MOVIE RECOMMENDATION SYSTEM

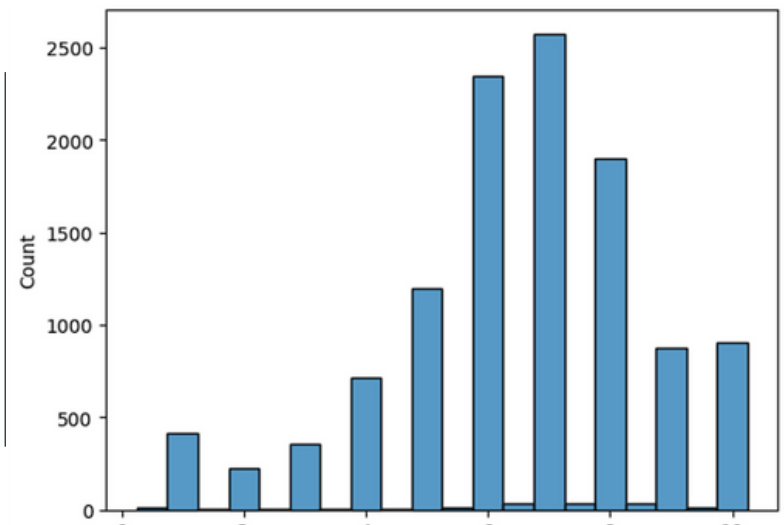
PERSONALIZED RECOMMENDATIONS & COLD-START SOLUTIONS

PROBLEM STATEMENT

This project addresses the challenge of personalized movie recommendations for users with limited data (cold-start problem) by integrating collaborative filtering with vector search. Effective recommendations are essential in e-commerce, streaming platforms, and personalized content delivery.

DATA

- Dataset: 11,506 American movies (from 1970 to 2023) with ratings from 11,675 users.
- Sources:
  - Movie details: Self-collected from Wikipedia.
  - Genres and Ratings: Gathered and cleaned from The Movie Database (TMDb) API.
- Data Preprocessing: Missing values and outliers addressed
- Challenges: The data contains a highly sparse utility matrix, meaning that many users have rated only a few movies. This sparsity limits traditional collaborative filtering approaches, which require substantial interaction data to perform well.



WE CAN SEE THAT MOST OF THE RATINGS IN THE DATA ARE 6.0, 7.0 AND 8.0. IT MEANS THAT MOST OF THE MOVIES ARE GOOD.

MODEL SELECTION CRITERIA AND COMPARATIVE ANALYSIS:

Model Selection

- Collaborative Filtering Algorithms: Selected for proven accuracy in recommendation systems.
- SVD++: Chosen due to lowest RMSE, indicating better prediction accuracy.
- Cold-Start Solution: Used vector embeddings via Sentence Transformers for similarity-based recommendations for users with limited data.

Comparative Analysis

- Metrics Used: Accuracy (RMSE), computational efficiency, and robustness in sparse data.
- Results: SVD++ outperformed other algorithms in prediction accuracy. Vector embeddings successfully addressed the cold-start problem by using movie content similarities.

MLOPS WORKFLOW

1. Data Collection: Retrieve data from Wikipedia and TMDb API.
2. Data Cleaning and Preprocessing: Handle missing values and outliers, encode categorical data, and create utility matrices.
3. Embedding Generation: Use Sentence Transformer to create vector embeddings for movies and users.
4. Model Training and Evaluation: Train collaborative filtering models with hyperparameter tuning. Evaluate using metrics like RMSE.
5. Vector Database Storage: Store embeddings in Chroma DB for efficient retrieval.
6. Recommendation System Deployment: Use Flask for the web app, enabling users to interact with the recommendation system.
7. Continuous Monitoring: Track performance and user satisfaction, updating embeddings and models as needed.

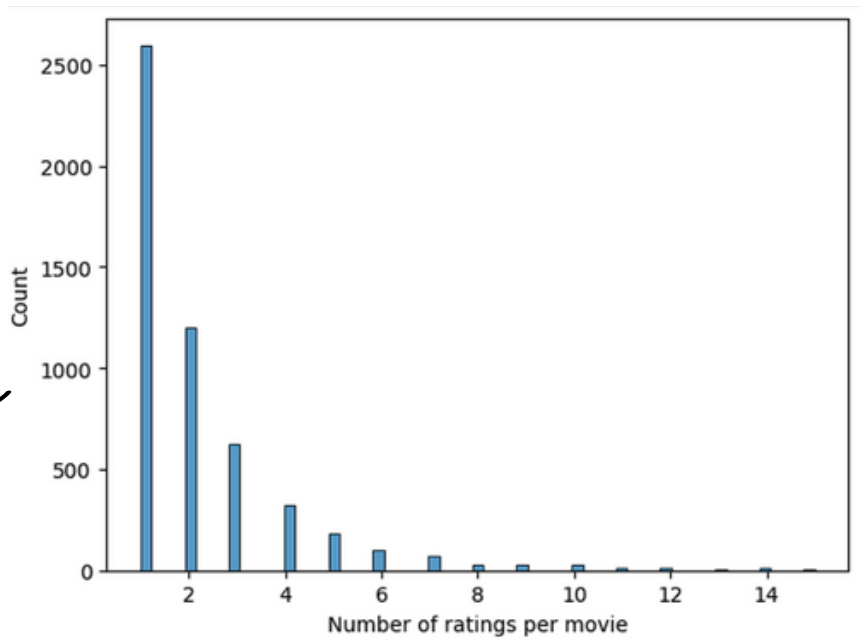
RESULT

```
RAW USER: 463
INNER USER: 463

RATED:
Jaws

RECOMMENDED:
Deliverance
Hickey & Boggs
Monty Python and the Holy Grail
The Princess Bride
Bert Rigby, You're a Fool
The Adventures of Baron Munchausen
Dead Bang
The Ballad of Little Jo
Forrest Gump
Ride with the Devil
The Lord of the Rings: The Two Towe
How to Lose a Guy in 10 Days
The Best Exotic Marigold Hotel
Nightcrawler
American Made
Coco
Alpha
Sound of Metal
Wish Dragon
Spider-Man: No Way Home
```

MOST OF THE MOVIES IN THE DATA RECIEVED LESS THAN 5 RATINGS, AND VERY FEW MOVIES HAVE MANY RATINGS, ALTHOUGH THE MOST RATED MOVIE '414906' RECIEVED 18 RATINGS.



TECHNOLOGIES

- pandas: Data manipulation and preprocessing.
- scikit-surprise: Collaborative filtering and evaluation.
- Flask: Web framework for the user interface.
- Chroma DB: Vector database for efficient storage and retrieval of embeddings.
- Sentence-Transformers: Embedding generation for users and movies

GROUP MEMBERS

SHRUTI GARAD - 202201070140
IRA PADOLE - 202201070141
APARNA RAJ - 202201070142
CHAITANYA CHAVHAN- 202201070214