Which Movie Should I watch? Analyzing different machine learning approaches to recommend movies

Bhagesh Gaur
bhagesh20558@iiitd.ac.in

Mohit Sharma
mohit20864@iiitd.ac.in

Ojus Singhal
ojus20094@iiitd.ac.in

Vaibhav Jaiswal
vaibhav20547@iiitd.ac.in

Abstract

Recommendation systems in 2022 have become quite complex and sophisticated owing to their financial value. We explore some of the techniques used in recommendation systems and try to create a robust hybrid system combining ideas from Content-Based Filtering and Collaborative filtering. For Content-Based Filtering, we tested multiple clustering algorithms such as K-Means, Spectral Clustering, and DBSCAN, and found out that DBSCAN gives us the best results owing to its robustness to noise and high dimensionality, combined with its run-time efficiency.

1. Introduction

With the growing popularity of OTT (over-the-top) platforms like Netflix, Prime Video and Disney Hotstar, it has become increasingly valuable to recommend a movie/show to the user that they will watch and enjoy with a high probability. A good recommendation can help OTT companies maximise revenue and increase customer satisfaction. A host of features such as cast, director, genre, budget, duration, language, votes, rating and many others may come into play while recommending movies/shows. Understanding the correlation and importance of each feature in a chosen dataset is essential. Since open movie datasets are large and have irregularities and missing values, they must undergo thorough pre-processing. Our work initially explores pre-processing methods like custom-built one-hot encoders to improve the dataset's quality and then analyses different machine-learning approaches for a movie recommendation, like collaborative filtering and content-based filtering in which different clustering algorithms are explored like K-mean, DBSCAN, Spectral etc. The performance metric utilised will be the Silhouette Score. Our work shines on the utilising importance of pre-processing techniques on movie datasets. In retrospect, our work will find critical applications among movie enthusiasts and OTT platforms to make movie recommendations based on similar interests.

2. Literature survey

1. Movie Recommender System Using K-Means Clustering and K-Nearest Neighbour [1] : The proposed recommender system predicts the user's preference for a movie based on different parameters like genre, rating, genome (genome encodes how strongly movies exhibit particular properties represented by tags (atmospheric, realistic etc.)) and the concept that users have common preferences. The recommender utilises collaborative and content-based filtering. The rating prediction for each user is made with the help of a utility clustering matrix that is made by utilising K-means clustering, and a similarity matrix is created using Pearson's correlation matrix to calculate the similarity between users. Both the similarity and utility clustered matrix is utilised by K-nearest neighbours to make a prediction for the movie ratings for top N users. The paper uniquely utilises Within-Cluster Sum of Squares (WCSS) to find the right amount of clusters. The paper claims to define a more optimised recommender system than existing papers, and shows that RMSE is reduced on reducing the no of clusters.

2. Item-Based Collaborative Filtering Recommendation Algorithms [3] : The paper identifies scalability issues using large datasets with a collaborative filtering algorithm. It addresses the issues by presenting an item-based technique that analyzes the user-item matrix to identify relationships between different items and then uses these items to make recommendations. They looked into various methods for computing item-item similarities (e.g. cosine similarities between item vectors) and techniques for obtaining recommendations from them (using weighted sum and regression model). They show results equivalent to CF methods with the ability for better scalability in-comparison to item-based techniques.

3. Personalized Real-Time Movie Recommendation

System: Practical Prototype and Evaluation [4] : The paper proposes a collaborative filtering algorithm called KM-Slope-VU (and weighted KM-slope-VU), which reduces the time complexity of real-time recommendation. K-means clustering is used to cluster users into different clusters. Each cluster then produces a virtual opinion leader to improve complexity. The algorithm created a new VU-item matrix to replace the user rating prediction and make the prediction using that. This method of reducing the dimensions of the user-item matrix significantly reduces time complexity. The paper mentions using the Singular Value Decomposition (SVD) and matrix-factorization method in recommendation systems. The prediction accuracy of the proposed method is less than SVD and SVD++ but gives faster results.

3. Dataset Information and Pre-Processing

3.1. Dataset Details

We have picked the Movies Dataset, which comprises 46628 movies with twenty features. The data set contained six separate CSV files: movie metadata, keywords, and credits. We mapped movie data on ids to get 46628 entries. Table 1 shows the raw combined file's different datatypes and features with missing values (count less than 46628). The data is categorical in nature.

A certain movie can have multiple genres, production companies and countries, cast and crew members, and spoken languages in the dataset. Thus, pre-processing involved techniques such as encoding in-order to improve the dataset.

