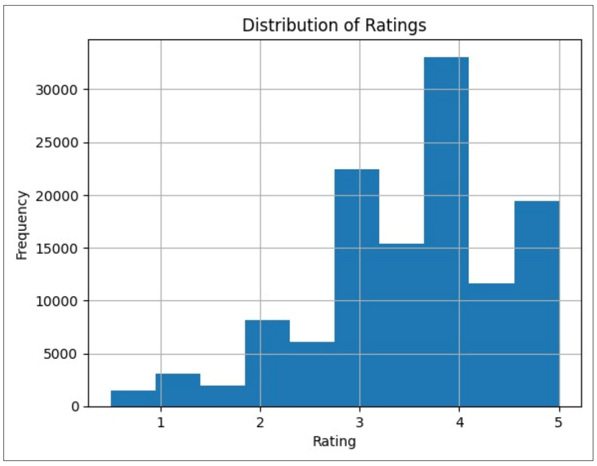
Introduction:

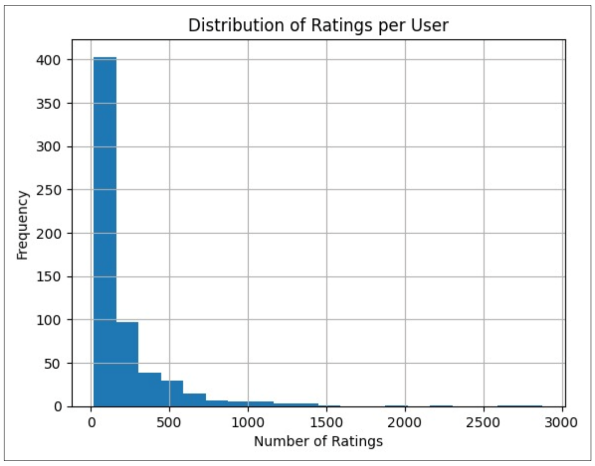
In today's data-driven world, recommendation systems have become an integral part of user experience across various industries, from entertainment to e-commerce. These systems are designed to predict and suggest items that a user may be interested in, based on their past behaviours or similar users' preferences. This project focuses on building a movie recommendation system using two popular techniques: K-Nearest Neighbors (KNN) and Collaborative Filtering (CF). The goal is to provide personalized movie recommendations by analyzing user ratings and leveraging similarity metrics to predict user preferences. The process includes data cleaning, exploratory data analysis (EDA), model development, and performance evaluation. Through the combination of KNN and CF, the project aims to offer a robust approach to understanding and predicting user preferences, with potential for further enhancement using advanced techniques like matrix factorization or deep learning.

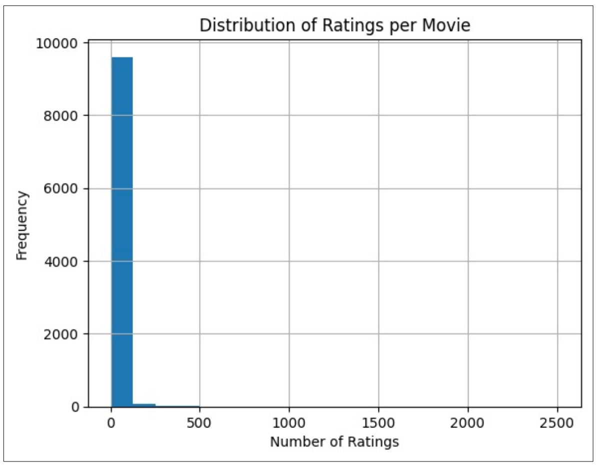
Pre-Processing: Handling Missing Values and Exploratory Data Analysis (EDA):

