# Movie Recommendation System

Yashvi Pipaliya (AU1841092)
*Information and Communication Technology-BTech. 3rd Year*
*Ahmedabad University*
Ahmedabad. India
yashvi.p@ahduni.edu.in

Kesha Bagadia (AU1841011)
*Information and Communication Technology-BTech. 3rd Year*
*Ahmedabad University*
Ahmedabad, India
kesha.b@ahduni.edu.in

Yashvi Gandhi (AU1841033)
*Information and Communication Technology-BTech. 3rd Year*
*Ahmedabad University*
Ahmedabad, India
gandhi.p@ahduni.edu.in

Manal Shah (AU1841026)
*Information and Communication Technology-BTech. 3rd Year*
*Ahmedabad University*
Ahmedabad, India
manal.s@ahduni.edu.in

***Abstract*—Movie recommendation system is very useful in our social life because of its strength in providing entertainment. It helps movie enthusiasts by suggesting movies without them having to go through a process of choosing from a large set of movies thus, saving time and efforts. The project focuses on finding different model using content based recommendation system approach with the help of different algorithms.**

***Index Terms*—Data preprocessing, Exploratory Data Analysis, Random Forest Regressor, Feature importance, Clustering, Kernel density estimation, Cosine similarity, TF-IDF Vectoriztion, Count Vectorization**

## I. Introduction

As we know, there's an explosive amount of growth on the Internet in terms of digital information and the number of users. Due to this, there's an information overload hindering timely access of different items on the Internet. Therefore, there's an increasing demand for recommendation systems. They are information filtering systems which handle the problem of information overload by filtering out the important chunks of data from the available data according to user's preferences or interests about the item and provide them with personalized recommendations.

There are different filter-based approaches to building recommendation systems.We are doing a content-based movie recommendation system. It suggests a set of movies based on other movies of the user's preference. It filters out movies based on different attributes like genre, directors, countries etc. using three different methods 1) clustering and TF-IDF Vectorization 2) TF-IDF Vectorization 3) Count Verctorization. We then compare the results from all three methods and recommend the most probable.

These systems are beneficial for organizations that collect data from large number of customers and who wish to effectively provide the best suggestions possible.

## II. Literature survey

Movies are enjoyed by everyone, across age, gender, race, color and geography, it's a medium that connects us. But our choices, even so, remain different from others. That is where data scientists come in. They extract behavioral patterns of the audience and the movies to give required results[1]

A recommendation system suggests users a number of resources which can be anything like songs or books, with the basis of a data set. In our case, we will be working for movies. On the input of preference, it will give recommendations that the user is likely to enjoy.[2]

Amongst different types, a content based recommendation system mainly works with data provided by the user, extracted from a source, or inputted on some interface. And generally, based on the data, a profile is generated, which is then used to make suggestions to the user. With more inputs, it gets more accurate.[3]

The method to model this approach is the Vector Space Model (VSM). As the algorithm basically gives recommendations for products that are similar to the preferences of the user, it uses the computation of similarity.[1] This similarity of the movies is derived from its description, applying the concept of TF-IDF, which is Term Frequency-Inverse Document Frequency.[4]

TF is the frequency of a word in a document and IDF is the inverse of the document frequency, very much as the name suggests. TF-IDF operates in a manner such that the weighting negates the effect of high frequency words in determining the importance of an item. For example, if one was the to look up ”the decline of feudalism” on Google, it is certain that “the” will occur more frequently than “feudalism”, but the relative importance of the latter will be higher than the former from a search query point of view.[1]

Vectors generated from the above concepts are used to compute similarity. One way to do this is cosine similarity. It basically measures the angle of cosine between the two objects and compares them on a normalized scale. This is done by calculating the dot product of the two identities. And lesser the angle between the two vectors, more is the similarity. [4]

To make these computations, a number of features are selected from the given data. This is done using different methodologies of feature importance. Feature importance refers to a class of techniques for assigning scores to input features to a predictive model that indicates the relative importance of each feature when making a prediction. [5]

But content based recommenders have their own limitations. They are not good at capturing inter-dependencies or complex behaviors. But they can still function pretty well on optimisation. [3]

To improve on the results, we've added an additional filter that's clusters. Clustering takes the required data points and divides them into groups which have similar features and these groups are called clusters.[6] Here, movie scores can be calculated using the previously computed feature importances to create the clusters so that the model can further only suggest movies with the same or similar score.

