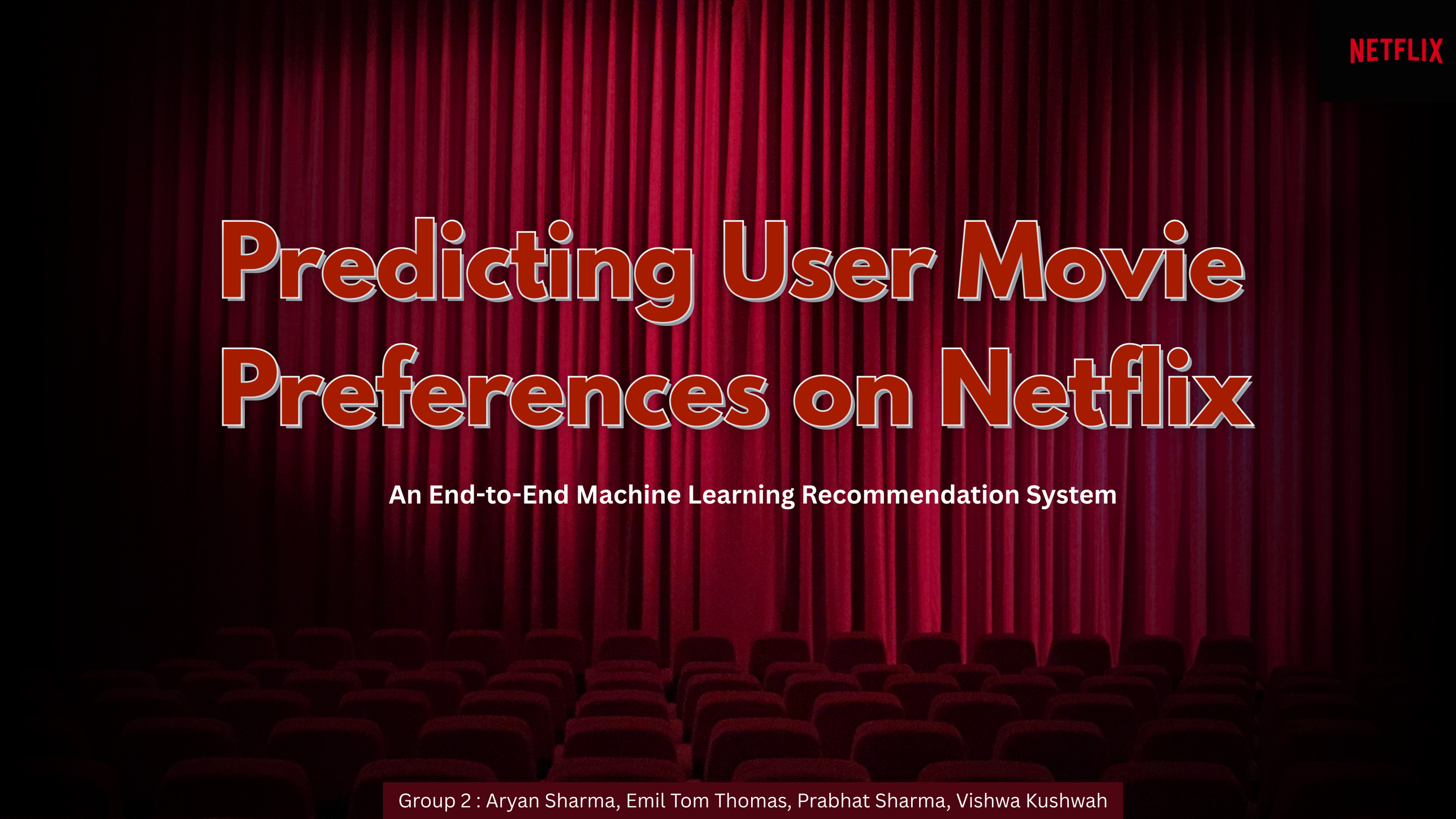


Predicting User Movie Preferences on Netflix

An End-to-End Machine Learning Recommendation System

Group 2 : Aryan Sharma, Emil Tom Thomas, Prabhat Sharma, Vishwa Kushwah

Problem Statement & Project Goal

- Netflix aims to improve its recommendation engine's accuracy.
- The challenge is predicting a user's binary preference: Like or Dislike for an unrated movie/show.

