NETFLIX MOVIE PREDICTION

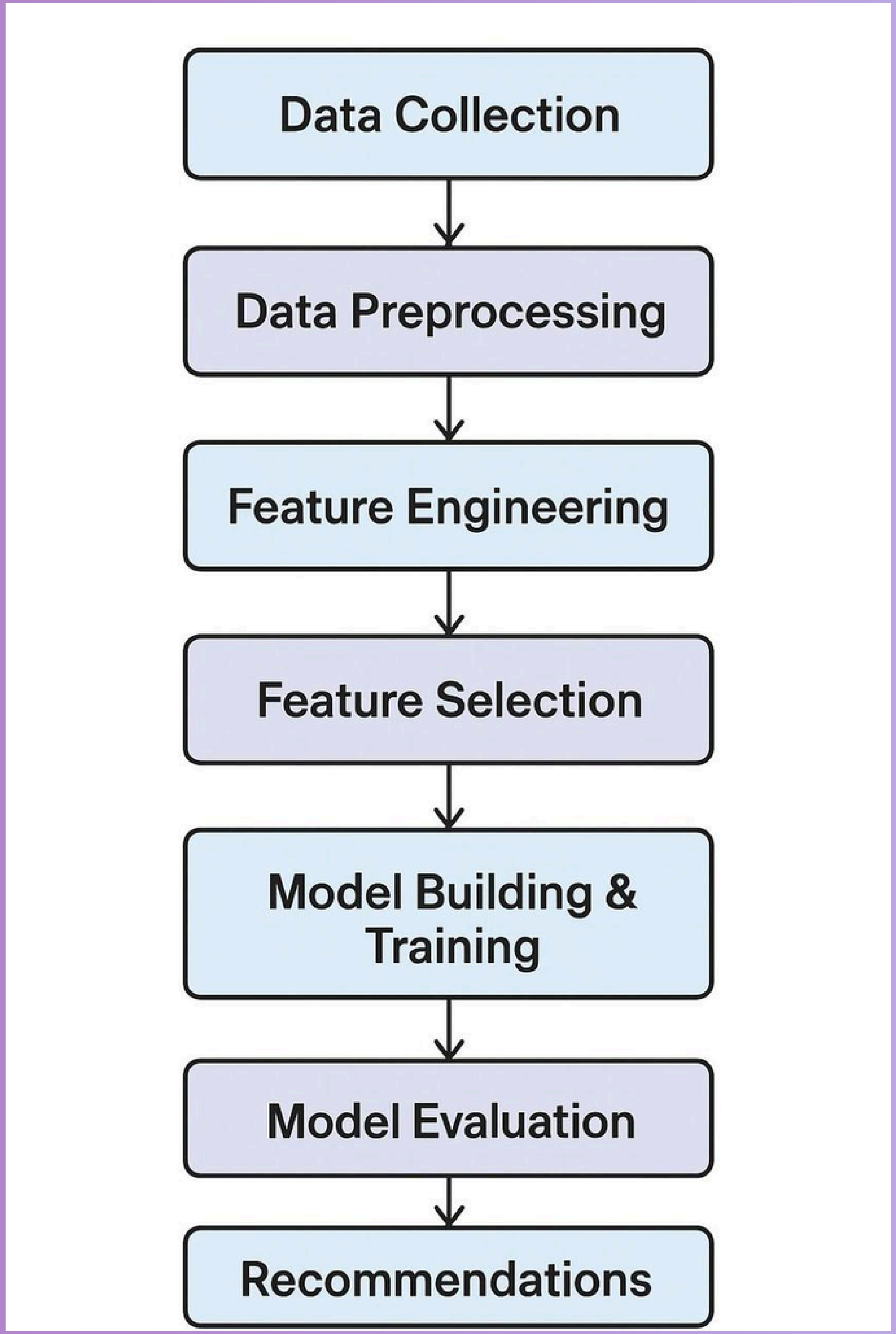
By Group 1:
Ayush Kumar Verma
Soumya Singh
Vashist Tiwari
Vikas Kumar
Vikash Rajput

Introduction

PROBLEM: Netflix wants to improve its recommendation engine by predicting whether a user will like a movie or TV show.

SOLUTION: Developed a machine learning based prediction system that predicts whether a user will like a movie or not using available dataset, feature engineering, and multiple classification models.