Kernel Density Estimation is used to extract data of where a higher number of points are situated and vice versa. In 1D arrays, it assists with clustering by segregating low density and high density points.

Another method adopted for the same is a Count Vectorizer. It is used to create a matrix in which each unique word is represented by a column of the matrix and another column. It helps to filter content based on multiple features so as to give better results.

## III. Implementation

We first started with data gathering, which is where we collected and merged different data-sets to obtain the required parameters for recommendation and prediction. After that, we proceeded with data preprocessing and cleaning, which is where we detected and corrected the corrupt or inaccurate records from the database that may negatively impact a predictive model. We followed that with an Exploratory Data Analysis to identify obvious errors, get a better idea of the patterns within the data, detect outliers or anomalous events and find interesting relations among the variables. While performing EDA we observed various trends among different parameters as shown in Figure 1 which gave us a more detailed understanding of our finalised data set.

We then split the entire dataset into test and training data as per the 20/80 ratio. We used Label encoder to convert the string parameter to integer data type as most of our data was in the string format. Next, we used Random Forest method provided by python which is an ensemble classifier that uses multiple models of several Decision Trees to obtain a better prediction performance. It creates many classification trees and a bootstrap sample technique is used to train each tree from the set of training data. Using the Random Forest we obtain the accuracy of 0.901638 using the rf.score method, which is basically the $R^2$ value i.e a statistic that gives some information about the goodness of fit of a model. Finally, we calculated feature importance using RandomForestRegressor, Permutation feature importance and Drop Column feature importance to find ideal parameters. After comparing the results obtained by these feature importances, we removed the parameters with least importance. We concluded that the Drop


Figure 1. SNS Plot

Column feature importance showed the best performance, increasing the accuracy by 0.003 which gave us the value of 0.9045671.

Using the feature importances calculated by Drop column feature importance, we computed a score by multiplying the respective feature importance values with the features and then summed up the products. Thus we now have a parameter score for every movie in the dataset. Next, we plot the distribution of score and then apply Kernel Density Estimation from the range of the peak from the plot (Figure 2), that is, -1050 to 10,000.

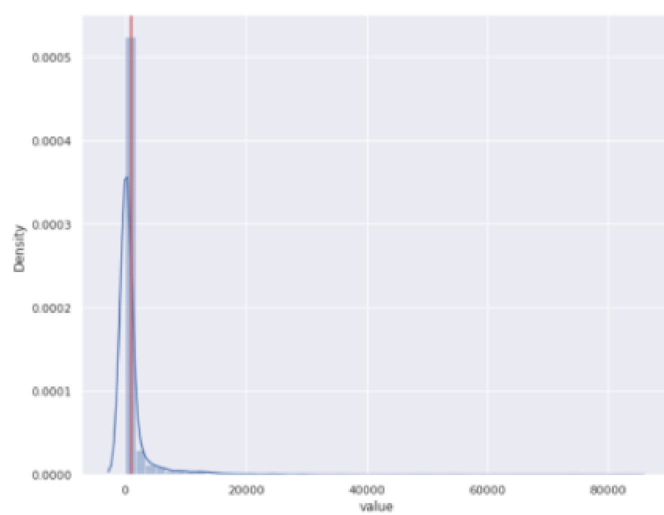



Figure 2. Movie 1 (KDE plot for distribution of "score" parameter)

After computing the density, we find the local maxima and minima values and form clusters based on these values. As there are 25 local maxima and local minima points, we obtain 25 clusters containing different movies. Once the clusters are formed, we move to computing TF-IDF vectorization matrix on the feature "genre" to obtain a similarity score for various movies using cosine similarity. Next, we define a function to get the input movie title from the user and based on the title, the function would compute the similarity score of the input movie to all the movies in the dataset and return the movies with similarity index greater than 0.7 from the same cluster or from the same director. Thus, a recommendation system using clustering is obtained.