- Build a robust Supervised Classification Model to predict user preference.
- Leverage combined viewing history, demographics, and movie metadata.
- Data Source: MovieLens 20M Dataset (Ratings, Movies, Tags).

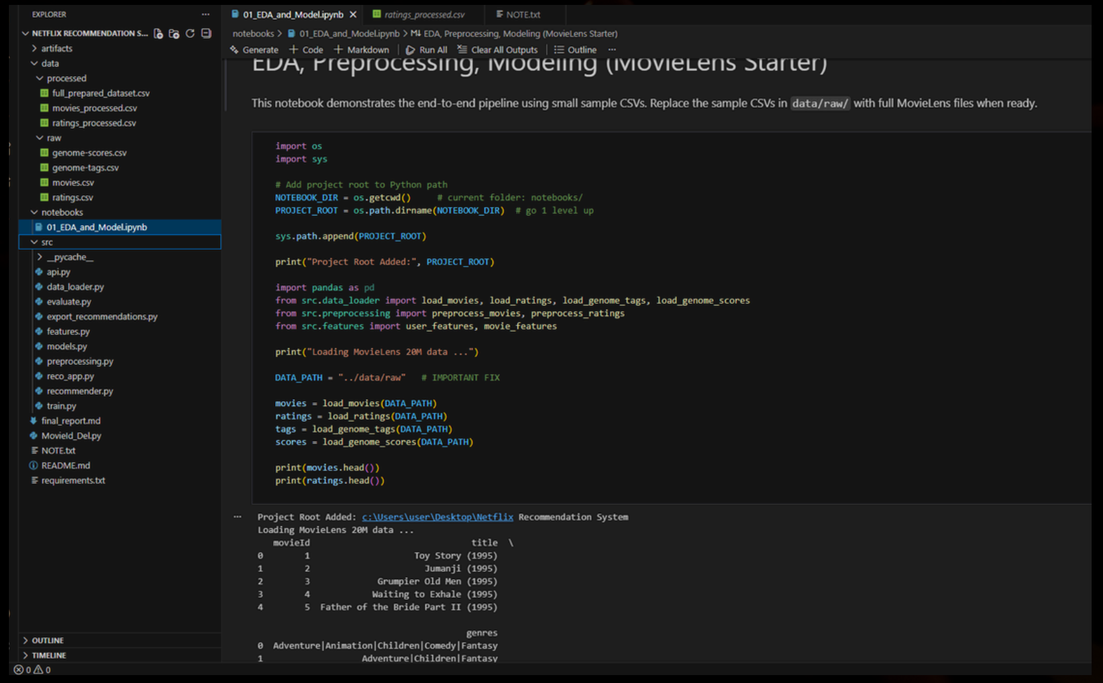
Data Preparation (Preprocessing)

Target Labeling:
Transformed continuous ratings into a binary target {Like = Rating (3.5 / 4.0)}

Missing Values:
Handled missing data via imputation and feature aggregation.

Encoding:
Applied One-Hot Encoding (MultiLabelBinarizer) to Movie Genres.

Normalization:
Scaled numerical features (timestamps, release year, rating aggregates) using StandardScaler .




```
Project Root Added: c:\Users\user\Desktop\Netflix Recommendation System
Loading MovieLens 20M data ...
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                        Comedy|Drama|Romance
4                                       Comedy
   userId  movieId  rating   timestamp
0       1        2     3.5  1112486027
1       1       29     3.5  1112484676
2       1       32     3.5  1112484819
3       1       47     3.5  1112484727
4       1       50     3.5  1112484580
```

Feature Engineering (User-Level)

Objective: Characterize the user's viewing habits and personal taste.

