MOVIE RECOMMENDATION SYSTEM

A machine learning-based system that recommends movies based on user preferences and content similarity.

1. OVERVIEW

The Movie Recommendation System suggests movies based on user preferences using a combination of content-based and collaborative filtering techniques. It utilizes movie metadata such as genres, keywords, and ratings to provide personalized recommendations.

2. DATASET DESCRIPTION

The dataset includes information about movies, such as:
- Title
- Genres
- Cast and Crew
- Keywords
- Average Ratings
- Number of Ratings

This data is cleaned and preprocessed before model training.

3. TECHNOLOGIES USED

The following libraries and technologies were used:
- Python
- Pandas, NumPy (Data Handling)
- Scikit-learn, Surprise (Machine Learning)
- Flask (Web Framework)
- TF-IDF Vectorization & Cosine Similarity (Content-Based Filtering)
- Singular Value Decomposition (Collaborative Filtering)

4. DATA PREPROCESSING

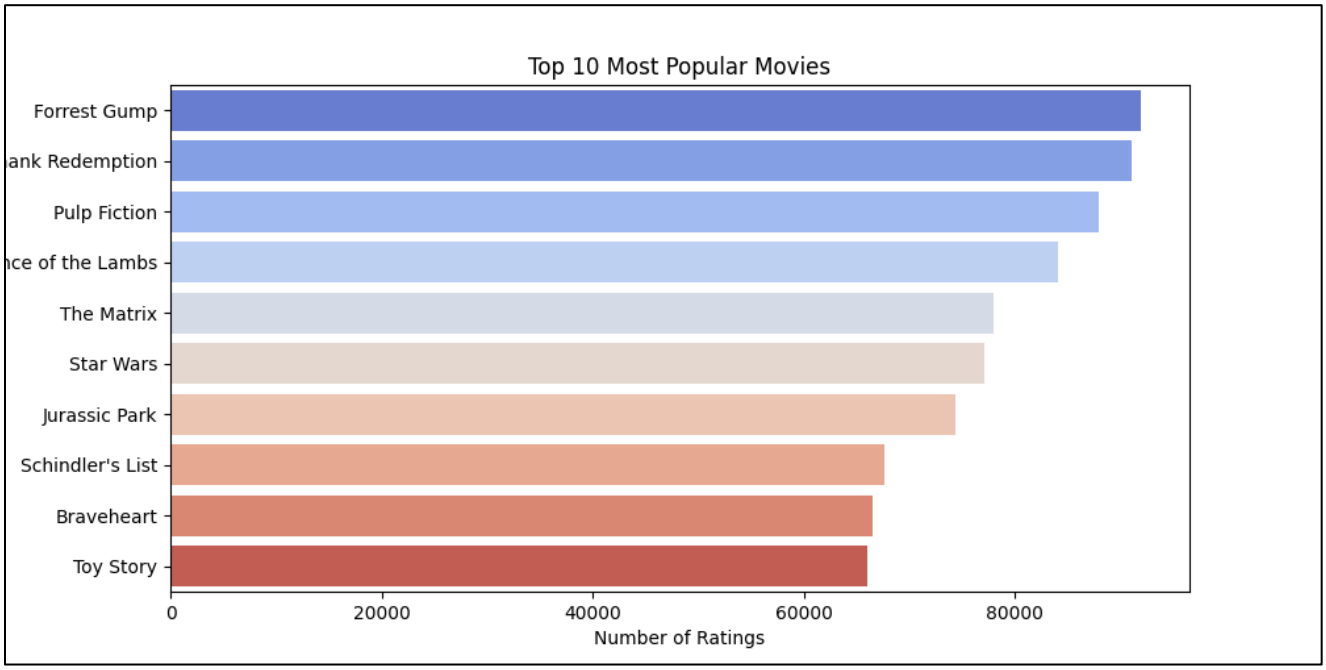
The dataset is preprocessed by handling missing values, extracting relevant features, and merging datasets.

