🎬 Detailed Report: Content-Based Movie Recommendation System (TF-IDF)

# 1. Introduction

This report documents the step-by-step process of building a personalized content-based recommendation system using the MovieLens 100k dataset. The focus is on User 196, and we aim to identify movies they are most likely to enjoy based on past preferences.  
  
Two approaches were used:  
1. Genre-based filtering using binary vectors  
2. TF-IDF-based filtering using simulated movie 'tags'  
  
The project includes filtering user ratings, generating movie feature vectors, computing similarity scores, visualizing genre preferences, and exporting recommendations.

# 2. Step-by-Step Process

## Step 1: Load Data

We loaded movie metadata from `u.item`, which includes movie IDs, titles, and 19 binary genre indicators. User rating data was loaded from `u.data`, containing user IDs, movie IDs, and rating values.  
  
We also confirmed the structure of the data and ensured alignment by matching movie IDs during joins.

## Step 2: Filter High-Rated Movies for User 196

We extracted ratings given by User 196 and filtered only those rated 4 or higher. This helped us identify movies the user enjoyed, as a reliable basis for generating recommendations.

## Step 3: Simulate Tags for TF-IDF

As the dataset does not contain real plot descriptions or tag data, we simulated 'tags' by combining the movie title and its associated genres. For example, 'Apollo 13 | Drama Sci-Fi' would become the tag text.  
  
This simulated text is rich enough to represent basic movie content.

## Step 4: TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) was applied on the tags column to convert text data into numerical vectors. This method helps downweight common terms and highlight more unique words, giving a better feature space for comparison.

## Step 5: Build User Profile Vector

We created the user profile vector by averaging the TF-IDF vectors of all highly-rated movies. This profile represents what kind of movie content (in terms of genres/titles) the user prefers.

## Step 6: Cosine Similarity & Ranking

Cosine similarity was computed between the user profile and all other movie vectors in the dataset. Movies the user had already rated were excluded, and the top 10 most similar movies were selected.

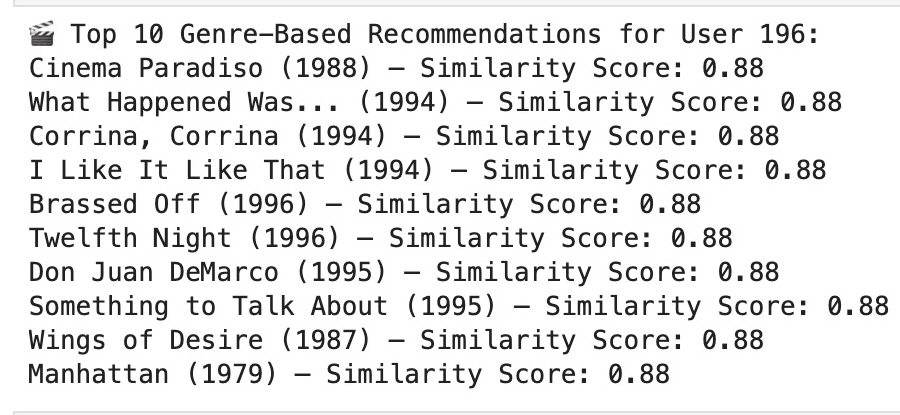
## Step 7: Export Recommendations

The final recommendations were exported to a CSV file at the following path:  
`/Users/anirudhravipudi/Desktop/AI/Practice/User196\_Top10\_TFIDF\_Recommendations.csv`  
  
This file can be used for display in a web interface or downstream pipeline.

# 3. Outputs & Interpretation

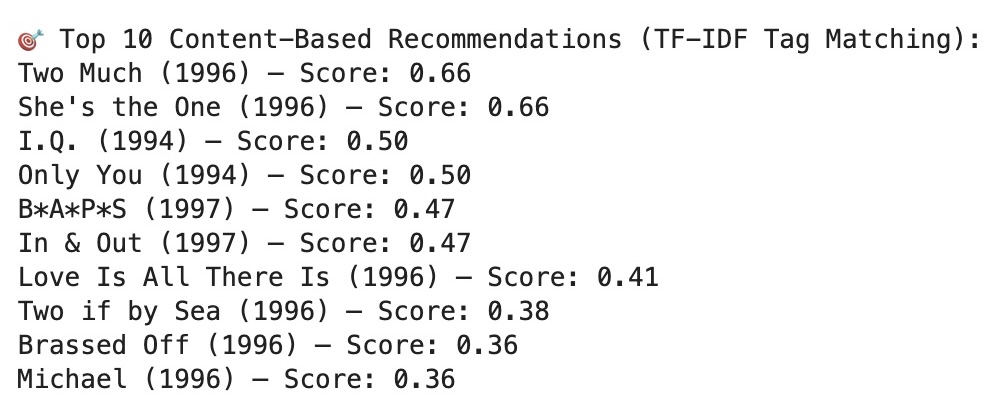
## 📽️ Top 10 Genre-Based Recommendations

The genre-based model returned 10 movies, all with identical similarity scores of 0.88. This shows the limitations of using binary genre indicators — they are too simplistic and treat all genre overlaps equally, resulting in non-diverse and flat recommendations.