Workflow

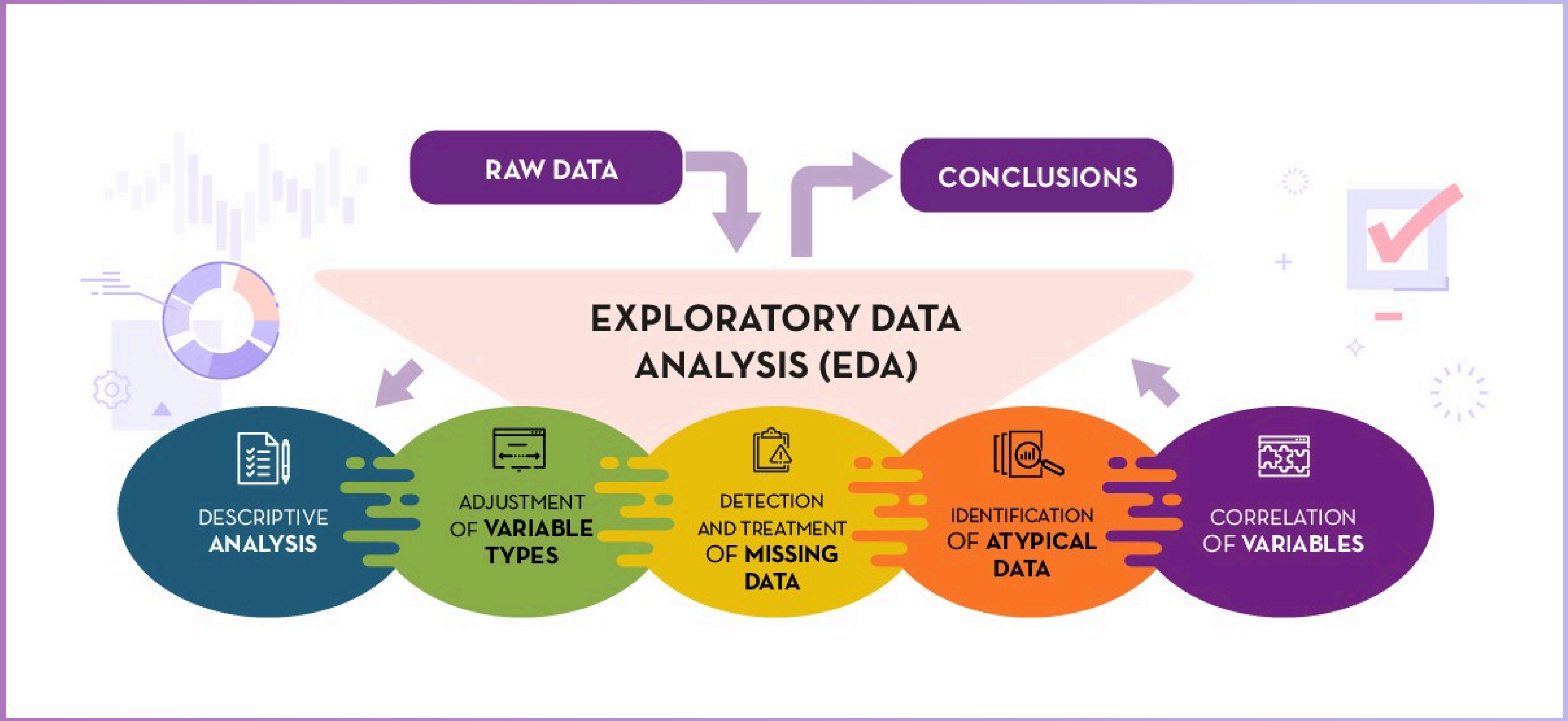

Data Collection
Data Preprocessing
Feature Engineering
Feature Selection
Model Building & Training
Model Evaluation
Recommendations