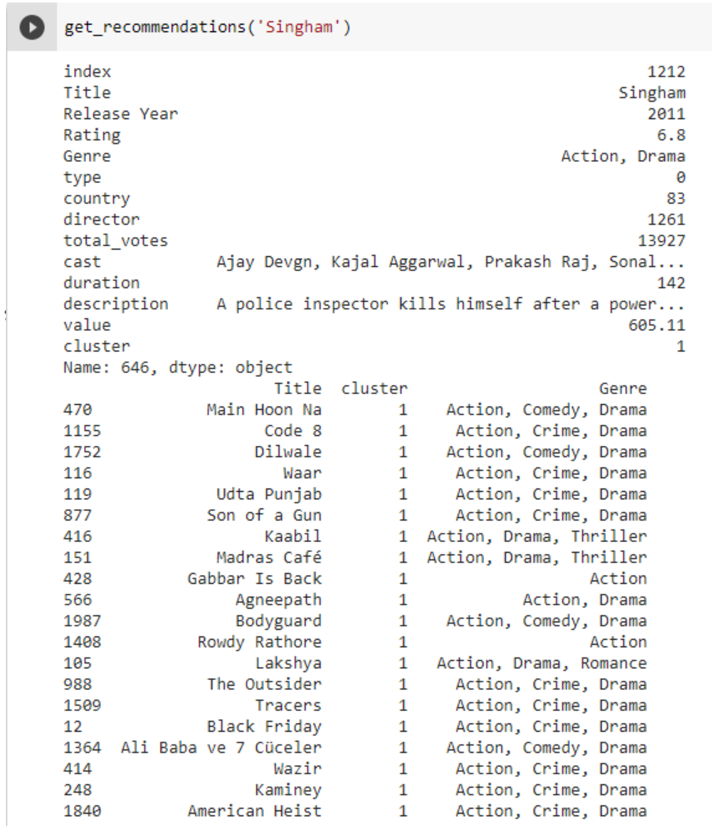
$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Cosine similarity formula

Now for the second approach, we built a single parameter based movie recommendation system by computing the TF-IDF vectorization matrix on the feature "description" and defined a function to compute the similarity score of the input movie with other movies in the dataset and return the top 10 movies with highest similarity score.

For the third type of recommendation system, we first chose top 5 features: genre, director, cast, country, and title. Following that, we conducted Data Prepossessing where we convert all the upper case data to lower case data and eliminate the unnecessary space and defined a new parameter "Bag of words" to combine all of the 5 parameters. After creating the BOW parameter, we compute the sparse matrix using count vectorization and cosine similarity. Lastly, we define a function which will compute the similarity score of the input movie to all the movies in the dataset using the sparse matrix. The function will return the top 10 movies with the highest similarity score as output. So, we now have a recommendation system using multiple parameters.