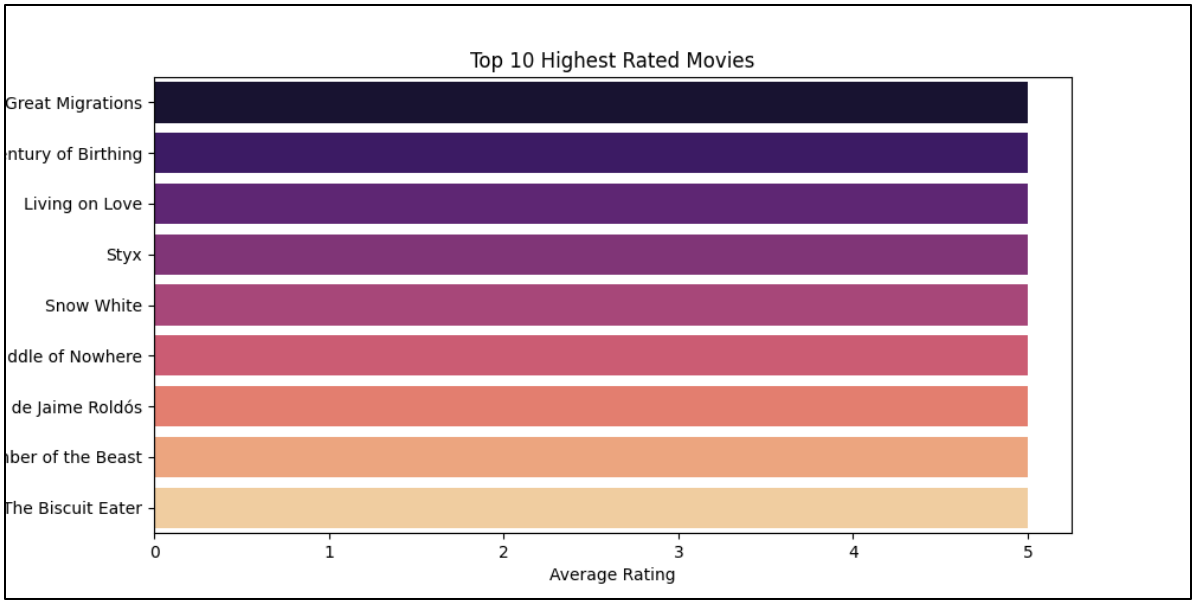
```
import pandas as pd
import numpy as np
import ast
```

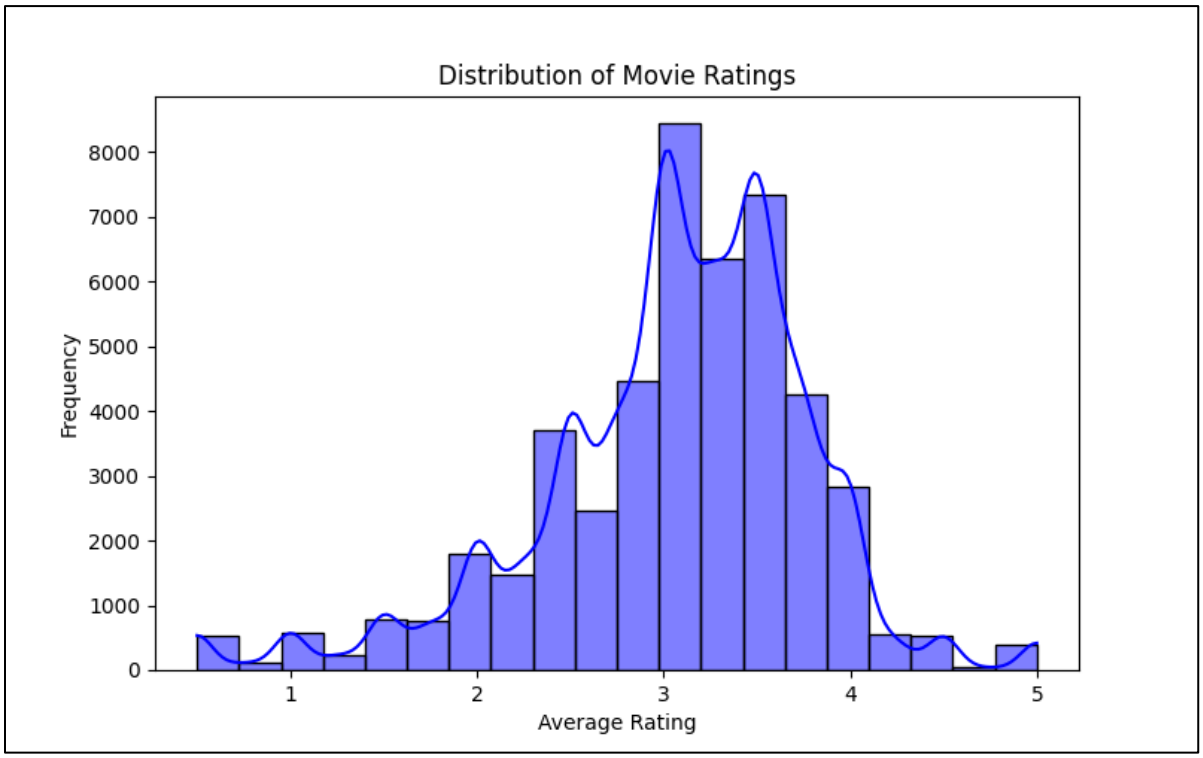
```
# Load datasets
movies = pd.read_csv("movies_metadata.csv", low_memory=False)
credits = pd.read_csv("credits.csv")
keywords = pd.read_csv("keywords.csv")
links = pd.read_csv("links.csv")
ratings = pd.read_csv("ratings.csv")

# Extract and clean data
movies['genres'] = movies['genres'].fillna('[]').apply(lambda x: [d['name'] for d in
ast.literal_eval(x)] if isinstance(x, str) else [])
ratings = ratings.groupby('movieId').agg({'rating': ['mean', 'count']}).reset_index()

# Merge datasets
movies = movies.merge(ratings, on='movieId', how='left')
movies.to_csv("processed_movies.csv", index=False)
```