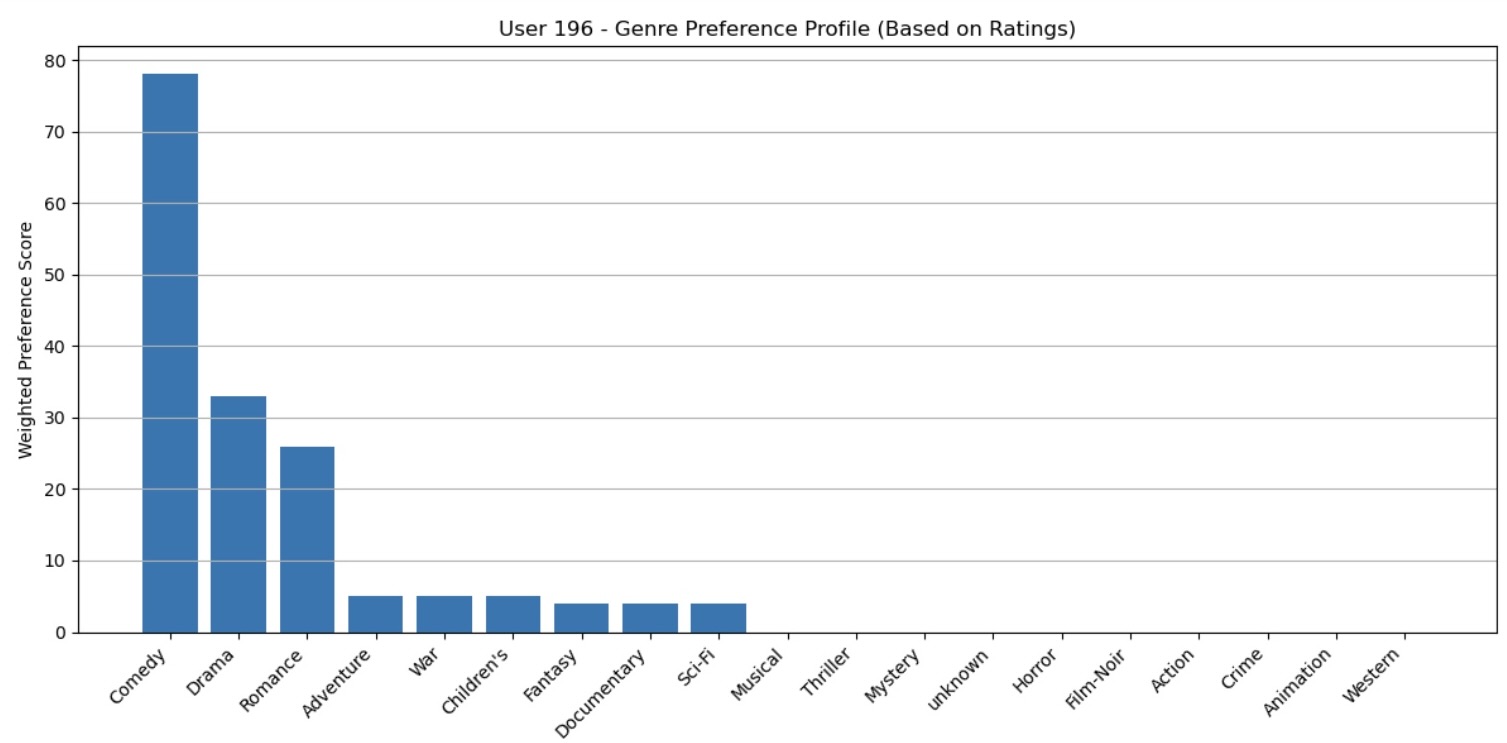
## 🎯 Top 10 TF-IDF Tag-Based Recommendations

This method provided more meaningful differentiation. Scores varied across recommendations, showing it could capture finer nuances in user preference. Movies like 'Two Much (1996)' and 'She's the One (1996)' were most similar to the user profile.  
  
The lowest-ranked movie had a score of 0.36, showing clear ranking separation.



## 📊 User 196 Genre Preference Profile

We created a bar chart to visualize the genre-wise preference profile of User 196. The bars were weighted by how highly the user rated movies in that genre.  
  
The chart reveals strong preferences for:  
- Comedy  
- Drama  
- Romance  
  
Other genres such as Adventure, Sci-Fi, and War had minor weights, while genres like Western and Crime were negligible.



# 4. Conclusion & Next Steps

This recommender system demonstrates how TF-IDF-based content filtering can provide nuanced and diverse recommendations compared to binary genre filtering. We successfully captured User 196's preferences and generated interpretable recommendations.  
  
For future improvements:  
- Integrate actual plot descriptions from TMDb or IMDb APIs  
- Incorporate actor/director metadata  
- Build a hybrid system that combines content-based and collaborative filtering  
- Deploy using Flask with interactive search & rating input