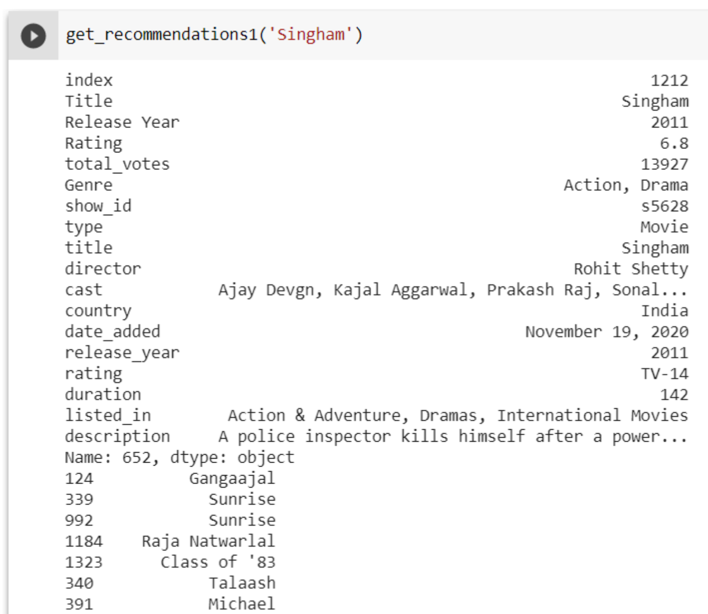
For the last type of recommendation system, we used clustering as well as count vectorization. Here, we took the clustered movie dataset obtained in the first type of recommendation system and then chose top 3 features, which are genre, cast and description and then followed the steps of the previous recommendation system. Now, using count vectorization and cosine similarity we computed the sparse matrix which helps in defining a function to calculate the similarity score of input movie with other movies in the clusters. The function will return all the movies with similarity score greater than 0.12. Here, we had to keep the limiting similarity score low as the number of parameters taken into consideration were higher and the number of movies in the cluster with similarity score similar to the input movie's score were less, due to which a problem like over-fitting may occur. But keeping the similarity score very less sometimes gives out extraneous movie recommendations as well.



```
get_recommendations('Singham')

index                                                      1212
Title                                                   Singham
Release Year                                               2011
Rating                                                      6.8
Genre                                             Action, Drama
type                                                          0
country                                                      83
director                                                   1261
total_votes                                               13927
cast           Ajay Devgn, Kajal Aggarwal, Prakash Raj, Sonal...
duration                                                    142
description    A police inspector kills himself after a power...
value                                                    605.11
cluster                                                       1
Name: 646, dtype: object
                     Title  cluster                    Genre
470            Main Hoon Na        1    Action, Comedy, Drama
1155                 Code 8        1     Action, Crime, Drama
1752                Dilwale        1    Action, Comedy, Drama
116                    Waar        1     Action, Crime, Drama
119             Udta Punjab        1     Action, Crime, Drama
877             Son of a Gun       1     Action, Crime, Drama
416                  Kaabil        1  Action, Drama, Thriller
151            Madras Café         1  Action, Drama, Thriller
428          Gabbar Is Back        1                   Action
566               Agneepath        1            Action, Drama
1987              Bodyguard        1    Action, Comedy, Drama
1408           Rowdy Rathore       1                   Action
105                 Lakshya        1   Action, Drama, Romance
988             The Outsider       1     Action, Crime, Drama
1509                Tracers        1     Action, Crime, Drama
12             Black Friday        1     Action, Crime, Drama
1364   Ali Baba ve 7 Cüceler       1    Action, Comedy, Drama
414                   Wazir        1     Action, Crime, Drama
248                 Kaminey        1     Action, Crime, Drama
1840          American Heist       1     Action, Crime, Drama
```

Figure 3. Model 1 (System using Clustering with TF-IDF Vectorization)



```
get_recommendations1('Singham')

index                                                      1212
Title                                                   Singham
Release Year                                               2011
Rating                                                      6.8
total_votes                                               13927
Genre                                             Action, Drama
show_id                                                   s5628
type                                                      Movie
title                                                   Singham
director                                           Rohit Shetty
cast           Ajay Devgn, Kajal Aggarwal, Prakash Raj, Sonal...
country                                                   India
date_added                                    November 19, 2020
release_year                                               2011
rating                                                    TV-14
duration                                                    142
listed_in       Action & Adventure, Dramas, International Movies
description    A police inspector kills himself after a power...
Name: 652, dtype: object
124           Gangaajal
339             Sunrise
992             Sunrise
1184    Raja Natwarlal
1323       Class of '83
340             Talaash
391             Michael
```

Figure 4. Model 2 (TF-IDF vectorization using single parameter)

### A. Results

## IV. Conclusion

After comparing the outputs of different models for different movies, we can conclude that the model using TF-IDF vectorization with a single parameter gives the worst performance amongst all the tested recommendation models.

We observed TF-IDF vectorization using clustering gives better results than the count vectorization method. We inferred

```
get_recommendations2('singham', similarity )

index                                                     1212
Title                                                  Singham
Release Year                                              2011
Rating                                                     6.8
total_votes                                              13927
Genre                                           Action, Drama
show_id                                                  s5628
type                                                     Movie
title                                                  Singham
director                                          Rohit Shetty
cast            Ajay Devgn, Kajal Aggarwal, Prakash Raj, Sonal...
country                                                  India
date_added                                   November 19, 2020
release_year                                              2011
rating                                                   TV-14
duration                                                   142
listed_in        Action & Adventure, Dramas, International Movies
description     A police inspector kills himself after a power...
Name: 652, dtype: object
254            Golmaal: Fun Unlimited
2092                       Himmatwala
274        Once Upon a Time in Mumbaai
61          The Legend of Bhagat Singh
1013                         Guna 369
257                          Apaharan
883                  Fakta Ladh Mhana
932                            Pitaah
1447             Singh Saab the Great
86                            Company
Name: Title, dtype: object
```