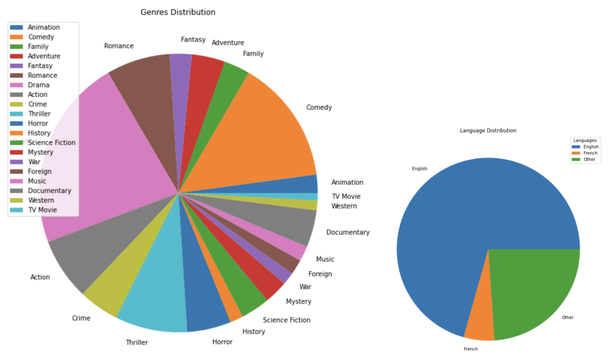

Figure 1. Genre and Language Distribution

Table 1. Raw Features (m: missing values)

Feature	Datatype	Count
adult	object	46628
belongstocollection	object	4574 (m)
budget	object	46628
genres	object	46628
homepage	object	8009 (m)
movieId	object	46628
imdbid	object	46611 (m)
originallanguage	object	46617 (m)
popularity	object	46624 (m)
overview	object	45633 (m)
popularity	object	46624 (m)
posterpath	object	46229 (m)
productioncompanies	object	46624 (m)
productioncountries	object	46624(m)
releasedate	object	46540 (m)
revenue	float64	46624 (m)
runtime	float64	46360 (m)
spokenlanguages	object	46624 (m)
status	object	46542 (m)
tagline	object	20783 (m)
title	object	46624 (m)
video	object	46624 (m)
voteaverage	float64	46624 (m)
votecount	float64	46624 (m)
keywords	object	46628
cast	object	46628
crew	object	46628

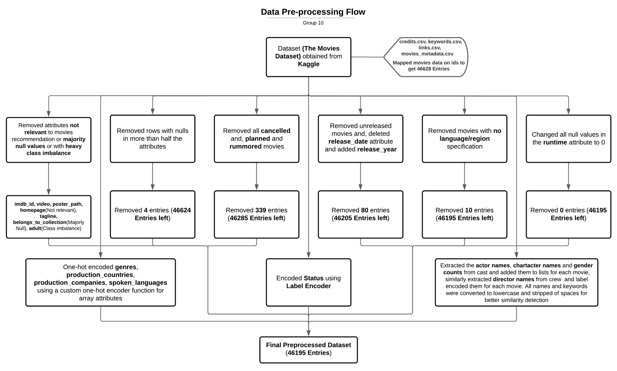
Data Pre-processing Flow
Group 10
Dataset (The Movies Dataset) obtained from Kaggle
credits.csv, keywords.csv, links.csv, movies_metadata.csv
Mapped movies data on ids to get 46628 Entries
Removed attributes not relevant to movies recommendation or majority null values or with heavy class imbalance
imdb_id, video, poster_path, homepage(Not relevant), tagline, belongs_to_collection(Majorly Null), adult(Class imbalance)
Removed rows with nulls in more than half the attributes
Removed 4 entries (46624 Entries left)
Removed all cancelled and, planned and rummored movies
Removed 339 entries (46285 Entries left)
Removed unreleased movies and, deleted release_date attribute and added release_year
Removed 80 entries (46205 Entries left)
Removed movies with no language/region specification
Removed 10 entries (46195 Entries left)
Changed all null values in the runtime attribute to 0
Removed 0 entries (46195 Entries left)
One-hot encoded genres, production_countries, production_companies, spoken_languages using a custom one-hot encoder function for array attributes
Encoded Status using Label Encoder
Extracted the actor names, chartacter names and gender counts from cast and added them to lists for each movie, similarly extracted director names from crew and label encoded them for each movie. All names and keywords were converted to lowercase and stripped of spaces for better similarity detection
Final Preprocessed Dataset (46195 Entries)

Figure 2. Pre-processing Flowchart

3.2. Pre-processing

Figure 2 represents a flowchart which shows the detailed pre-processing pipeline. The steps followed resulted in a processed encoded dataset with 46195 entries with over 61 attributes and eliminated all null values.

3.2.1 Relevancy, imbalance and null values in Attributes

The raw data set defined earlier has 26 attributes. Attributes like imdbid, video, posterpath and homepage are irrelevant to movie recommendations. Attributes tagline and belongs to the collection have a significant number of values as null (4000+ and 20,000+ respectively). The attribute adult has a class imbalance. Thus, we have removed the attributes mentioned to improve the dataset's quality.

3.2.2 Changes in rows of dataset

We started by removing rows with nulls in more than half the attributes since they would not help during training or testing. Followed by the removal of all cancelled, planned and rumoured movies. We finally removed movies with no language/region specification. We changed the release date attribute to the release year feature to reduce the data's complexity and relevance. We changed all the null values in the ”runtime” and revenue attribute to zero in-order to maintain consistency in the attribute.

