Final Report
Afeya Jahin
Spring 2024

Enhancing Movie Discovery: A Comparative Analysis of Content-Based, Hybrid, and K-Nearest Neighbors approaches in Movie Recommendation Systems

Abstract

This Report evaluates three distinct machine learning approaches implemented in movie recommendation systems: Content-Based Filtering using cosine similarity and weighted averages, Hybrid Collaborative Filtering that leverages Neural Networks and matrix factorization, and K-Nearest Neighbors. Each method aims to improve user engagement by
predicting and recommending movies tailored to individual preferences. Our study employs the Movies Dataset from Kaggle, which includes over 45,000 films with detailed metadata. Findings indicate that the Hybrid Collaborative Filtering model offers the most personalized recommendations by integrating deep learning with collaborative filtering techniques. The K-Nearest Neighbors approach is effective in predicting a movie's popularity, while Content-Based Filtering provides highly accurate recommendations based on movie similarity. We conclude that combining different approaches can maximize the relevance of movie recommendations and significantly enhance user satisfaction.

Introduction

A movie recommendation engine utilizes machine learning techniques to sift through and forecast the movie preferences of users, informed by their historical selections and activities. This sophisticated filtering system estimates the movie options that a user is inclined to favor, focusing on specific interests within the domain—in this case, movies.

The foundational principle of a movie recommendation engine is straightforward. There are two critical components: the users and the movies. The engine curates movie suggestions tailored to individual preferences, with movies serving as the items of interest.

The main objective of these systems is to identify and suggest movies that a user is most likely to enjoy watching. Machine learning algorithms analyze user data from the system's database to forecast future

preferences based on past behavioral data. The significance of such recommendations is underscored by their impact:

At Netflix, two-thirds of the films streamed are based on recommendations.
Google News's recommended articles lead to a 38% increase in user engagement.
Amazon attributes 35% of its sales to its recommendation engine.
Choicestream found that 28% of consumers would purchase more music if they could more easily discover tunes that suit their tastes.
In today's era, where digital platforms offer overwhelming volumes of content, movie recommendation systems have become essential tools for navigating this abundance. Their success directly influences user satisfaction and business outcomes, as evidenced by streaming services like Netflix, where two-thirds of watched films are algorithmically recommended, and Amazon, where 35% of sales come from recommendations. Different machine learning techniques offer unique advantages and limitations in personalizing movie suggestions for each user.

This report explores three primary methodologies for building effective recommendation systems: Content-Based Filtering, Hybrid Collaborative Filtering, and K-Nearest Neighbors. Content-Based Filtering uses cosine similarity and weighted averages to suggest movies that are similar to those the user has already enjoyed. The Hybrid approach combines deep learning with matrix factorization, providing a more accurate recommendation by considering explicit ratings and implicit signals. K-Nearest Neighbors (KNN) predicts whether a previously unseen movie will be popular based on user viewing history, which is particularly useful for identifying likely-watched titles.

The goal of this report is to compare and analyze these methods in their ability to predict user preferences and accurately recommend relevant movies. By leveraging the Movies Dataset from Kaggle, which contains comprehensive metadata on over 45,000 films, we aim to demonstrate how each approach addresses specific recommendation challenges and how combining them could further optimize movie discovery.

Literature review

Content-Based Recommendation Systems

Several studies have explored content-based approaches for developing movie recommendation systems. S.R.S. Reddy et al. emphasized the importance of genre correlation in predicting movies based on user preferences and prior ratings. Their system "aims at recommending movies to users based on the

similarity of genres". Similarly, N. Pradeep et al. developed a recommendation system that utilized cast, crew, and keywords to tailor recommendations. They argue, "Content-based recommendation systems can recommend movies based on one or a combination of two or more attributes". Both studies indicated the efficiency of content-based filtering, as it effectively analyzes item characteristics to suggest similar titles to those the user already liked.The primary limitations of content-based systems, as highlighted in these papers, are the need for sufficient user data and the potential narrowing of recommendations to overly similar items. A significant challenge here is the lack of diversity, which could lead to a monotonous user experience.

Collaborative Filtering and Hybrid Models

Hybrid recommendation models that blend collaborative filtering with deep learning offer significant improvements over individual methods. The collaborative filtering approach leverages user-item interactions and ratings to uncover patterns among users with similar tastes. P. Campos et al. compared collaborative algorithms like CorNgbr, NNCosNgbr, and PureSVD, concluding that collaborative filtering generally performs well in suggesting personalized recommendations. However, these algorithms are challenged by cold-start problems, scalability issues, and data sparsity, often addressed by incorporating deep learning techniques.

Hybrid models like those incorporating Neural Networks with collaborative filtering can effectively predict user preferences by using both user-specific information and item metadata. In the current project, a hybrid collaborative filtering approach uses matrix factorization along with Neural Networks to predict how a user would rate specific items and to recommend new titles based on genre and historical patterns. Basilico and Hofmann have explored how unified models that integrate both user-item ratings and item attributes can "create more accurate hybrid recommendations".

K-Nearest Neighbors Approach

The K-Nearest Neighbors (KNN) approach offers an alternative to collaborative filtering. It seeks to classify unseen movies based on feature similarity, predicting their popularity by comparing them to existing popular items in the training dataset. According to N. Pradeep et al., KNN achieves significant accuracy in identifying new and popular movies due to its ability to classify unseen titles into binary popularity labels. Nevertheless, its computational cost increases with data size, and it requires well-balanced datasets to prevent biases.

Evaluation of Recommendation Systems

Quality evaluation plays a pivotal role in the adoption of recommendation systems. P. Campos et al. found that while accuracy remains important, other factors like novelty and satisfaction significantly influence the perceived quality of recommendations. They noted, "objective metrics are not always good predictors of the perceived quality of RSs," and balancing novelty with relevance is critical. This aligns with S.R.S. Reddy et al., who recommended incorporating user emotions into the evaluation process to provide more personalized recommendations.

Datasets

I am using the movies dataset from kaggle.
https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
This dataset is a compilation of information gathered from TMDB and GroupLens.
The Movie Details, Credits and Keywords have been collected from the TMDB Open API.
The Movie Links and Ratings have been obtained from the Official GroupLens website.

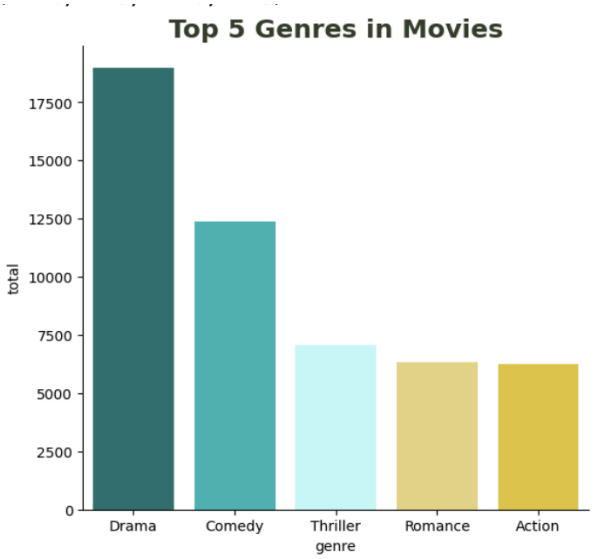
The MovieLens dataset by GroupLens stands as a commonly utilized resource in the creation of recommendation engines. This extensive collection of data was compiled by the GroupLens Research team at the University of Minnesota.