Feature Type	Specific Features Implemented	Purpose
User Profile	Avg User Rating, Rating Count, Rating Std	Measures user engagement, sentiment, and consistency.
Temporal	Most Watched Hour	Captures preferred time of day for watching content.
Interaction	User-Genre Preferences (Average rating per genre)	Most critical feature: Measures a user's strength of liking for each movie genre.

Feature Engineering (Movie-Level)

Feature	*TypeSpecific Features*	*Implemented Purpose*
Quality/Popularity	Avg Movie Rating, Movie Rating Count	Movie Rating Count for overall quality and widespread appeal.
Metadata	Release Year, Genre One-Hots	Provides contextual data for the models
Interaction	Implicitly captured by the model combining User-Genre Preference with Movie Genre One-Hots	

Validation Strategy

Data Split:

- Train/Validation/Test Split used in pipeline for rigorous evaluation.
- Stratified sampling applied to ensure balanced representation of Like/Dislike labels.

Critical Leakage Prevention:

- Ensured Test Set Users in Training Set.
- Rationale: User-level features (e.g., Avg Rating) require a user's past history to be calculated. If a user is entirely new (cold-start), these features cannot be computed, leading to artificially inflated results.

Model Building

Model	Type	Rationale
Logistic Regression (LR)	Linear Baseline	Fast, interpretable, establishes minimum performance benchmark.
Random Forest (RF)	Ensemble / Bagging	Handles non-linearity and feature interactions well.
XGBoost Classifier	Ensemble / Gradient Boosting	State-of-the-art for structured data; highly optimized for speed and performance.

src > models.py > train_random_forest

```python
import joblib
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import xgboost as xgb

def train_logistic(X, y, **kwargs):
    clf = LogisticRegression(max_iter=1000, **kwargs)
    clf.fit(X, y)
    return clf

def train_random_forest(X, y, **kwargs):
    clf = RandomForestClassifier(n_estimators=100, **kwargs)
    clf.fit(X, y)
    return clf

def train_xgboost(X, y, **kwargs):
    model = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss', **kwargs)
    model.fit(X, y)
    return model

def save_model(model, path):
    joblib.dump(model, path)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\e430543\Downloads\Netflix Recommendation System\Netflix Recommendation System> python src/train.py
 - user_mean_rating
 - user_rating_count
 - user_rating_std
 - most_watched_hour
 - movie_mean_rating
 - movie_rating_count
 - popularity_score
 - trending_flag
 - year
 - pref_Action
 - pref_Adventure
 - pref_Animation
 - pref_Children
 - pref_Comedy
 - pref_Crime
 - pref_Documentary
 - pref_Drama
 - pref_Fantasy
 - pref_Film-Noir
 - pref_Horror
 - pref_IMAX
 - pref_Musical
 - pref_Mystery
 - pref_Romance
 - pref_Sci-Fi
```