3.2.3 Label Encoder

The ”status” attribute has multiple classes: released, rumoured, cancelled, post-production, in-production and planned. The previous step removed rumoured, cancelled and planned movies from the dataset. The rest of the classes were encoded using a label encoder (encoded classes to 0, 1, 2). The machine learning algorithms can use this integer encoding to understand and harness the relationship between each other.

3.2.4 Name extraction and mapping

Actor names, character names, director names and gender counts are given in the format as shown in the **example:** [”castid”: 242, ”character”: ”Jake Sully”....], ['castid': 1, 'character': 'George Banks', '...]
All names and keywords were initially stripped of spaces and converted to lowercase for better similarity detection (such as cosine similarity). A list for each movie is created for which actors, character names, and gender count are added. Director names have been labelled and encoded for each movie.

3.2.5 Custom one-hot encoder

In the case of the ”genre”, ”production companies”, ”production countries” and ”spoken languages” attributes, ordinal relationships do not exist between classes. Assuming an order between classes can result in poor performance of the model. In this case, a one-hot encoding can be applied to the integer representation. Since, for the attributes mentioned before, each row in the attribute can have multiple classes. This means that the in-built hot encoder [2] cannot be used as it only works for one dummy variable. Thus, we defined our custom one-hot encoder. The function traverses the entire attribute and creates a count of the classes of that attribute. (if multiple classes are present for a movie, then the count is increased according to the times the classes have appeared). Depending on the attribute, the custom one-hot encoder allows us to choose the top n classes (in terms of count) that the function will encode. In the case of ”genre”, we use all 19 classes, whereas for ”spoken language”, ”production companies” and ”production countries”, the top 10 classes have been used. The function creates a new feature/attribute for each chosen top class. The original attribute is traversed, and the values in the new feature column are updated to 1 or 0 based on the presence in the class. This process is done for all top n classes in an attribute. The process can successfully convert the data into a numerical form that can be input into machine learning algorithms.

3.3. Normalization using LDA

Initially, we used Principal Component Analysis to reduce the dimensionality of the dataset (normalization). Unfortunately, PCA only performs well on dense data. On the other hand, our dataset is sparse. Thus, it is more advisable to use Linear Discriminant Analysis (LDA) which works well on sparse data. LDA is used when we class with multiple features and need to separate them efficiently. LDA reduces the dimensionality for our dataset to 30.

4. Methodologies

We will be focusing on the ideas of two types of recommender systems - 1) Content-Based Filtering: The general idea is that if a person likes a particular movie, they will also like a similar one. The ideas are similar to item-based filtering [3].
2) Collaborative filtering: The general idea is that people with similar preferences can be recommended similar movies [1]. This does not require movie metadata

We aim to create a robust recommendation system based on a combination of these two methods. For the initial stage, we have focused on Content-Based filtering. This focus is in our favour since we have a large corpus of metadata on movies.

4.1. Content Based Filtering using Clustering

This method groups related content together using metadata like genre, cast, director etc., and recommends related content to users who have liked similar content. This approach boils down to clustering methods. Figure 3 shows the host of clustering algorithms from which we select the ones relevant to our dataset and problem requirements.

4.1.1 K-means Clustering

We start with baseline as K-means clustering since it is one of the most standard, basic and simple clustering al-

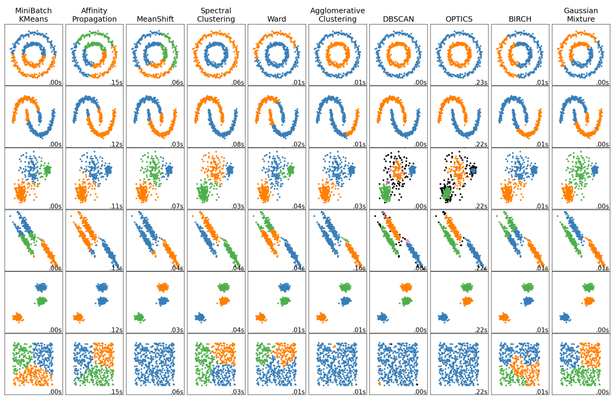

Figure 3. Clustering Algorithms from Sklearn Documentation

gorithms. The optimal number of clusters is 12, combining the elbow and silhouette methods results. Since our data is highly dimensional, and k-means fails in high dimensional data due to the **'curse of dimensionality'**, we will now use other clustering algorithms that perform better with high-dimensional data.