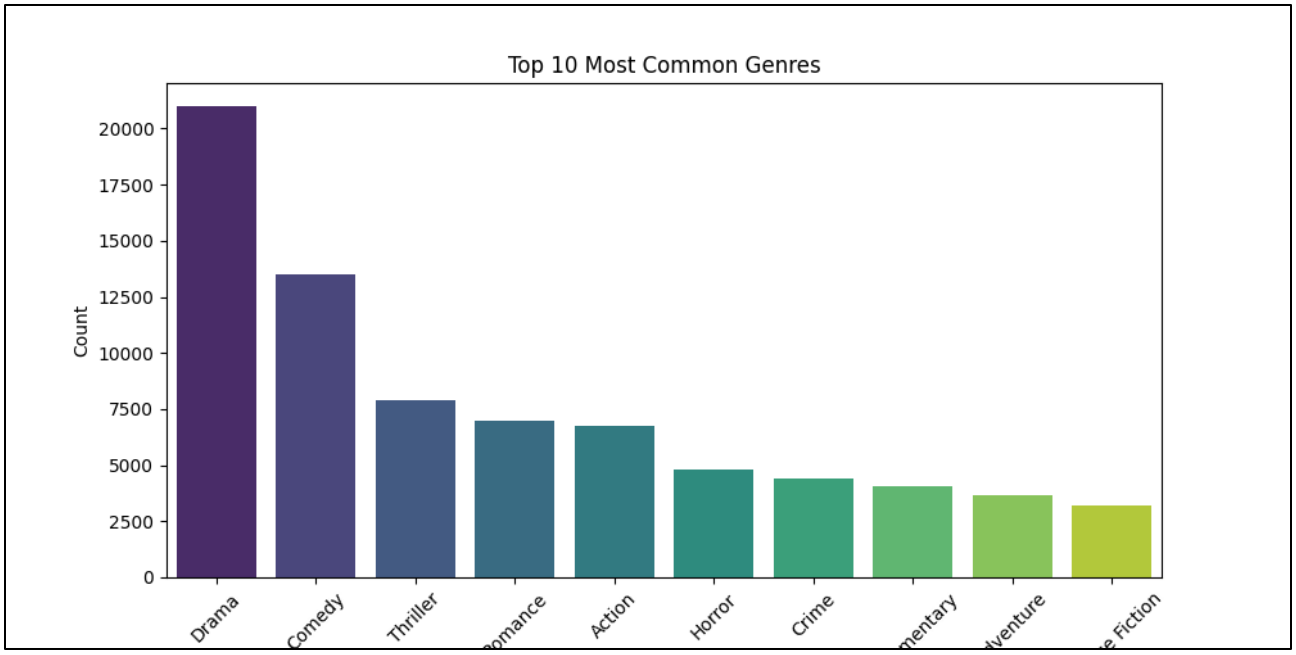
5. EXPLORATORY DATA ANALYSIS(EDA)