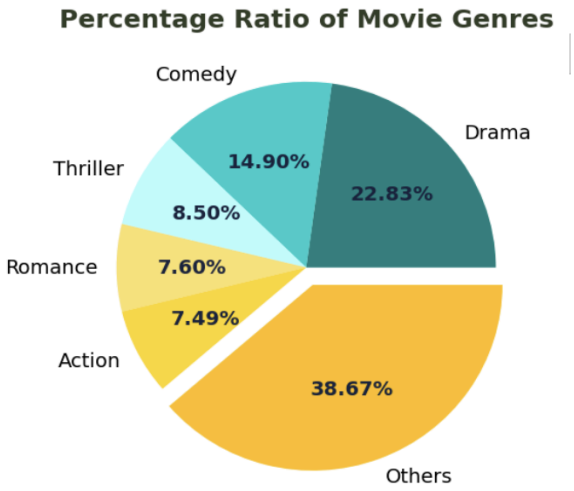
movies_metadata.csv: The main Movies Metadata file. Contains information on 45,000+ movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.
keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.
credits.csv: Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.
ratings_small.csv: The subset of 100,000 ratings from 700 users on 9,000 movies.
Regarding processing and cleaning, the following steps were undertaken:
Data Importation and Initial Cleanup: The movies dataset was trimmed down by removing non-essential columns. Specific rows with corrupt data were removed to ensure consistency in data types, particularly for the 'id' column.

Methodology

1.Exploratory Data Visualization

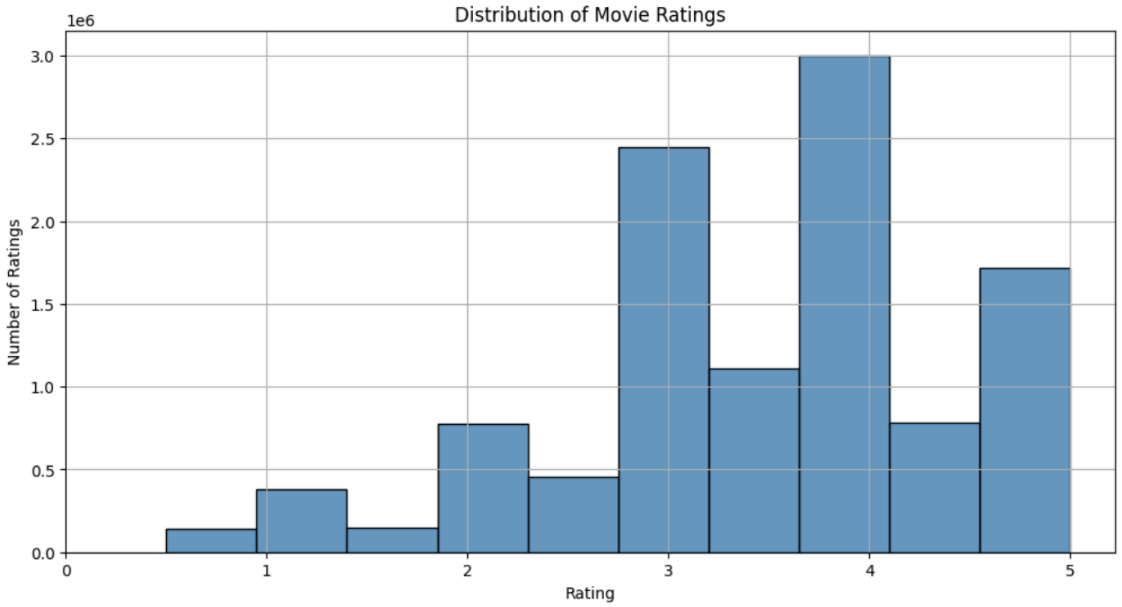
Genre Popularity: Displaying the percentage of movies per genre could help in understanding user preferences and genre popularity





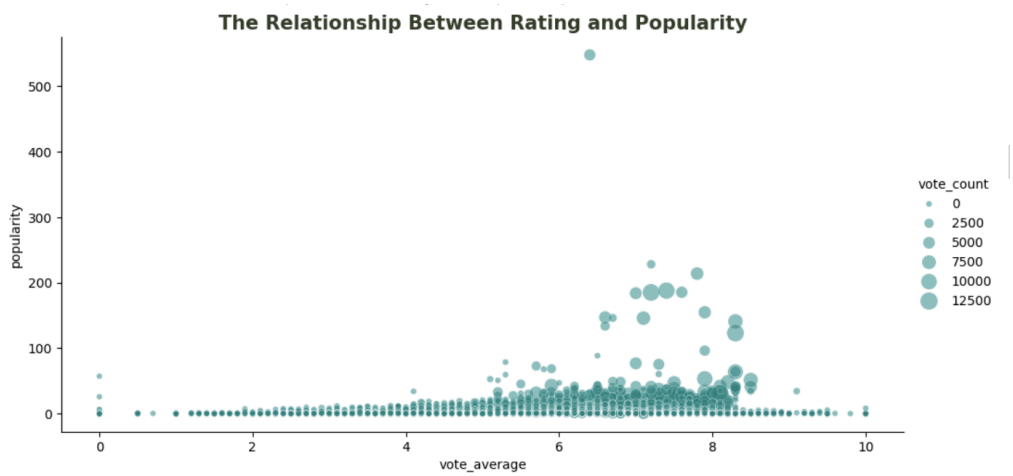
1. Drama is the most dominant genre with over 18000 movies
2. Out of 5 top genres, there are still many genres in the dataset. They hold 38.67% of total genres in the movies

Distribution of Movie Ratings: Shows how users generally rate movies, which can help in understanding the central tendency and dispersion of ratings.



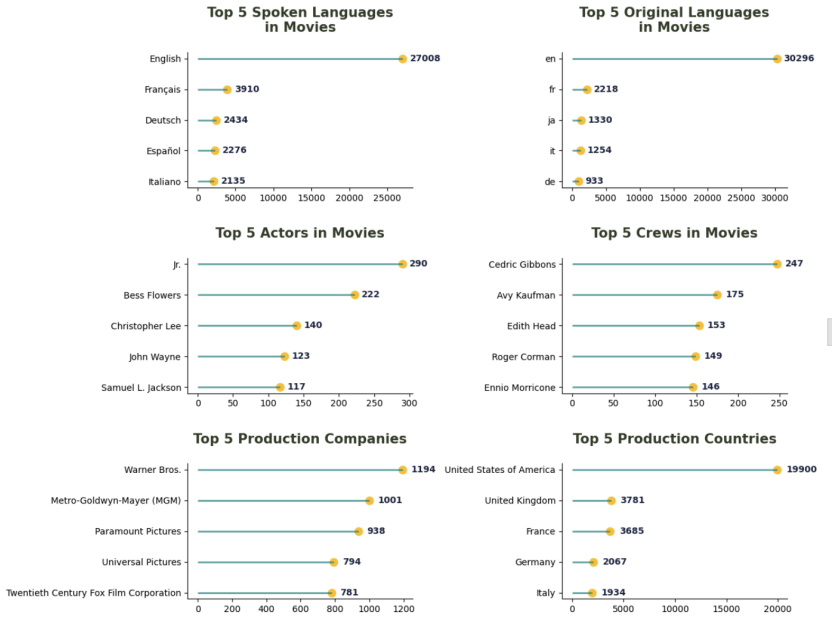
1. Skewness: The distribution is left-skewed, meaning that there are more high ratings than low ratings. Most movies seem to receive ratings of 3.5 and above, with the frequency decreasing as the rating value decreases.
2. Rating Preference: Users tend to rate movies positively. This could be indicative of a tendency to rate movies they like and not rate those they dislike, or it could mean that most of the movies in the dataset are of relatively high quality according to the user base.
3. Data Density: There's a dense concentration of ratings between 3.5 and 4.5. These ratings are more common than the extremes, suggesting that users are more often moderately satisfied with the movies they watch.
4. Implications for Recommendation Systems: For a recommendation system, this skew could imply that it might be challenging to distinguish between highly-rated movies since there are many of them; therefore, other features might be necessary to improve the recommendations, such as user-specific or content-based information.
5. Data Transformation Considerations: The skewness might necessitate data transformation if algorithms sensitive to non-normal distributions are used, as many machine learning models work better with normally distributed data.

