

Abstract

Every day a new multimedia streaming platform comes into the market but not all are successful. To be a successful multimedia streaming platform in the market providing the content for as low cost as possible isn't the only factor to be considered. Providing the end user what to view or watch next is really important. The better the suggestions are the best the application is. This is what a recommender system does it suggests the content based on patterns of others users in system. The pattern observation can be done based on different attributes like ratings, watch time, categories, content etc., Apart from selection of attributes it is also important to select a right approach for the finding out the suggestions. These types of suggestions are always user specific. This proposal is to create a Movie Recommender System using a Hybrid K-Means Clustering and Navie Bayes Classifier (HKMNB). Most of the time data will be unsupervised data. But supervised classifiers are always better and efficient, this approach will merge the unsupervised classifier with supervised classifier so that the final recommendations might be more efficient. The approach starts with collections of movies dataset and ratings dataset and once preprocessing is completed the data will be given KMeans to create the clusters and convert the data to supervised data. The supervised data is used to train the Navie Bayes model. This model is used to later to find out the patterns in the user watch history and provide the recommendations.

Keywords: *Multimedia streaming, K-Means, Navie Bayes, Recommender System, Preprocessing, Supervised, Unsupervised*

Table of Content

	Title	Pages
1.	Introduction	6 - 7
2.	Literature Reviews	8
3.	Methodology	9
3.1.	Algorithms	9 – 10
3.1.1.	K-Means Clustering Algorithm	
3.1.2.	Navie Bayes Classification Algorithm	
3.1.3.	Hybrid K-Means and Navie Bayes Algorithm	
3.2.	Environment Setup	10
3.2.1.	Hardware Requirements	
3.2.2.	Software Requirements	
3.3.	Dataset	10 – 11
3.3.1.	movies.csv	
3.3.2.	ratings.csv	
3.4.	Analysis	11 - 12
4.	Experimental Results	13
4.1.	Solution	13
4.2.	Results	13 - 15
4.2.1.	movies.csv	
4.2.2.	Web UI	
5.	Conclusions and Suggestions for Future Work	16
6.	Bibliography	17

List of Figures

Fig. No	Name	Page
1.1.	Navie Bayes Formula	6
3.1.	Architecture	9
3.2.	Working Model	9
3.3.	Scatter plot between movieId and rating fields	11
3.4.	Scatter plot between movieId and genre_rating fields	12
3.5.	Scatter plot between movieId and score fields	12
4.1.	Jupyter Notebook Running	13
4.2.	Webpage initial	14
4.3.	Webpage after adding watch history	14
4.4.	Webpage movie recommendations	15

List of Tables

Table No	Name	Page
3.1.	Movies Dataset	11
3.2.	Ratings Dataset	11
4.1.	Output Movies Dataset	13

1. Introduction

The amount of data generated by the web is growing at a huge rate. The very fact is that more products and services are available to end users now than ever has, not only compounded the matter but has also increased its scope and diversity. This presents a major challenge to e-commerce because consumers cannot simply explore and compare every possible product. Businesses are forced to return up with smarter solutions to prevent consumers from being confused by seemingly countless choices and more importantly to be still relevant to them by directing their attention to commodities they might be curious about. Recommendation systems (RSs) are introduced to e-commerce to assist businesses tackle this challenge. Among all the prevailing techniques, collaborative filtering is one among the foremost promising approaches to build RSs. This task is accomplished by searching the database for user profiles with similar tastes.

Businesses that sell a humongous assortment of products on a day to day both in world like Walmart, Target, etc. on their e-business find RS immensely useful. RSs help businesses gauge consumer interests and retain them by actively engaging them. The system provides recommendations supported the ratings on commodities that users have previously rated. A RSs is efficient since in practice the system handles tens of thousands of ratings and prediction is calculated in real-time. The more data the RS has got to work with, more accurate the predictions done by the system are. Recommender systems are often classified supported the approach won't to provide recommendations and therefore the technique used.