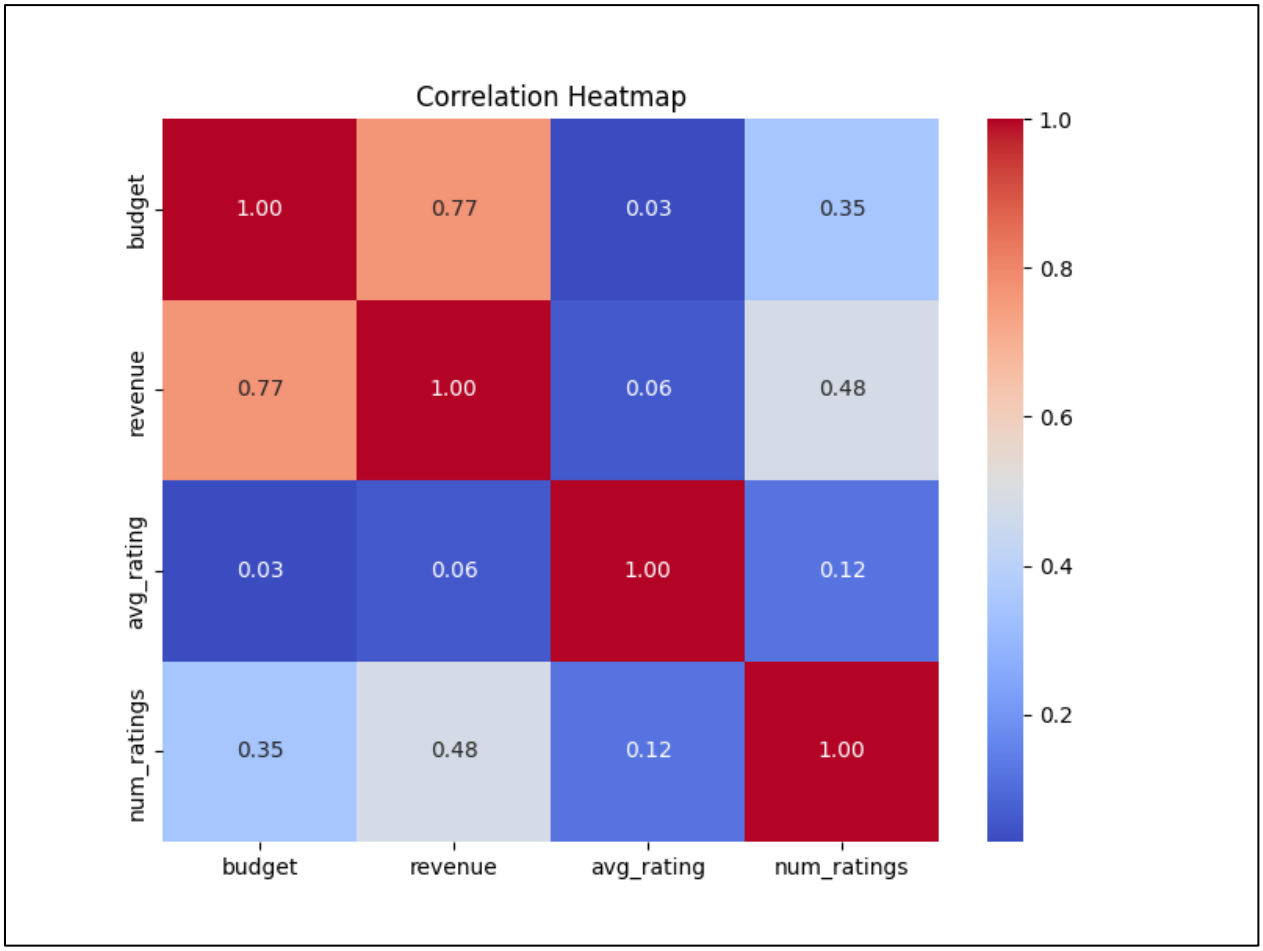
EDA was performed to analyze movie ratings, popular genres, and correlation between budget, revenue, and ratings.




Top 10 Highest Rated Movies
Great Migrations
ntury of Birthing
Living on Love
Styx
Snow White
ddle of Nowhere
de Jaime Roldós
ber of the Beast
The Biscuit Eater
0
1
2
3
4
5
Average Rating


Distribution of Movie Ratings
Frequency
8000
7000
6000
5000
4000
3000
2000
1000
0
1
2
3
4
5
Average Rating


Top 10 Most Common Genres
Count
20000
17500
15000
12500
10000
7500
5000
2500
0
Drama
Comedy
Thriller
Action
Horror
Crime


Correlation Heatmap
budget
revenue
avg_rating
num_ratings
1.00
0.77
0.03
0.35
0.77
1.00
0.06
0.48
0.03
0.06
1.00
0.12
0.35
0.48
0.12
1.00
budget
revenue
avg_rating
num_ratings
1.0
0.8
0.6
0.4
0.2

6. MODEL TRAINING

The system uses both content-based filtering (TF-IDF + Cosine Similarity) and collaborative filtering (SVD).