The Relationship Between Rating and Popularity



1. Movies that either got rating 0 or 10 are basically caused by small number of voter. As the vote count increase, the rating is most likely around 5 to 8.5
2. It's clear that popular movies will get more vote count as shown from above plot

Top 5 of Movie features



What we can deduce from the visualisation:

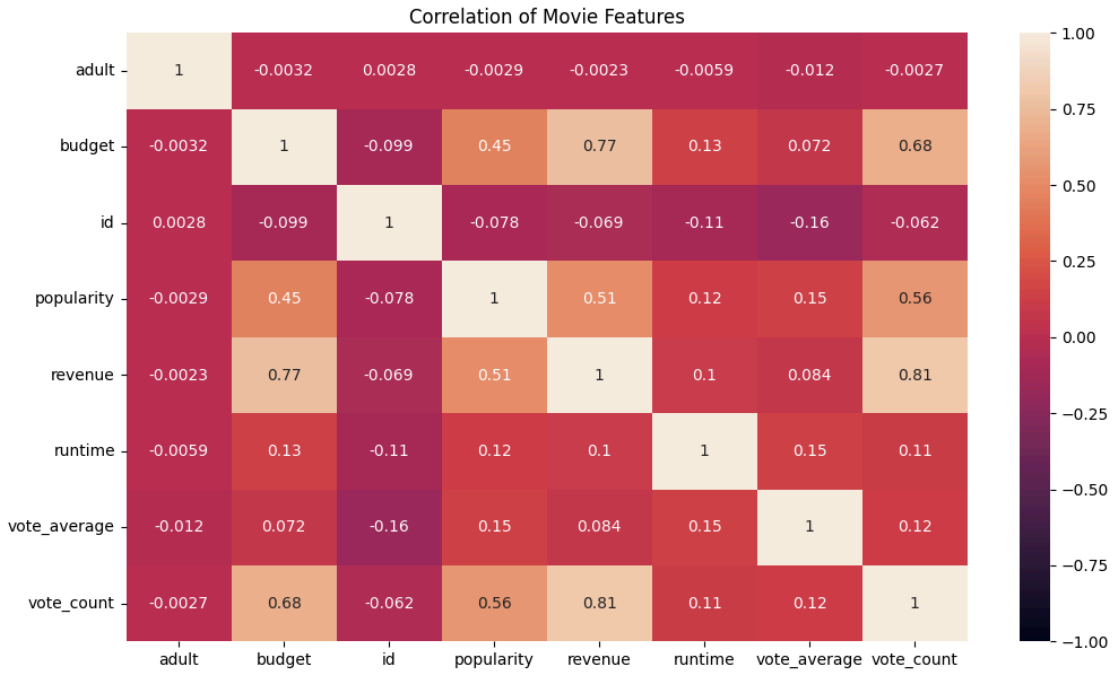
1)For this particular dataset, english is on top of the list for the original and spoken language in the movies

2)Jr. and Cedric Gibbons are actor and crew involved in the most movies in the list respectively

3)Warner Bros. with 1194 movies make it become top 1 production company in the list

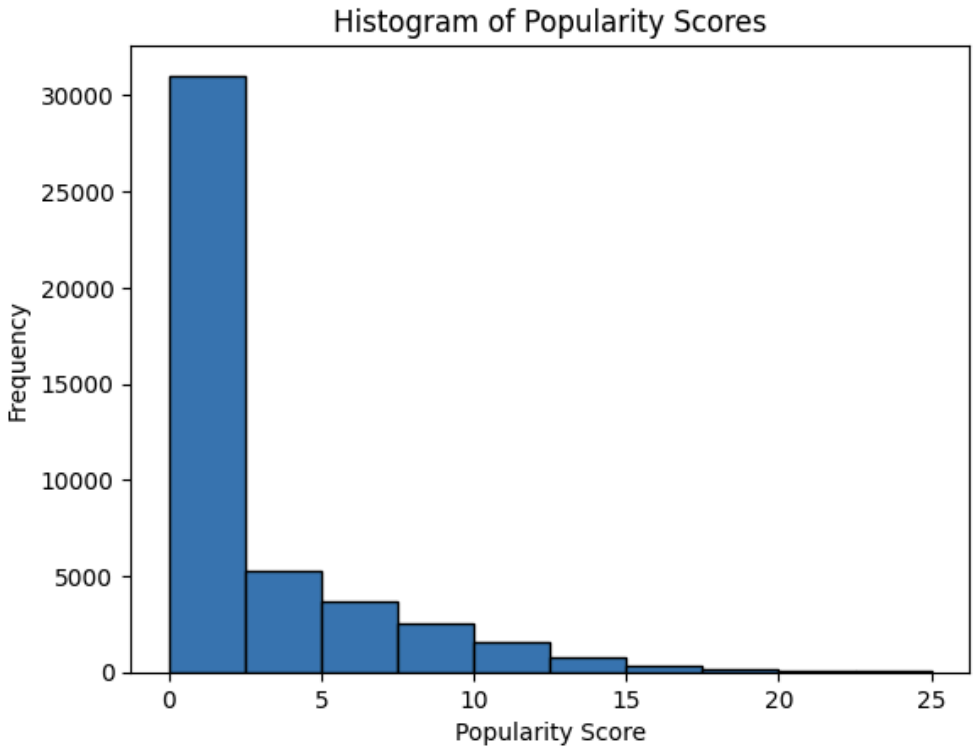
4)Many great production companies come from USA. So, it's not a surprise if USA is become our top 1 for production country

Correlation of Movie Features using Heatmap



1. Budget and Revenue: There is a relatively strong positive correlation (0.77) between budget and revenue. This suggests that movies with higher budgets tend to generate higher revenues.
2. Popularity and Revenue/Vote Count: Both popularity and vote_count have a moderate positive correlation with revenue (0.51 and 0.81 respectively), indicating that more popular movies, or those with more votes, generally bring in more revenue.
3. Budget and Vote Count: There's a moderate positive correlation (0.68) between budget and vote_count, which could imply that movies with larger budgets tend to get more votes, possibly due to more extensive marketing or higher production values that attract viewers.

Distribution of Popularity Scores



1. Skewed Distribution: The data is heavily skewed to the right. Most scores lie between 0 and 5, with the highest frequency in the 0-2.5 bin.
2. Gradual Decline: After the first bin, the frequency decreases steadily across the subsequent bins, highlighting that items with higher popularity scores become progressively fewer.
3. Rare High Scores: Very few items have scores greater than 10, and those beyond 15 are even rarer.

2. Data preparation

For the content based and Hybrid Recommendation System

Importing Data and Primary Cleanup

Initial Cleanup: For the movies_metadata.csv file, several non-essential columns ('belongs_to_collection', 'homepage', 'imdb_id', 'poster_path', 'status', 'title', and 'video') were dropped. Rows with indices 19730, 29503, and 35587 were removed due to corrupt data.