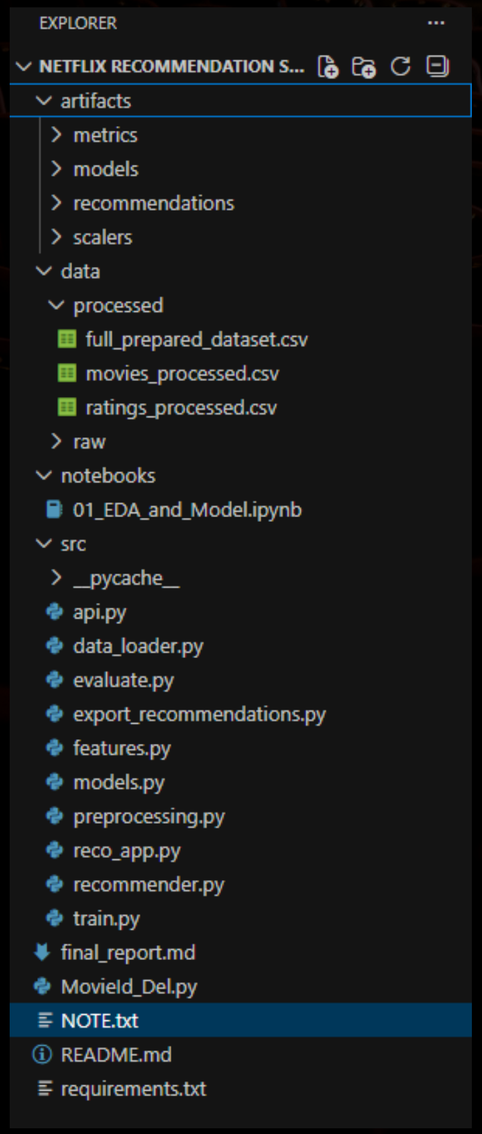
Results: Model Comparison

	accuracy / test			accuracy / val			f1 / test			f1 / val			precision / test			precision / val			recall / test			recall / val			roc_auc / test	
model	baseline	grid	random	baseline	grid	random	baseline	grid	random	baseline	grid	random	baseline	grid	random	baseline	grid	random	baseline	grid	random	baseline	grid	random	baseline	grid
logistic_regression	0.778	None	None	0.764	None	None	0.816	None	None	0.809	None	None	0.79	None	None	0.768	None	None	0.845	None	None	0.853	None	None	0.871	None
logistic_regression_tuned	None	0.778	None	None	0.766	None	None	0.816	None	None	0.81	None	None	0.789	None	None	0.769	None	None	0.845	None	None	0.855	None	None	0.871
random_forest	0.76	None	None	0.741	None	None	0.8	None	None	0.79	None	None	0.778	None	None	0.751	None	None	0.824	None	None	0.832	None	None	0.854	None
random_forest_tuned	None	None	0.774	None	None	0.771	None	None	0.813	None	None	0.816	None	None	0.788	None	None	0.768	None	None	0.839	None	None	0.87	None	None
xgboost	0.757	None	None	0.767	None	None	0.798	None	None	0.808	None	None	0.774	None	None	0.776	None	None	0.823	None	None	0.844	None	None	0.856	None
xgboost_tuned	None	None	0.777	None	None	0.77	None	None	0.814	None	None	0.814	None	None	0.792	None	None	0.77	None	None	0.838	None	None	0.863	None	None

- **Accuracy**: Overall correctness.
- **Precision & Recall**: Critical for recommendation systems.
- **High Precision**: Avoiding suggestions the user will dislike (reduces user frustration).
- **High Recall**: Ensuring all movies the user would like are suggested (increases coverage).

Folder Structure

Folder	Content	Purpose
src/	Core Python Modules (train.py, features.py, models.py)	Contains all the custom logic for the ML pipeline (preprocessing, feature engineering, training, and recommendation).
data/	raw/, processed/	Storage for the raw MovieLens data and the prepared (joined) feature datasets.
artifacts/	models/, scalers/, recommendations/	ML Outputs: Stores the trained models (XGBoost.pkl), the fitted StandardScaler, and recommendation exports.
Root Level	reco_app.py, api.py	Deployment & Service: Files for the Streamlit web application and the Flask API endpoint for real-time recommendations.