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from surprise import SVD
from surprise import Dataset, Reader
import pickle

# Load dataset
movies = pd.read_csv("processed_movies.csv")

# Content-based filtering
tfidf = TfidfVectorizer(stop_words="english")
tfidf_matrix = tfidf.fit_transform(movies["genres"] + " " + movies["keywords"])
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Collaborative filtering
reader = Reader(rating_scale=(0.5, 5.0))
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
svd = SVD()
trainset = data.build_full_trainset()
svd.fit(trainset)

# Save models
with open("movie_recommendation_model.pkl", "wb") as file:
    pickle.dump({"movies": movies, "tfidf_matrix": tfidf_matrix, "svd": svd}, file)
```

7. MODEL EVALUATION

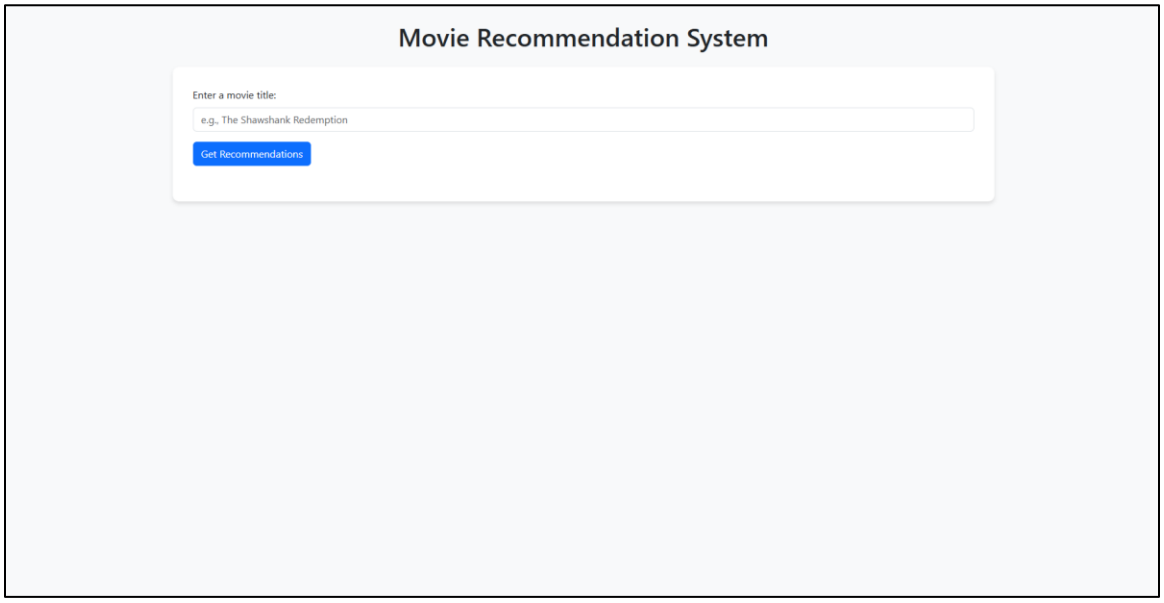
The model was evaluated using Root Mean Square Error (RMSE) for collaborative filtering and precision-recall metrics for content-based filtering.

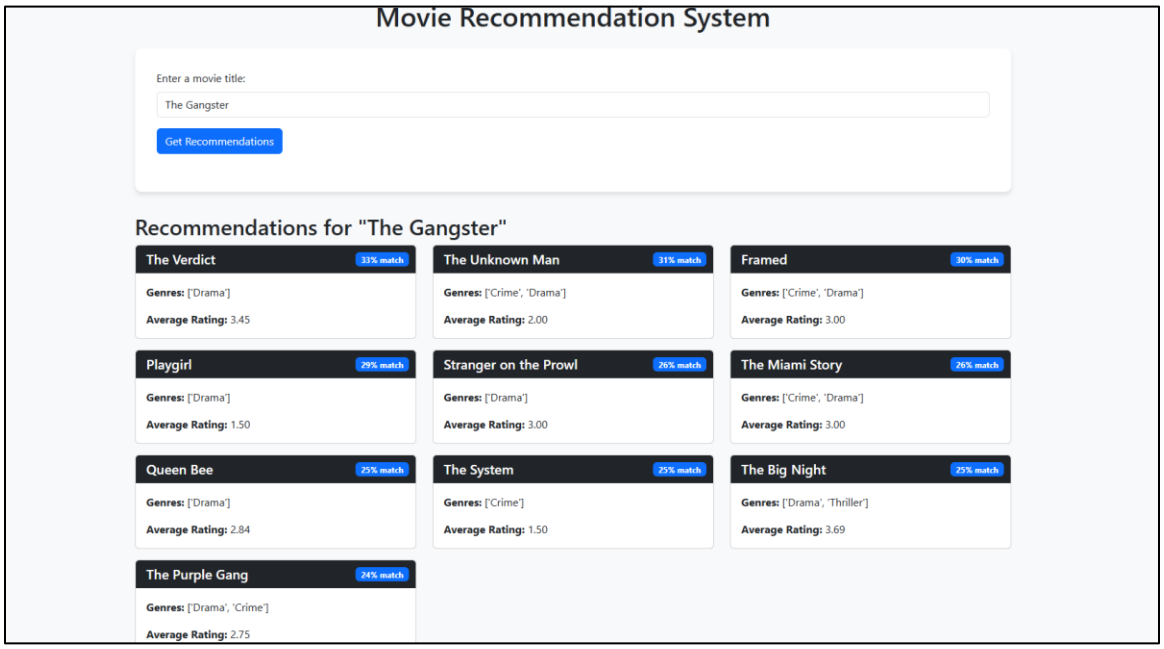
8. USER INTERFACE & FEATURES

A Flask-based web application allows users to input a movie title and receive recommendations. The system also suggests similar titles in case of incorrect inputs.