Type Conversion: The id column was explicitly converted to an int64 data type to maintain numerical consistency for merging operations.

Data Merging

The data from movies, credits, and keywords were merged into a single DataFrame (df) on the id column. This step ensured that each movie had consolidated information available for modeling. After merging The DataFrame contains 42,373 rows, which represents the movies in the dataset. There are 21 features (columns) in the DataFrame. Each column represents a specific characteristic of the data, such as genres, release_date, and overview.

Handling Missing Values

Missing values were filled in the original_language, runtime, and tagline columns with empty strings or zeros (for runtime). This step prevented errors related to null values during data analysis and model training.Any remaining rows with missing values were dropped.The index was reset to ensure a continuous sequence of indices after removing rows.

Extracting Meaningful Information:

Purpose: Some columns contained structured text data stored as strings of lists/dictionaries. The function literal_eval was used to convert these strings into actual lists or dictionaries.

Column Transformations: The custom function get_text was applied to several columns to extract relevant information:

Genres, Production Companies, Keywords, etc.: Textual data like genre names and production company names were extracted into more readable forms.

Characters and Actors: Separate columns were created to list the actors and the characters they played.

Reducing Redundancy:

The cast column was dropped after relevant information was extracted.

Duplicate rows based on original_title were removed to ensure unique movie titles.

Index Reset: After removing duplicates, the DataFrame's index was reset for accuracy and continuity.

Changing Data Types:

The release_date column was converted to a DateTime format.

The budget and popularity columns were converted to float64.

Importing Ratings Dataset and Performing Cleanup

Importing Ratings Data: The ratings.csv file was imported and cleaned similarly to the other datasets.

Converting Timestamps: The timestamp column was converted into a readable date format and stored in a new date column. The original timestamp column was dropped.

Merging with the Main DataFrame:

The ratings data was merged with the main df DataFrame to include movie details.
Rows with missing IDs post-merge were removed.
The redundant id column was dropped after ensuring valid movie details.
Index Reset: After filtering, the index was reset for continuity and DataFrame integrity.

For KNN model

Merge Datasets:

First, we merged the credits and keywords datasets with the movies dataset to form a new comprehensive DataFrame. This step ensures that all relevant data is collected in one place for analysis.

Remove Irrelevant Columns:

Columns that were not essential for analysis, such as certain identifiers and metadata columns, were removed to streamline the dataset. Removing unnecessary features helps to simplify the data and improve computational efficiency.

Fill Missing Values:

Missing values were addressed to ensure consistency across the dataset:
For string columns (str), missing values were filled with empty strings ('').
For numerical columns (int or float), missing values were replaced with zeros (0).

Drop Remaining NaN Rows:

Any rows that still contained missing values after the imputation step were removed entirely. This ensures that no records with incomplete data remain in the dataset.

Convert Column Types:

Appropriate columns were converted from object types to either int or float, where applicable, to enable numerical operations. For instance, the budget column was converted to a numerical type.

Map the 'Adult' Category:

The 'adult' category, originally labeled as True or False, was mapped to binary numerical values (1 for True and 0 for False).

Create Numerical DataFrame:

A new DataFrame containing only numerical data types (integers or floats) was created to facilitate numerical analyses and model training.

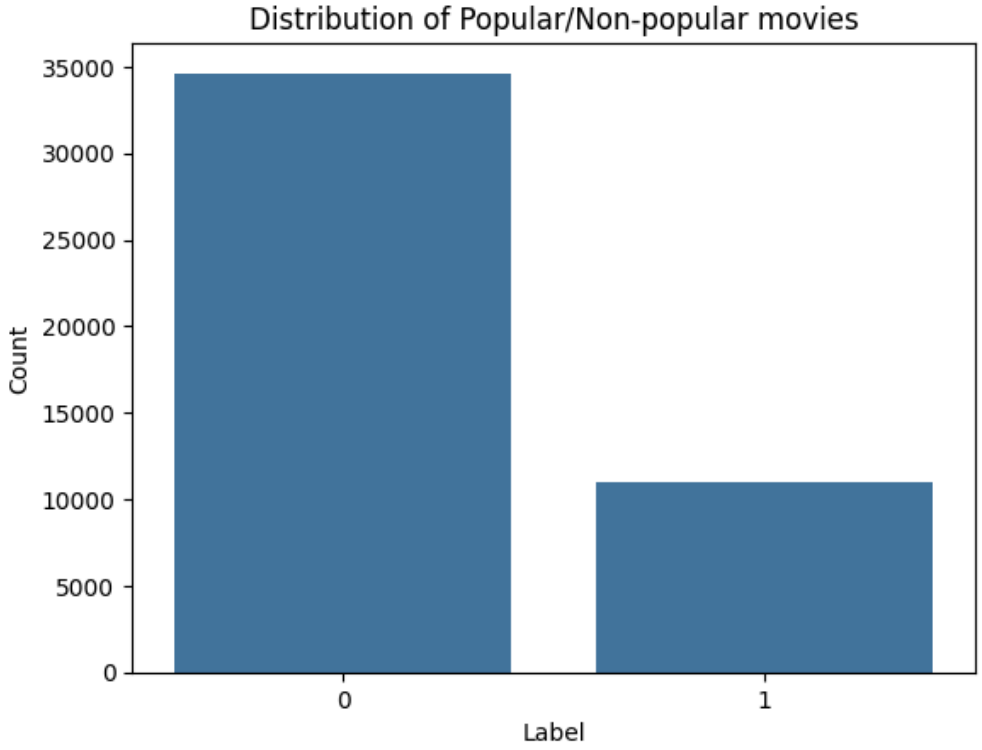
Extract Release Year and Month:

The release date was split into two new columns: release_year and release_month. This provides more granular insights into movie release patterns.

Encode Popularity Scores:

A function called label_encoder was defined to convert the popularity score into a binary classification label. If a movie's popularity score is greater than 4, it's labeled as 1 (popular); otherwise, it's labeled as 0 (non-popular).

Distribution of the popular/unpopular after converting to binary



df before and after processing:

3.Machine Learning Models

Content Based Model

The first method we are using to build a recommendation system is content based. We calculate a weighted rating for movies that uses both the average votes a movie receives (vote_average) and the number of votes (vote_count) to create a score that considers both popularity and critical acclaim. This method is commonly used to prevent movies with very few high votes from ranking too highly.

Here's a breakdown of the variables and calculations:

1. R: the average rating for each movie (vote_average).
2. v: the number of votes for each movie (vote_count).
3. m: the minimum votes required to be listed in the chart. Set to the 80th percentile of the vote_count, which means only movies with more votes than at least 80% of the movies in the dataset will be considered.
4. C: the mean vote across the whole report (vote_average.mean()).

The weighted average is calculated by this formula: $W = (R \cdot v + C \cdot m) / (v + m)$

where:

- W = weighted rating
- R = average for the movie as a number from 1 to 10 (mean) = (Rating)
- v = number of votes for the movie = (votes)
- m = minimum votes required to be listed in the Top 250 (currently 25,000)
- C = the mean vote across the whole report (currently 7.0)

The part R*v gives more weight to movies with more votes.

The part C*m adds a baseline score based on the average rating.

The denominator v+m normalizes the score.

After computing the weighted average, the code scales both popularity and weighted_average features between 0 and 1 using MinMaxScaler. This ensures that both features contribute equally to the final score since they are now on the same scale.

Then, the DataFrame weighted_df is created with these scaled values, and an index is set to the movie titles (original_title).