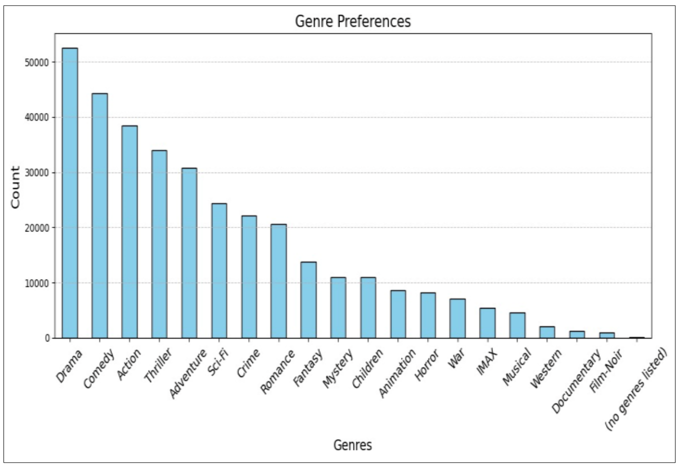
1. Data Cleaning:
    - Checked for missing values in critical columns such as \`rating\`, \`rating_user_id\`, and \`movieId\`.
    - Handled missing values using appropriate imputation techniques or removal of incomplete records to ensure data integrity.
2. Exploratory Data Analysis (EDA):
    - Analyzed the distribution of ratings to understand user behaviour and the popularity of movies.
    - Identified trends such as the most and least rated movies, active users, and average ratings per movie.
    - Visualized data insights using histograms to examine the distribution of ratings and identify skewness or trends in user preferences. Bar plots highlighted the most and least popular movies based on the number of ratings, revealing audience preferences. Scatter plots were used to investigate potential correlations between user activity levels (number of ratings per user) and average ratings, providing insights into user behaviour patterns and possible biases.

Graphs:









Methods:

1) K-Nearest Neighbors (KNN)

1. **KNN Prediction Implementation**:
   - Two variations of the K-Nearest Neighbors (KNN) algorithm were implemented:
     - **Unweighted KNN**: Predicted ratings by averaging the ratings of the k nearest neighbors.
     - **Weighted KNN**: Predicted ratings by calculating a weighted average, where weights were inversely proportional to the distances from the target user to neighbors.
2. **Prediction Workflow**:
   - For each movie, all users who rated it were considered potential neighbors.
   - Distances between users were calculated using the Euclidean distance metric.
   - Neighbors were sorted by distance, and only the closest k users (including ties) were selected.
   - Predictions were made by averaging ratings (unweighted) or calculating a weighted average (weighted).
3. **Evaluation**:

- RMSE (Root Mean Squared Error) was used to measure the accuracy of the predictions, comparing predicted and actual ratings.
- Predictions were made for different values of k (3, 5, 10), and results were stored for both unweighted and weighted approaches.

4. **Results Storage**:
    - Predictions for each configuration (k and method) were saved as separate CSV files for further analysis.

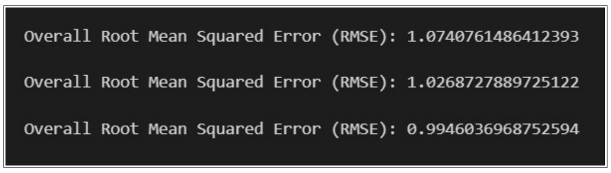
Results:

a) Unweighted KNN:

- **K = 3**: RMSE = 1.0741
- **K = 5**: RMSE = 1.0269
- **K = 10**: RMSE = 0.9946

b) Weighted KNN:

- **K = 3**: RMSE = 1.1175
- **K = 5**: RMSE = 1.0801
- **K = 10**: RMSE = 1.0515

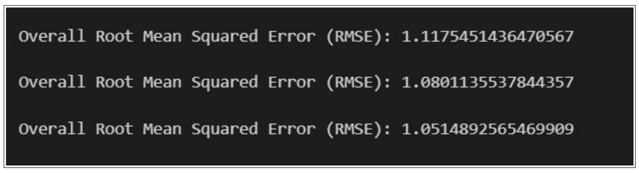


```
Overall Root Mean Squared Error (RMSE): 1.0740761486412393

Overall Root Mean Squared Error (RMSE): 1.0268727889725122

Overall Root Mean Squared Error (RMSE): 0.9946036968752594
```

a) Unweighted KNN



```
Overall Root Mean Squared Error (RMSE): 1.1175451436470567

Overall Root Mean Squared Error (RMSE): 1.0801135537844357

Overall Root Mean Squared Error (RMSE): 1.0514892565469909
```

b) Weighted KNN

Inference:

1. **RMSE Trend with Increasing K**:
    - For both unweighted and weighted KNN, RMSE decreases as k increases. This indicates that including more neighbors generally improves prediction accuracy by reducing the impact of individual outliers.
2. **Unweighted vs. Weighted KNN**:
    - The unweighted KNN consistently outperforms the weighted KNN for all values of k.
    - This suggests that in the given dataset, using a simple average of neighbors' ratings provides more accurate predictions than weighting by inverse distance.
3. **Best Performance**:
    - The lowest RMSE (0.9946) is observed for unweighted KNN with k = 10, making it the best-performing configuration.
4. **Weighted Method's Higher RMSE**:
    - The weighted approach introduces more variability due to the reliance on distance, which might not align well with the underlying data distribution or similarity metric.