Figure 5. Model 3 (Count vectorization using multiple parameters)

```
get_recommendations3('Singham', similarity1)

level_0                                                    646
index                                                     1212
Title                                                  Singham
Release Year                                              2011
Rating                                                     6.8
Genre                                             action,drama
type                                                         0
country                                                     83
director                                                  1261
total_votes                                              13927
cast            Ajay Devgn, Kajal Aggarwal, Prakash Raj, Sonal...
duration                                                   142
description     apoliceinspectorkillshimselfafterapowerfulgang...
value                                                   605.11
cluster                                                      1
bag_of_words    action,drama apoliceinspectorkillshimselfafter...
Name: 646, dtype: object
                          Title  cluster                  Genre
47                        Black        1                  drama
49              Paan Singh Tomar        1  action,biography,crime
605                 Jersey Boys        1   biography,drama,music
1478              Hold the Dark        1     action,drama,horror
470                Main Hoon Na        1     action,comedy,drama
744              The Grandmaster        1  action,biography,drama
1155                      Code 8        1      action,crime,drama
1752                     Dilwale        1     action,comedy,drama
1649                       After        1          drama,romance
858                  The Endless        1    drama,fantasy,horror
35                      Drishyam        1    crime,drama,thriller
275               Organize Isler        1           comedy,drama
919                Ghost Stories        1           drama,horror
116                         Waar        1      action,crime,drama
1815                      Sliver        1         drama,thriller
227     Instructions Not Included        1           comedy,drama
70                         Dev.D        1          drama,romance
158                   Death Note        1     crime,drama,fantasy
```

Figure 6. Model 4 (System using Clustering with Count Vectorization)

this must be so because the chosen 5 parameters in the latter have the same importance while in the former, the parameter importance is calculated using the computed feature importance for all parameters, using which the clusters are formed. Moreover, we filter the movies in the clusters using TF-IDF vectorization on the most important parameter "genre" and thus, the results are more accurate.

The results obtained after combining clustering and count vectorization methods were observed to be less accurate compared to the combined model of clustering and TF-IDF cectorization. It happens because in the former system the higher number of parameters considered causes high specificity and less similarity and hence a low threshold similarity score which allows possibly and relatively irrelevant movie recommendations in the output.

| Model Name | With Clustering | TF-IDF Vectorization | Count Vectorization | Parameters considered for Vectorization | Parameters considered for Clustering | Performance |
|---|---|---|---|---|---|---|
| Recommendation Model 1 | ✓ | ✓ | ✗ | Genre | Total_votes, Duration, Director, Release Year, Country | **Best** |
| Recommendation Model 3 | ✗ | ✗ | ✓ | Title, Genre, Cast, Director, Country | NA | **Better** |
| Recommendation Model 4 | ✓ | ✗ | ✓ | Genre, Cast, Description | Total_votes, Duration, Director, Release Year, Country | **Good** |
| Recommendation Model 2 | ✗ | ✓ | ✗ | Description | NA | **Poor** |

Model 1 : System using Clustering with TF-IDF Vectorizer
Model 2 : System using only TF-IDF Vectorization
Model 3 : System using only Count Vectorization
Model 4 : System using Clustering with Count Vectorization

Project link :
https://github.com/YashviPipaliya/CSE523-Machine-Learning-Abraca-data

## V. References


[1] Das, S. (2020, November 24). Create Your Own Movie Movie Recommendation System. Analytics Vidhya.

[2] Reddy, S. (2019). Content-Based Movie Recommendation System Using Genre Correlation. SpringerLink.

[3] Das, S. (2015, September 24). Beginners Guide to learn about Content Based Recommender Engines. Analytics Vidhya.

[4] Movie Recommendation System using Cosine Similarity and KNN. (2020). International Journal of Engineering and Advanced Technology, 9(5), 556–559. https://doi.org/10.35940/ijeat.e9666.069520

[5] Brownlee, J. (2020, August 20). How to Calculate Feature Importance With Python. Machine Learning Mastery. https://machinelearningmastery.com/calculate-feature-importance-with-python/

[6] Conlen, B. (n.d.). Kernel density estimation. Retrieved April 10, 2021, from https://mathisonian.github.io/kde/