The final score is computed as a weighted sum of weighted_average (with a weight of 0.4) and popularity (with a weight of 0.6). This score aims to balance a movie's popularity with its critical acclaim.

Finally, the movies are sorted by their score in descending order, and the top 10 movies are displayed.

The output shows the top 10 movies with their respective popularity, weighted_average, and the calculated score. The original_title serves as the index of the DataFrame. For example, "Minions" has a popularity score of 1.0 (the highest among the top 10 after scaling) and a weighted_average of approximately 0.60, resulting in a final score of 0.840453. "Whiplash", despite having the highest weighted_average of 1.0, has a lower popularity score, which results in a final score of 0.470465. The mix of these two factors is what places these movies at their respective ranks in the list.

original_title	popularity	weighted_average	score	bag_of_words
Minions	1.000000	0.603532	0.841413	false family animation adventure comedy minion...
Big Hero 6	0.390602	0.827561	0.565386	false adventure family animation action comedy...
Baby Driver	0.416507	0.727736	0.540998	false action crime after being coerced into wo...
Guardians of the Galaxy Vol. 2	0.338511	0.794867	0.521054	false action adventure comedy science fiction ...
Pulp Fiction	0.257449	0.908395	0.517827	false thriller crime a burgerloving hit man hi...

original_title	popularity	weighted_average	score
Minions	1.000000	0.603532	0.841413
Big Hero 6	0.390602	0.827561	0.565386
Baby Driver	0.416507	0.727736	0.540998
Guardians of the Galaxy Vol. 2	0.338511	0.794867	0.521054
Pulp Fiction	0.257449	0.908395	0.517827
Deadpool	0.343132	0.764657	0.511742
Gone Girl	0.282748	0.843413	0.507014
The Dark Knight	0.224968	0.909123	0.498630
Avatar	0.338036	0.732643	0.495879
John Wick	0.335843	0.699476	0.481297

We already got the first result of our recommender system (above), and now we will combine those score with the similarity score. A common method to find similarity between 2 movies is a method called cosine similarity.
However, calculate similarity for all the movies require an expensive resources. So, we only take the first 10000 movies from weighted_df_sorted

The model is a content-based filtering system rather than a traditional predictive model like regression or classification. It does not predict a continuous value or classify input into categories but instead recommends items based on similarity scores. The structure of the model is built upon the calculation of a "bag of words" from textual data and applying TF-IDF vectorization to this data. Cosine similarity is then used to measure how similar the movies are to each other based on their content. It relies on a matrix operation that calculates the cosine similarity between vectors representing movies.

Model Parameters and Predictions:

In this content-based recommender system, the "parameters" could be considered as the features extracted from the TF-IDF vectorization process and the cosine similarity scores. Once the TF-IDF matrix is created, the system does not explicitly learn from data in the same way that a neural network or regression model does; thus, it does not have model parameters in the traditional sense.

Hyperparameters in the code:

min_df in the TfidfVectorizer - This is set to 5, meaning that terms appearing in fewer than 5 documents will be discarded.

similarity_weight in the predict() function - This is used to weight the importance of the content-based similarity relative to other scoring metrics; I used 0.7 in my model.

Machine Learning and Non-Machine Learning Aspects

Machine Learning Aspects:

1. Feature Extraction and Similarity Calculation:
    - The model uses textual data (movie descriptions, genres, etc.) and applies a method called TF-IDF (Term Frequency-Inverse Document Frequency) to create numerical vectors representing the features.
    - Cosine similarity calculates the similarity between different movies based on these vectors.
    - This scoring is directly tied to machine learning due to the use of natural language processing (NLP) and vectorization.
2. Weighting Scores for Ranking:
    - By combining popularity and critical acclaim through a weighted average formula, the system attempts to give a balanced recommendation that accounts for different factors.
    - Scaling scores to the same range using MinMaxScaler ensures that they contribute equally to the final score.
3. Predicting Recommendations:
    - Although the model doesn't "predict" like a regression or classification model, it does recommend based on patterns derived from feature extraction and similarity calculations.
    - The recommendations rely on the model's ability to learn and identify relevant patterns between movies using the above techniques.

Non-Machine Learning Aspects:

1. Rule-Based Filtering:
    - The weighted rating formula relies on a predefined set of rules to balance popularity and critical acclaim.

- Setting a threshold for the minimum number of votes (m) ensures only well-voted movies are considered.
- These rules do not learn dynamically but are predetermined.

2. Pre-Defined Weight Assignment:
    - The specific weight assignments (e.g., 0.4 and 0.6) are chosen manually rather than optimized through iterative training.

Differences from Classical Programming:

1. Content-Based vs. Rule-Based Filtering:
    - Classical programming involves explicit rule-based recommendations, often hardcoded (e.g., "if genre is 'action', then suggest 'movie X'").
    - Content-based filtering leverages similarity measures derived from numerical features, which generalize better for unseen data and adapt to new inputs.
2. Generalization:
    - The content-based approach generalizes to new and diverse inputs beyond predefined rules and dynamically ranks items using mathematical scoring functions.

Hybrid Recommendation Model

The second method we are using to build a recommendation system is a hybrid recommendation system.The model used combines deep learning with traditional collaborative filtering techniques.

Deep learning: Using artificial NNs to analyze large amounts of movie data and make personalized movie recommendations based on a user's past preferences and behaviors.

Traditional Collaborative Filtering: Uses past interactions between users and items to make recommendations. It assumes that similar users will like imilar items and generates recommendations based on the past behavior of similar users.

The model takes into account both explicit ratings given by users and implicit signals such as genres and uses a combination of neural networks and matrix factorization to make personalized movie recommendations. The machine learning model implemented for the recommender system is based on TensorFlow Recommenders (TFRS), a high-level Keras API.

The structure of the model can be categorized as a hybrid neural network model that deals with both regression and retrieval tasks. It is not a simple or multivariate regression; rather, it's a complex system that combines these regression tasks with classification-like retrieval tasks.

For the regression part of the model, it predicts the rating a user would give to a movie, which is a continuous output making it a regression problem. For the retrieval part, the model classifies which items should be presented to the user, which can be seen as a multi-class classification task where the classes are the items to recommend.

The model is composed of several layers and components:

1. User Model: A neural network layer that maps user input to an embedding space. This layer utilizes a StringLookup layer to encode the user IDs followed by an Embedding layer.
2. Movie Model: Similarly, this layer maps movie input to an embedding space using a StringLookup layer for movie titles followed by an Embedding layer.
3. Rating Model: This part of the network is responsible for predicting the rating. It comprises a sequence of Dense layers with ReLU activations, indicating that the model is deep enough to capture non-linear relationships.The network has two Dense layers with 256 and 128 neurons respectively, before the final output layer which provides the predicted rating.
4. Task Layers: The model includes two tasks: a Ranking task which calculates the loss for the rating prediction, and a Retrieval task that handles the identification of relevant items to recommend to the user.

The model's design allows it to learn both user preferences and item characteristics in a unified manner, thus enabling personalized recommendations that take into account both explicit feedback (ratings) and implicit feedback (retrieval).

Evaluation Process of the Hybrid Recommendation System:

1. Training and Validation Split:
   The dataset is split into training and validation sets to evaluate the model’s performance on unseen data. This ensures that the system is not overfitting and generalizes well.

2. Evaluation Metrics:

   The system is evaluated based on two key tasks: regression (rating prediction) and retrieval (item classification). Here's a detailed breakdown of the metrics used for each:

   Root Mean Squared Error (RMSE):

   RMSE measures the accuracy of the rating predictions by calculating the square root of the mean squared differences between predicted and actual ratings. This is crucial in evaluating the model's ability to predict user ratings. Lower RMSE values indicate higher predictive accuracy.