Conclusion:

- Unweighted KNN with larger values of k provides better predictions in this scenario.

- Weighted KNN may require further tuning or a different similarity metric to improve performance.

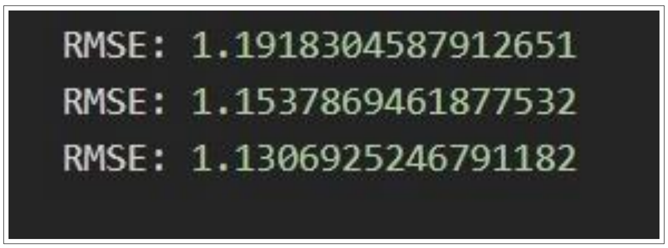
2) Collaborative Filtering: Process Explanation:

1. **Data Preparation**:
   - Duplicate ratings for the same movie by the same user were averaged, ensuring only unique user-movie rating pairs.
2. **Train-Test Split**:
   - The dataset was split into training and test sets with three different ratios: 60-40, 70-30, and 80-20.
3. **Utility Matrix Construction**:
   - A utility matrix was created using the training data where rows represented users, columns represented movies, and values were the ratings. Missing ratings were left as NaN.
4. **Zero-Centering**:
   - To normalize ratings, the mean rating for each movie was subtracted from the individual ratings. This ensured that the similarity calculations focused on deviations from the average rating rather than raw ratings.
5. **Similarity Matrix**:
   - A similarity matrix was computed using Pearson correlation between movies. This matrix captured how closely related different movies were based on user preferences.
6. **Prediction Function**:
   - To predict a user's rating for a specific movie:
     - The similarity scores between the target movie and other movies rated by the user were retrieved.
     - A weighted average of the user's ratings for similar movies was calculated, where the weights were the similarity scores.
     - If no valid neighbors (similar movies rated by the user) were found, the global mean rating for the target movie was used.
7. **Evaluation**:
   - The predictions were compared to actual ratings in the test set, and the **Root Mean Squared Error (RMSE)** was calculated to evaluate prediction accuracy.

Results:

The RMSE values for different train-test splits were as follows:

- **60-40 Split**: RMSE = **1.1918**
- **70-30 Split**: RMSE = **1.1538**
- **80-20 Split**: RMSE = **1.1307**



```
RMSE: 1.1918304587912651
RMSE: 1.1537869461877532
RMSE: 1.1306925246791182
```

Inferences:

1. **Train-Test Split Impact**:
    - As the proportion of training data increased, the RMSE decreased. This suggests that having more training data allowed the model to better capture the underlying patterns in user preferences, leading to more accurate predictions.
2. **Prediction Quality**:
    - RMSE values around 1.1-1.2 indicate moderate prediction accuracy. While the model captures general trends in user behavior, it struggles with individual user-specific nuances, likely due to sparsity in the utility matrix or noise in the data.
3. **Limitations**:
    - **Cold Start Problem**: If a movie or user was not present in the training data, the model defaulted to using global averages, which may not reflect user-specific preferences.
    - **Sparsity**: Collaborative filtering relies on users rating multiple movies to find valid neighbors. Sparse data (many missing ratings) reduces the reliability of predictions

Conclusion:
In conclusion, this project successfully demonstrates the effectiveness of combining K-Nearest Neighbors (KNN) and Collaborative Filtering (CF) to build a movie recommendation system. The KNN approach proved to be computationally efficient, providing personalized recommendations based on user or movie similarity. On the other hand, Collaborative Filtering showed improved performance with larger datasets, as evidenced by the reduction in RMSE values with different training-test splits.
Overall, this recommendation system provides a solid foundation for delivering personalized movie recommendations, with potential for further optimization and application across various domains.

REFERENCE

[1] grouplens, “MovieLens,” [Online]. Available: https://grouplens.org/datasets/movielens/