Dataset

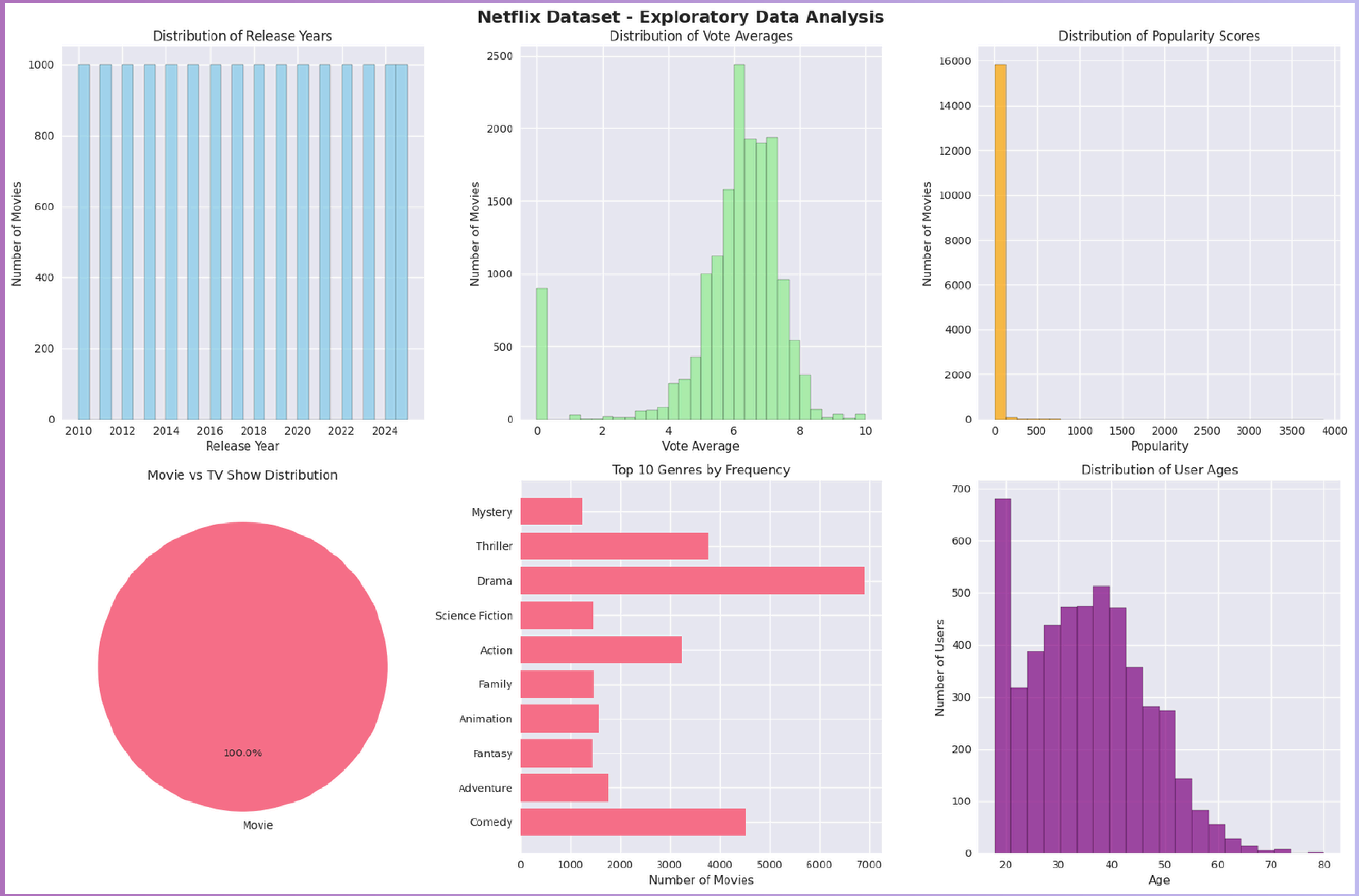
- Movies Dataset (movies.csv)
  - ~16,000 records
  - Columns: title, director, cast, rating, language, genre, etc.
- Users Dataset (users.csv)
  - ~5,000 records
  - Columns: age, gender, country, user_id, etc.
- Interactions Dataset (interactions.csv)
  - ~49,950 records
  - Columns: user_id, show_id, liked, watched_percentage, minute_watched
- Merged Dataset (netflix_merged_data.csv)
  - ~50,000 records
  - Created by combining the three datasets for further processing and model building

Exploratory Data Analysis

- Finding the missing value- count and precentage.
- Finding unique values in each column.
- Count genre frequency: To find top 10 most common genres.


RAW DATA
CONCLUSIONS
EXPLORATORY DATA ANALYSIS (EDA)
DESCRIPTIVE ANALYSIS
ADJUSTMENT OF VARIABLE TYPES
DETECTION AND TREATMENT OF MISSING DATA
IDENTIFICATION OF ATYPICAL DATA
CORRELATION OF VARIABLES

Results of EDA



Preprocessing

Handled missing values:

- Filled missing numerical fields (ratings, popularity, budget, revenue, release year) with median or zero.
- Replaced missing categorical fields (director, cast, country, language, genres, description) with default values.