   Top-K Categorical Accuracy:

   This metric measures the accuracy of the retrieval task. Specifically, it determines how often the recommended items are among the top-K relevant items. For example, top-10 accuracy indicates how often the relevant item appears within the top 10 suggestions provided to a user.

3. Training Dynamics:

   The model undergoes several epochs of training, and metrics like training RMSE and validation RMSE are monitored to identify trends in model performance. The retrieval task uses precision and recall-like metrics (e.g., FactorizedTopK) to measure recommendation quality.

Regression and Retrieval Processes in Overall Performance:

- Regression (Rating Prediction):

  The regression component directly predicts a user's rating for a particular item (e.g., a movie). This prediction helps gauge a user's potential interest level. The system uses this numeric rating to rank items before recommendation. A low RMSE in this task means that the rating predictions are accurate, which can indirectly improve the retrieval ranking.

- Retrieval (Item Classification):

  The retrieval component classifies items into "relevant" or "irrelevant" based on user preferences and behaviors. The FactorizedTopK metric ensures that the retrieved items align with user interests. The retrieval task directly influences recommendation relevance and diversity by filtering through numerous items to recommend a subset that aligns with the user's preferences.

Relationship to Overall Evaluation/Performance:

- The regression task ensures that the system ranks items correctly by predicting the level of interest accurately. This improves ranking quality and ensures that recommendations are relevant to each user.
- The retrieval task filters out irrelevant items and prioritizes those most suitable for a user. This leads to improved relevance, increasing the likelihood that users will engage with the recommended items.

KNN model

In a Movie recommendation system, we should aim to recommend good amount of popular movies since users are more likely to watch a more popular movie. The classification task at hand is to predict whether a movie is "popular" or not, as defined by a popularity score threshold (in next slide). The label column (popular/not popular) is the target variable for my classification model, and adult, budget, popularity, revenue, runtime, vote_average, vote_count, release year & month are input features for classification. The goal of the KNN classifier is to learn from the input features how to predict this binary label (for new, unseen data)

Model Overview:
The KNN model is used for a classification task to predict whether a movie is "popular" or "not popular," based on a specified popularity score threshold. The label column (popular/not popular) is the target variable for classification, while input features include:

- Adult (True/False)
- Budget
- Popularity
- Revenue
- Runtime
- Vote Average
- Vote Count
- Release Year
- Release Month

Model Structure:
The KNN classifier aims to learn from the input features to predict this binary label for unseen data. Key steps in building the model include:

1. Data Cleaning and Feature Extraction - Explained earlier
2. Training/Validation Split:
    - An 80/20 split is used to train and evaluate the model.
    - An initial random value of K = 2 is selected to build a baseline model.

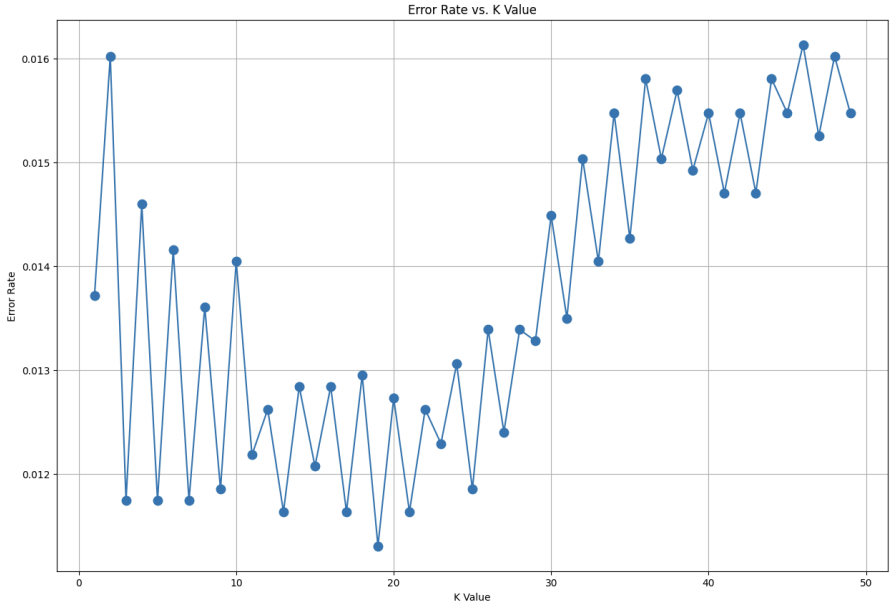
Hyperparameter Tuning:

Finding the optimal K is critical for improving classification performance. An iterative approach was used to identify the optimal K value:

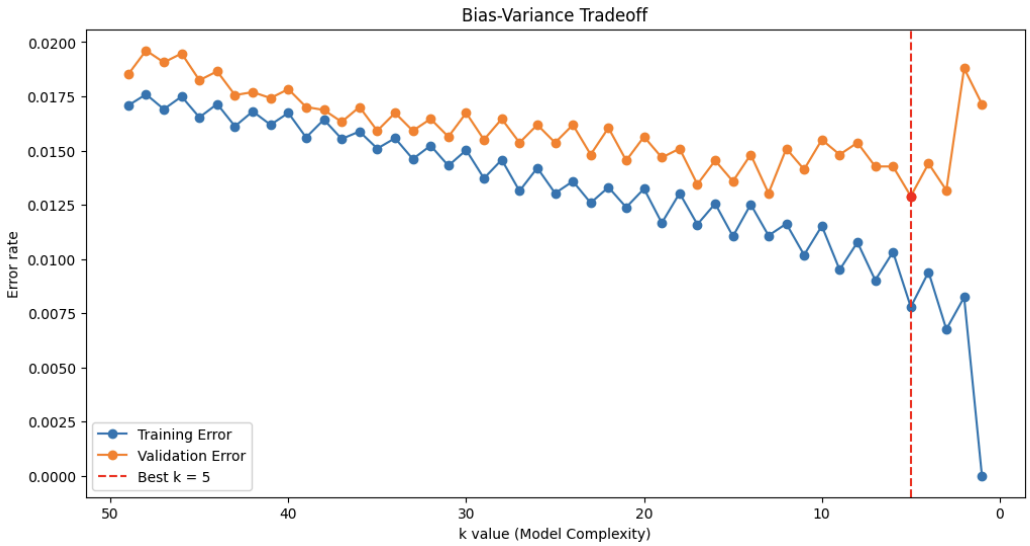
Initial Results:With K = 2, the model showed 98% accuracy and a balanced confusion matrix.Further tuning showed that the minimum error rate occurs at K = 19, although this does not guarantee it is optimal.