```
from flask import Flask, request, jsonify
import pickle
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity

app = Flask(__name__)

# Load model
with open("movie_recommendation_model.pkl", "rb") as file:
    model_data = pickle.load(file)

movies = model_data["movies"]
tfidf_matrix = model_data["tfidf_matrix"]

@app.route('/recommend', methods=['POST'])
def recommend():
    movie_title = request.form.get('movie_title')
    movie_idx = movies[movies['title'].str.lower() == movie_title.lower()].index[0]
    sim_scores = cosine_similarity(tfidf_matrix[movie_idx], tfidf_matrix).flatten()
    top_indices = sim_scores.argsort()[-10:][::-1]
    recommendations = movies.iloc[top_indices][['title', 'genres',
'avg_rating']].to_dict(orient='records')
    return jsonify(recommendations)

if __name__ == '__main__':
    app.run(debug=True)
```

9. OUTPUT





10. DEPLOYMENT

The application is deployed using Flask and can be hosted on cloud platforms like Heroku or AWS. The model is loaded into memory for fast predictions.

11. APPLICATIONS & IMPLICATIONS

The Movie Recommendation System has wide-ranging applications across different industries:

• Entertainment Platforms– Streaming services like Netflix, Amazon Prime, and Disney+ can integrate this system to enhance user experience by suggesting personalized content.
• E-commerce– Online retailers selling movies and DVDs can use recommendation models to suggest relevant products based on customer preferences.
• Education – Online learning platforms can adapt this recommendation approach to suggest relevant courses, books, or research papers.
• Marketing & Advertising – Businesses can use recommendation systems to personalize advertisements and target specific audiences.
• Social Media– Platforms like YouTube and TikTok can implement similar models to recommend videos or content based on user history and interests.

12. CONCLUSION

The Movie Recommendation System effectively showcases how machine learning and artificial intelligence can enhance user experience by providing personalized movie suggestions. By leveraging content-based filtering using TF-IDF and cosine similarity, as well as collaborative filtering with Singular Value Decomposition (SVD), this system ensures relevant and diverse recommendations.

This project demonstrates the end-to-end machine learning pipeline, from data preprocessing and EDA to model training, evaluation, and deployment. The **Flask-based web application ensures an interactive and user-friendly experience.

13. FUTURE SCOPE

• Improve accuracy by incorporating hybrid filtering techniques.
• Add personalized recommendations based on user preferences.
• Deploy as a scalable API for integration with other platforms.