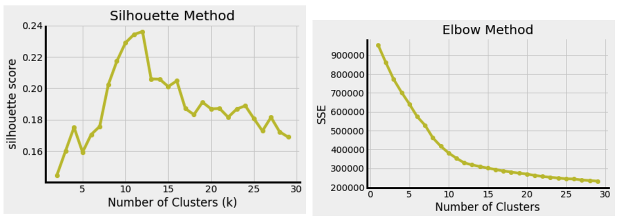

Figure 4. K-Means: Silhouette and Elbow

4.1.2 Spectral Clustering

Spectral clustering performs better with high-dimensional data in comparison to K-means clustering. K-means algorithm assumes that clusters are spherical within k-radius from the cluster centroid. In K means, many iterations are required to determine the cluster centroid. In spectral clustering, the clusters do not follow a fixed shape or pattern, hence it is more versatile. We can see from both the Eigengap heuristics and the Silhouette method that the optimal number of clusters is 41 (ignore lower values), which means our data contains 41 clusters according to this algorithm. This spectral algorithm kept crashing for the limited computational resources that we had.

4.1.3 DBSCAN (Density-Based Spatial Clustering of Applications With Noise)

We use DBSCAN here to combat both the high dimensionality of our data and the fact that our data is very noisy. DBSCAN is very robust in reducing noise points as it is a density-based clustering algorithm. We use the following steps to find the optimal parameters:

1. We use Eigengap heuristics to get the epsilon value and estimate the number of clusters ($eps = 4.758$).

2. We use the Silhouette method to determine the minimum samples parameter ($MinimumSamples = 23$).

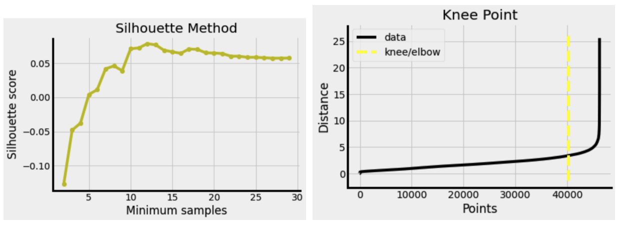

Figure 5. DBSCAN Graphs

It is a much faster clustering algorithm than spectral clustering, which is almost infeasible with our computational resources. The number of noise points were 2862 and 14 clusters according to Eigengap Heuristics.

5. Result and Analysis

The pre-processing and feature extraction provided a more intensive quality dataset. We created a detailed pre-processing pipeline and a custom one-hot encoder for multiple classes on a single data point. The data is categorical, high-dimensioned and sparse and contains outliers. Some of the dataset's limitations are that most movies are in English, and most directors have done 1-2 movies. After exploration of clustering algorithms, we successfully identified DBSCAN for clustering in the case of content-based filtering in terms of time, computational power and predictions.

6. Conclusions

The project helped us in two domains - soft and technical skills development. Regarding technical skills, we understood the significance of pre-processing in machine learning and explored various techniques like encoding in feature extraction. We further explored algorithms for movie recommendations based on K-means clustering. In soft skills, we learnt the importance of teamwork, collaboration and empathy. We are currently on our proposed timeline and have completed our work till week 7, which is feature extraction and basic model building. We will continue exploring more machine learning algorithms and present the results in the final submission. Vaibhav and Bhagesh worked on selecting the dataset, completing the literature survey and performing pre-processing on the dataset. Mohit and Ojus worked on Exploratory data analysis and model building, respectively.

References

[1] Rishabh Ahuja, Arun Solanki, and Anand Nayyar. Movie recommender system using k-means clustering and k-nearest neighbor. In *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pages 263–268, 2019. 1, 3

[2] Alexandru Niculescu-Mizil, Claudia Perlich, Grzegorz Swirszcz, Vikas Sindhwani, Yan Liu, Prem Melville, Dong Wang, Jing Xiao, Jianying Hu, Moninder Singh, Wei Xiong Shang, and Yan Feng Zhu. Winning the kdd cup orange challenge with ensemble selection. In Gideon Dror, Mar Boullé, Isabelle Guyon, Vincent Lemaire, and David Vogel, editors, *Proceedings of KDD-Cup 2009 Competition*, volume 7 of *Proceedings of Machine Learning Research*, pages 23–34, New York, New York, USA, 28 Jun 2009. PMLR. 3

[3] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001. 1, 3

[4] Jiang Zhang, Yufeng Wang, Zhiyuan Yuan, and Qun Jin. Personalized real-time movie recommendation system: Practical prototype and evaluation. *Tsinghua Science and Technology*, 25(2):180–191, 2020. 2