Duration normalization: Extracted numeric values from duration (e.g., "120 min" → 120).

Feature engineering:

- Binary features: is_movie, has_budget, has_revenue.
- Calculated features: profit, ROI (return on investment), movie_age.
- Popularity ranking using percentile rank.
- Categorized movies into rating categories: Poor, Average, Good, Excellent.

User Data Preprocessing

- Created age groups, watch time categories, and account maturity levels, and calculated user engagement through the number of preferred genres.

Feature Engineering & Dataset

Intro: Aggregated statistics were created for users and movies to capture behavioral patterns and content characteristics.

Dataset Merging: Combined movies, users, and interactions into a single dataset.

User-Level Features: Average rating, rating deviation, total likes, watch time patterns.

Movie-Level Features: Average rating, popularity, like rate, total interactions

Genre Features:

- Genre match score → overlap between user’s preferences and movie genres

Interaction Features:

- Difference between user avg rating & movie rating.
- Popularity vs. user interactions, user selectivity, movie appeal.
- Final Dataset: ~50,000 rows with 98 engineered features for ML modeling.

Data Validation & Train-Test Split

- Target Variable: liked (binary)
  - Distribution: ~25,220 likes, ~24,768 dislikes
  - Like rate: 0.505
- Stratified Split (user-level to avoid cold-start):
  - Training: 32,034 samples, Like rate 0.504
  - Validation: 7,966 samples, Like rate 0.508
  - Test: 9,988 samples, Like rate 0.503
  - Light users all in training; heavy users split across sets
- Top Features (examples):
  - rating_x, minutes_watched, watch_percentage, age, avg_daily_watch_time, budget, revenue, vote_average

Split	Samples	Like Rate
Training	32,034	0.504
Validation	7,966	0.508
Test	9,988	0.503