But the question is can we do better? To find out we performed few steps:

1. Check for overfitting and underfitting
2. Balance between bias and Variance
3. K fold cross validation
4. Check accuracy, precision, recall etc with different values of K

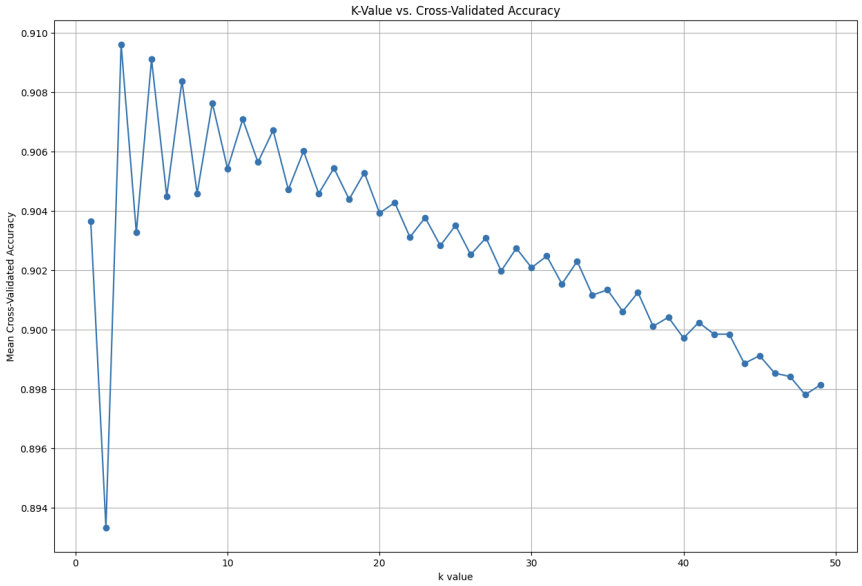


The minimum error rate is at k = 19. It gives the minimum error rate, but it does not guarantee that this K is the optimal choice



We are looking for a 'k' value that results in both low training and validation errors, which indicates a model that performs well on both the data it was trained on and unseen data.

The best 'k' value is typically where the validation error is at its lowest, or just before it starts to increase which is at k = 5 in our case. After that we can see that Validation error increases while training error keeps decreasing (Overfitting). The training error is also low at k = 5, which is an added bonus, as it suggests that the model has learned well from the training data without overfitting.



This graph tells us that best k value may be 3 with a cross-validated accuracy of 0.909

Results and Evaluation

Content Based

The model could be evaluated using Precision and Recall However for this dataset and the model it seemed too complicated to perform this and I did not want to delve deeper since content based is not purely a ML model we learned in class.

Alternative evaluation Methods that could be used:

- User Feedback: Recommendations can be evaluated by directly collecting feedback from users. If the suggested movies align well with their preferences, the system is considered effective.
- A/B Testing:By providing different sets of recommendations to different user groups, you can analyze the relative performance and satisfaction rates of the content-based model compared to other approaches.

```
predict('Toy Story', similarity_weight=0.7, top_n=10)
# predict('Mean Girls', similarity_weight=0.7, top_n=10)
```

original_title	score	similarity	final_score
Toy Story	0.348515	1.000000	0.804555
Toy Story 2	0.317785	0.537320	0.471460
Toy Story 3	0.336500	0.274778	0.293295
Toy Story of Terror!	0.282269	0.294860	0.291082
Small Fry	0.256223	0.271028	0.266586
Hawaiian Vacation	0.266277	0.263819	0.264556
Minions	0.841413	0.005376	0.256187
Finding Nemo	0.346185	0.203631	0.246397
WALL·E	0.348682	0.196733	0.242317
A Bug's Life	0.284638	0.215011	0.235899

In the final results we can see that when we input toy story, it suggests us similar movies like toy story 2,3 and more animation movies

Hybrid model

Epoch	RMSE	Val RMSE	ftop_k/top10	Valftop_k/top10
1	1.446037	1.149693	0.019571	0.018564
2	1.032969	1.333894	0.045057	0.012335
3	1.007649	1.283364	0.057171	0.009648
4	0.980151	1.323807	0.064543	0.008793
5	0.961967	1.346160	0.074543	0.007938
6	0.945836	1.361902	0.086629	0.007328
7	0.932139	1.368814	0.096286	0.007450
8	0.918718	1.370226	0.102057	0.007450
9	0.906064	1.372158	0.106314	0.007328
10	0.894248	1.366074	0.108657	0.007572

Over the 10 epochs, the training RMSE generally decreases, indicating that the model is learning and improving its predictions on the training dataset. An RMSE of 0.89 is considered good because the dataset is quite complex. An RMSE of 0.89 means that, on average, the predicted movie ratings differ from the actual ratings by 0.89 points on the rating scale.The values for Top-K Categorical Accuracy are relatively low, suggesting that the model does not rank the relevant item as highly as desired. This issue is common in recommendation systems because of thelarge number of possible items.

```
predict_movie(3, 5)
```

```
Top 5 recommendations for user 3:

1. Carne trémula
2. Un long dimanche de fiançailles
3. The Ewok Adventure
4. 300
5. Finding Neverland
```

```
[96] predict_rating(3, "Carne Tremula")

     Predicted rating for Carne Tremula: 3.159895658493042
```

```
[97] predict_rating(3, "Finding Nemo")

     Predicted rating for Finding Nemo: 2.566805601119995
```

We see that Carne Tremula is suggested at the top and it has a higher predicted rating than nemo since the user's historical data suggests they watch more Drama than animation movies

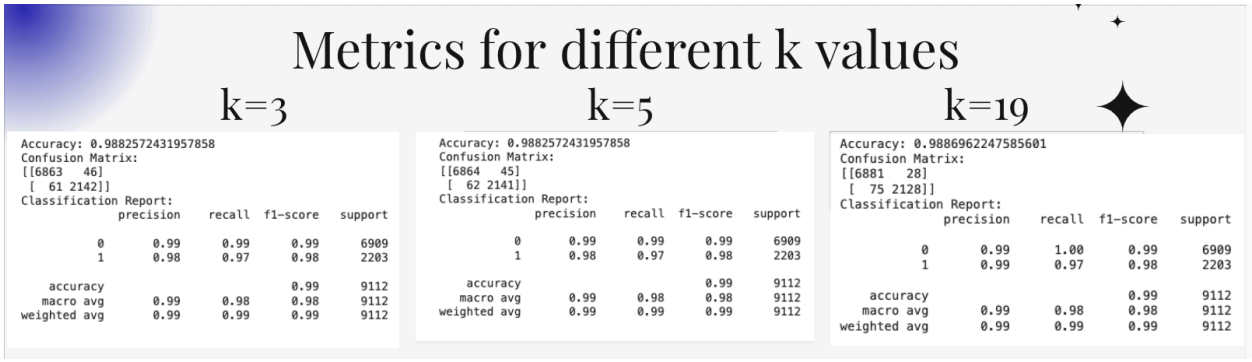
Genres of suggested top 5 movies

	original_title	genres	overview
1	Carne trémula	Drama, Romance, Thriller	A scorned ex-convict forces himself into the l...
2	Un long dimanche de fiançailles	Drama	In 1919, Mathilde was 19 years old. Two years ...
3	The Ewok Adventure	Adventure, Family, Fantasy, Science Fiction, T...	The Towani family civilian shuttlecraft crashe...
4	300	Action, Adventure, War	Based on Frank Miller's graphic novel, "300" i...
5	Finding Neverland	Drama	Finding Neverland is an amusing drama about ho...