Deployment & Recommendation App

Netflix Recommendation ML Pipeline

Builds an end-to-end ML system with:

- Data cleaning & feature engineering
- User profiles & rating behaviour
- NLP-driven content embeddings (TF-IDF + SVD)
- Multiple models (LogReg, RF, XGBoost/GBM)
- Hyperparameter tuning (GridSearchCV / RandomizedSearchCV)
- Collaborative Filtering (pure NumPy SVD)
- Personalized recommendations & similar content

Data & EDA | Models | Recommendations | Similar Content

Dataset Overview

Users	Movies	Ratings	Avg Rating
83	2821	8999	3.68

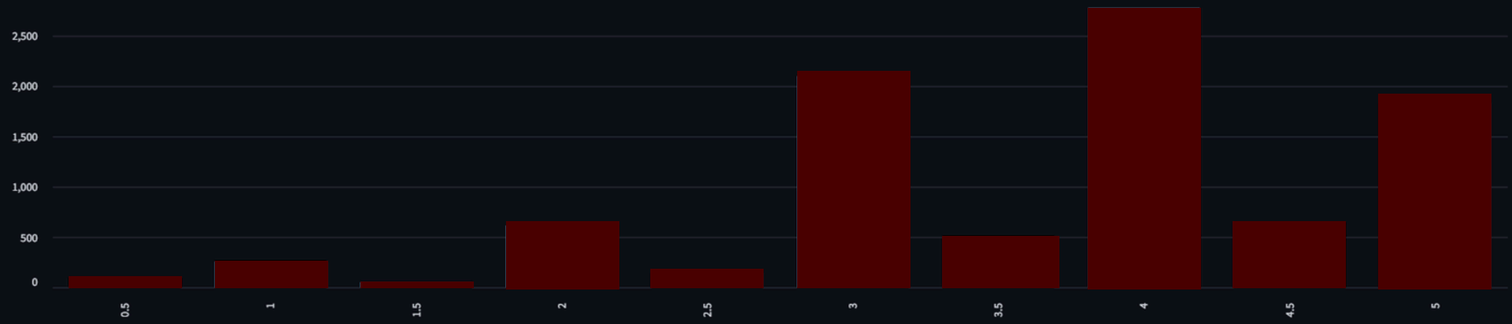
Deliverable: Ready for real-time inference via a web service.

- API: Developed a Flask API (api.py) to serve real-time predictions.
- Front-End: Implemented a Streamlit Web Application (reco_app.py) for demonstration.
- Functionality: The app accepts a User ID and provides Top N Recommended Movies based on the XGBoost prediction score (probability of Liking the movie).

Sample Interactions

	userId	movieId	rating	timestamp	title	genres	tmdb_id	overview	poster_url
0	1	2	3.5	1,112,486,027	Jumanji (1995)	Adventure|Children|Fantasy	8,844	When siblings Judy and Peter discover an enchanted board game that opens the doo	https://ima
1	1	29	3.5	1,112,484,676	City of Lost Children, The (Cité des enfants perdus, La) (1995)	Adventure|Drama|Fantasy|Mystery|Sci-Fi	902	A scientist in a surrealist society kidnaps children to steal their dreams, hoping that tl	https://ima
2	1	32	3.5	1,112,484,819	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)	Mystery|Sci-Fi|Thriller	None		None
3	1	47	3.5	1,112,484,727	Seven (a.k.a. Se7en) (1995)	Mystery|Thriller	None		None
4	1	50	3.5	1,112,484,580	Usual Suspects, The (1995)	Crime|Mystery|Thriller	629	Held in an L.A. interrogation room, Verbal Kint attempts to convince the feds that a m	https://ima

Rating Distribution



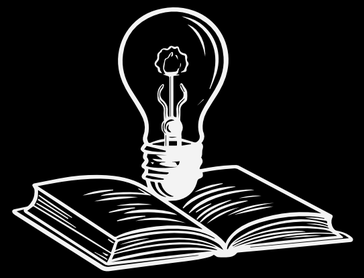
Conclusion & Key Learnings

Final Model Selection:

Radom Forest is the best performing model, achieving a strong Accuracy, Precision & Recall

Key Learnings:

- Feature Engineering is King: The aggregated User/Movie features and the User-Genre Preference feature were the primary drivers of performance, showing a significant jump over the linear baseline.
- Robust Pipeline: Successful implementation of an end-to-end ML pipeline capable of handling large-scale data and complex feature dependencies.



Thank *You*