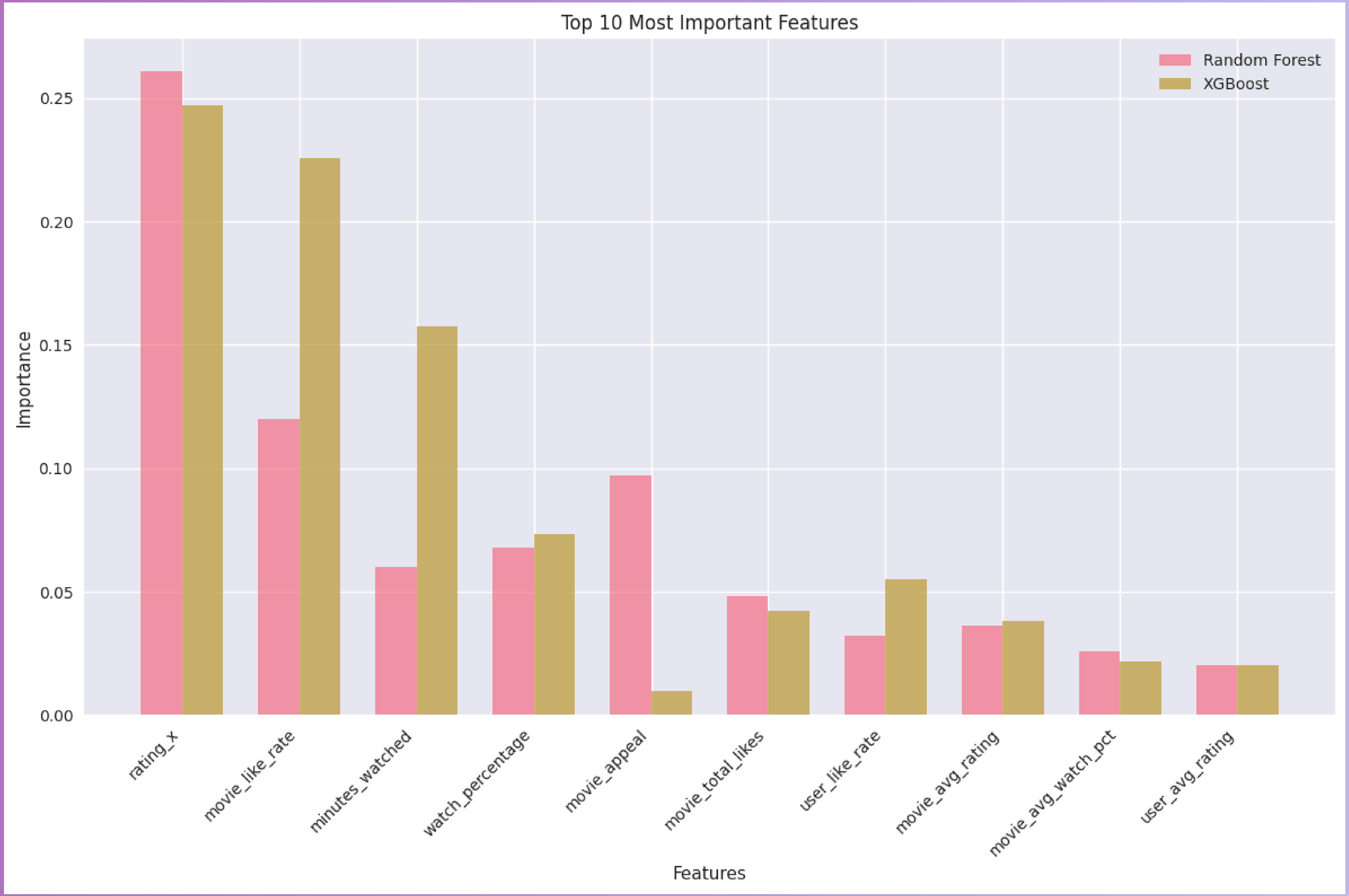
Feature Selection & Preprocessing

- Preprocessing Pipeline:
  - Numerical features (75) scaled using StandardScaler
  - Categorical features (2) encoded using LabelEncoder
  - Preprocessing completed: Feature shape = 32,034 × 77

- Feature Selection Methods:
  - Statistical Selection (SelectKBest) → top 30 features
  - Random Forest Importance → top 30 features based on importance
  - Correlation Filtering → removed highly correlated features (correlation > 0.95), remaining 70 features

- Purpose:
  - Reduce dimensionality, remove redundant features, and retain most predictive features for modeling.

Top 10 Most Important Features



Model Building and Training

Model Building & Training

- Baseline Models Trained:
  - MLP Classifier
  - Random Forest
  - XGBoost

- Feature Sets Used:
  - Statistical (30 features)
  - Random Forest selected (30 features)
  - Correlation-filtered (70 features)

- Observation:
  - XGBoost with statistical feature set performed best on validation data

Model	Feature Set	Val Acc	Val F1	Val AUC
XGBoost	Statistical	0.897	0.898	0.966
MLP Classifier	Statistical	0.8993	0.8996	0.97
Random Forest	Statistical	0.883	0.885	0.963
XGBoost	Correlation-filtered	0.89	0.892	0.963

Hyperparameter Tuning

- Purpose: Improve model performance by tuning key parameters

- Models Tuned:
  - Random Forest
  - XGBoost
  - Neural Network (MLPClassifier)

- Approach:
  - RandomizedSearchCV with cross-validation
  - Train + validation data combined for tuning

- Outcome:
  - Best performing baseline: XGBoost on statistical feature set
  - Hyperparameter tuning improved CV F1 scores and optimized parameters