Genre of movies in user's historical data

	userId	movieId	rating	date	original_title	genres	overview
63	3	110	4.0	2011-02-28 19:40:49	Trois couleurs : Rouge	Drama, Mystery, Romance	Red This is the third film from the trilogy by...
64	3	247	3.5	2011-02-28 02:53:57	The Killing	Drama, Action, Thriller, Crime	The Killing was Stanley Kubrick's first film w...
65	3	267	3.0	2011-02-28 02:56:01	Carne trémula	Drama, Romance, Thriller	A scorned ex-convict forces himself into the l...
66	3	296	4.5	2011-02-28 03:06:58	Terminator 3: Rise of the Machines	Action, Thriller, Science Fiction	It's been 10 years since John Connor saved Ear...
67	3	318	5.0	2011-02-28 03:02:01	The Million Dollar Hotel	Drama, Thriller	The Million Dollar Hotel starts with a jump fr...
68	3	377	2.5	2011-02-28 20:00:42	A Nightmare on Elm Street	Horror	Teenagers in a small town are dropping like fl...
69	3	527	3.0	2011-02-28 03:08:48	Once Were Warriors	Drama	A drama about a Maori family lving in Auckland...
70	3	588	3.0	2011-02-28 19:41:40	Silent Hill	Horror, Mystery	The eerie and deserted ghost town of Silent Hi...
71	3	592	3.0	2011-02-28 20:00:47	The Conversation	Crime, Drama, Mystery	Surveillance expert Harry Caul (Gene Hackman) ...
72	3	593	3.0	2011-02-28 19:37:20	Солярис	Drama, Science Fiction, Adventure, Mystery	Ground control has been receiving strange tran...
73	3	595	2.0	2011-02-28 20:01:00	To Kill a Mockingbird	Crime, Drama	In a small Alabama town in the 1930s, scrupulo...
74	3	778	4.0	2011-02-28 03:19:17	Les Vacances de Monsieur Hulot	Comedy, Family	Monsieur Hulot comes to a beachside hotel for ...
75	3	866	3.0	2011-02-28 02:54:47	Finding Neverland	Drama	Finding Neverland is an amusing drama about ho...
76	3	1271	3.0	2011-02-28 02:53:25	300	Action, Adventure, War	Based on Frank Miller's graphic novel, "300" i...
77	3	1378	4.0	2011-02-28 02:54:18	Shortbus	Romance, Drama, Comedy	A group of New Yorkers caught up in their mili...
78	3	1580	3.5	2011-02-28 19:41:29	Rope	Crime, Drama, Mystery, Thriller	Two young men strangle their "inferior" classm...
79	3	1721	4.5	2011-02-28 20:00:36	...Più forte ragazzi!	Adventure, Action, Comedy	The "Trinity" crew makes another modern era fi...
80	3	1884	4.0	2011-02-28 03:19:03	The Ewok Adventure	Adventure, Family, Fantasy, Science Fiction, T...	The Towani family civilian shuttlecraft crashe...
81	3	2028	4.0	2011-02-28 19:37:42	Say Anything...	Comedy, Drama, Romance	A budding romance between noble underachiever ...
82	3	2694	3.0	2011-02-28 03:11:50	Tuya de hun shi	Drama, Romance, Foreign	Set in Inner Mongolia, a physical setback caus...
83	3	2762	3.5	2011-02-28 19:40:57	Young and Innocent	Drama, Crime	Derrick De Marney finds himself in a 39 Steps ...
84	3	2841	4.0	2011-02-28 02:55:33	Un long dimanche de fiançailles	Drama	In 1919, Mathilde was 19 years old. Two years ...
85	3	2959	5.0	2011-02-28 03:14:34	License to Wed	Comedy	Newly engaged, Ben and Sadie can't wait to sta...
86	3	44191	3.5	2011-02-28 22:39:00	Loose Screws	Comedy	Brad, Steve, Hue, and Marvin are four get-nowh...
87	3	58559	3.0	2011-02-28 19:41:11	Confession of a Child of the Century	Drama	Paris, 1830: Octave, betrayed by his mistress,...

KNN Model



Accuracy: The accuracy is very high (>=98%) for all three k values and does not vary much between them. So we can predict if a unseen movie is popular or not with over 98% accuracy.

Confusion Matrix:

For k=3, there's a balance between false positives (FP) and false negatives (FN), with a slightly higher number of FNs.

For k=5, the number of FNs decreases slightly compared to k=3, indicating better sensitivity or recall for the positive class.

For k=19, the number of FNs drops significantly, suggesting the model is becoming better at classifying positive instances correctly. However, the FPs have slightly increased, showing a trade-off between the two types of errors.

Classification Report:

Precision and recall for both classes are high across all k values, which is good. However, there's a slight improvement in recall for the positive class (1) as k increases.

The f1-score, is also high across all k values, indicating a good balance between precision and recall.

The support shows that the dataset is imbalanced with more instances of class 0 than class 1. Despite this imbalance, the model performs well across both classes.

Model Complexity: The metrics do not degrade as k increases, suggesting that the model is not overfitting at lower k values. We can stick to lower value of k (5) as increasing model complexity is computationally more expensive especially for large amount of data.

The model performs well for all tested k values with high accuracy, precision, recall, and f1-scores. The choice of the best k would depend on the specific costs associated with false positives (predicting a movie as popular when it is not popular) versus false negatives (predicting a movie as not popular when it is popular). And also the cost of computation

Bias & Ethical Issues

1. Data Bias:

   The datasets used may be skewed due to the historical behavior of specific demographics. For instance, a bias toward Western and English-language movies can result in underrepresentation of non-English films in recommendations.
2. Popularity Bias:

   Models like K-Nearest Neighbors favor popular items, leading to a repetitive cycle where only highly-rated and popular movies are recommended while less mainstream films are marginalized.
3. Stereotyping and Diversity:

   Models often categorize users based on historical preferences. This can reinforce stereotypes by limiting exposure to diverse genres and creating echo chambers of repetitive suggestions.
4. Algorithmic Fairness:

   Over-reliance on data-driven algorithms could lead to unfair recommendations, excluding niche audiences and reinforcing biases. This can affect smaller studios and less popular filmmakers.

Limitations & Future work

1. Cold-Start Problem:

   New users and movies without significant historical data face challenges in receiving or providing accurate recommendations. Exploring techniques like matrix factorization and leveraging external data could mitigate this issue.
2. Scalability:

   While the KNN approach is accurate, it struggles with computational efficiency as the dataset expands. Efficient algorithms and dimensionality reduction can help scale recommendations for larger user bases.
3. Evaluation Metrics:

   Current metrics focus heavily on accuracy. However, future work should incorporate novelty and diversity measures to ensure recommendations maintain a broader appeal.
4. Explainability:

   Users increasingly demand transparency and understanding of recommendations. Future models should include explainable frameworks that clearly articulate why specific movies are suggested.

5. Integration of Social Data:
   Incorporating social media activity or user-generated reviews could improve the personalized recommendations by adding another layer of preference data.
6. Multi-Dimensional Recommendations:
   While current models focus on predicting ratings or popularity, future systems could consider factors like mood, time of day, or special events to suggest movies tailored to specific situations.

References

1.Campos, P., Garzotto, F., Negro, S., Papadopoulos, A. V., & Turrin, R. (2011). Looking for "Good" Recommendations: A Comparative Evaluation of Recommender Systems. In P. Campos et al. (Eds.): INTERACT 2011, Part III, LNCS 6948 (pp. 152–168). Springer-Verlag. https://doi.org/10.1007/978-3-642-23765-2_11

2.Pradeep, N., Rao Mangalore, K. K., Rajpal, B., Prasad, N., & Shastri, R. (2020). Content-Based Movie Recommendation System. International Journal of Research in Industrial Engineering, 9(4), 337–348. https://doi.org/10.3923/jrindeng/v9n4

3.Reddy, S.R.S., Nalluri, S., Kunisetti, S., Ashok, S., & Venkatesh, B. (2019). Content-Based Movie Recommendation System Using Genre Correlation. In S.C. Satapathy et al. (Eds.): Smart Intelligent Computing and Applications, Smart Innovation, Systems and Technologies 105 (pp. 391–398). Springer. https://doi.org/10.1007/978-981-13-1927-3_42