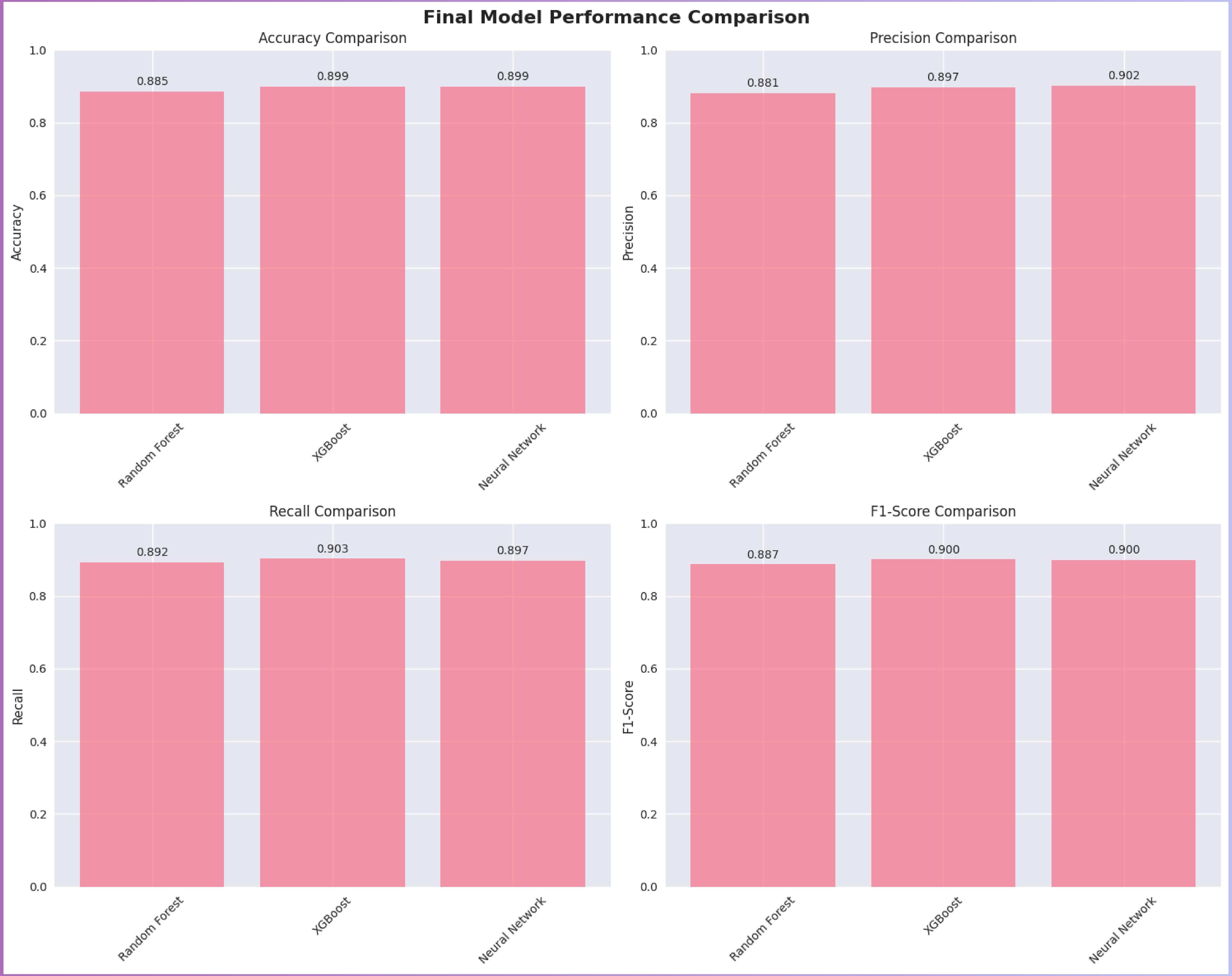
Model Evaluation and Testing

1. Evaluation Setup:

- Evaluated the tuned models on the test set using the best feature set (Statistical features).
- Metrics calculated: Accuracy, Precision, Recall, F1-Score, ROC-AUC.
- Confusion matrices also analyzed for detailed performance.

2. Trade-off:
Increasing precision usually lowers recall and vice versa. In our project, focusing on high precision reduces wrong recommendations but may miss some liked shows, while high recall captures most liked shows but may include unwanted ones

3. Observations from our models:
Our best model, XGBoost, achieves a balanced precision (~0.897) and recall (~0.903), resulting in a high F1-score (~0.900), effectively recommending most liked shows while minimizing bad suggestions.


Final Model Performance Comparison
Accuracy Comparison
Accuracy
0.885
0.899
0.899
Precision Comparison
Precision
0.881
0.897
0.902
Recall Comparison
Recall
0.892
0.903
0.897
F1-Score Comparison
F1-Score
0.887
0.900
0.900
Random Forest
XGBoost
Neural Network
0.0
0.2
0.4
0.6
0.8
1.0

Conclusion

The project successfully developed a personalized recommendation system to predict user preferences. Through thorough data preprocessing, feature selection, and model evaluation, XGBoost was identified as the best-performing model. It achieved high accuracy and a balanced precision-recall (F1 ≈ 0.900), effectively recommending shows users are likely to enjoy. The system ensures high user engagement by suggesting relevant content while minimizing irrelevant recommendations, demonstrating the effectiveness of combining machine learning techniques with user behavior data.

THANK YOU