MovieLens [1] is a website which has these huge movies collection, along with the ratings collected from multiple users for each movie. Using these datasets RSs can build to recommend the movies. To improve the efficiency and increase the speed of the system a hybrid classifier is being used here. The dataset which has been collected will be processed so that it has the average rating of the individual movie along score i.e., numbers of users who has rated the movie and finally the average rating of each genre.

Processed dataset will be undergoing clustering process using KMeans [2] clustering algorithm. It uses recursion method to find n centroids for the given data. In this movie's dataset 100 clusters are required since the score of the movies will be in between from 0.1 to 10.0 every time. One the clustering is done the cluster value indicates the category of the movie. A movie might be on multiple clusters based on average genre rating and score.

K-Means generated output is used to train the Navie Bayes [3] model. This model can be used to determine the patterns of any user's watch history and recommend movies to watch next. And one more thing to consider here is Navie Bayes only accept continuous data as input and provides output of nominal data.

The diagram illustrates the Navie Bayes formula with arrows indicating the components of the equation:

$$P(c | x) = \frac{P(x | c) P(c)}{P(x)}$$

Labels and arrows:

- Likelihood** points to $P(x | c)$
- Class Prior Probability** points to $P(c)$
- Posterior Probability** points to $P(c | x)$
- Predictor Prior Probability** points to $P(x)$

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

Figure 1.1 Navie Bayes Formula

A python webserver created using Flask [4] package will be providing a REST API which takes input of user's watch history and processes it using the trained Navie Bayes model to recommend the Movies. A webpage is used an interface to capture the user's input and display the output.

2. Literature Review

Collaborative filtering is the better approach for movie RSs. Based on candidate user's purchase history, the ratings of things for a target user are predicted. A collaborative filtering RS adopts the nearest neighbor approach. Neighbors are chosen based on the similarity of item ratings using collaborative filtering. [6]

Clustering models operate on the concept of splitting the sample space into clusters. These models are widely used in scalability problems while still delivering recommendations of comparable efficiency. The aim of the clustering models is to partition items into clusters in such a way that the objects will have the minimum distance between each of them. [7]

Content-based algorithms approach the task of making suggestions as a query problem. They look at the history of ratings, descriptions of products by the user and products the user has searched for. All the search queries that are made in the system for similar items are found and products that have similar attributes like same music band or author or director are recommended to the user. One major challenge this approach faces is that the recommendations made are too generic or too narrowed down and predictable. As there are millions of queries made, it is almost impossible to look at all the queries. In order to remedy that only one or more subsets of all the queries made are looked at. Even then, the overall recommendation quality is relatively poor. [8]

The giant success of the e-commerce ventures can be greatly attributed to recommender systems. As a result, they are used extensively in the e-commerce industry. Recommender systems are used to generate highly personalized suggestions for books at Amazon, movies at MovieLens and Netflix. They help filter out humongous masses of irrelevant information, also called noise, from the information that can be leveraged to make highly intelligent suggestions and business decisions. [9]

Also, with the increase in e-commerce, there has been a collection of staggering amounts of user data. Recommender systems make intelligent use of big data and also helps tackle the growingly intimidating information overload. A lot of academic work has also been done across the world and domain literature has also flourished. Universities are offering recommender systems courses, and an annual conference dedicated only to the subject is also held. The scope of recommender systems has also widened. Even though, initial work was almost entirely done in collaborative filtering, it has grown to accommodate a host of content-based and knowledge-based methods. While such diversity in systems are significant, recommender systems that are based on collaborative filtering are of main interest. [10] [11]

3. Methodology

This proposed Movie Recommender System uses a Hybrid Classifier consisting of KMeans and Navie Bayes. KMeans is an unsupervised based clustering algorithm whereas Navie Bayes is an supervised based classification algorithm. The both combined to form a hybrid classifier working to find out the patterns in the users movie's watch history and recommend movies for them based on the patterns.

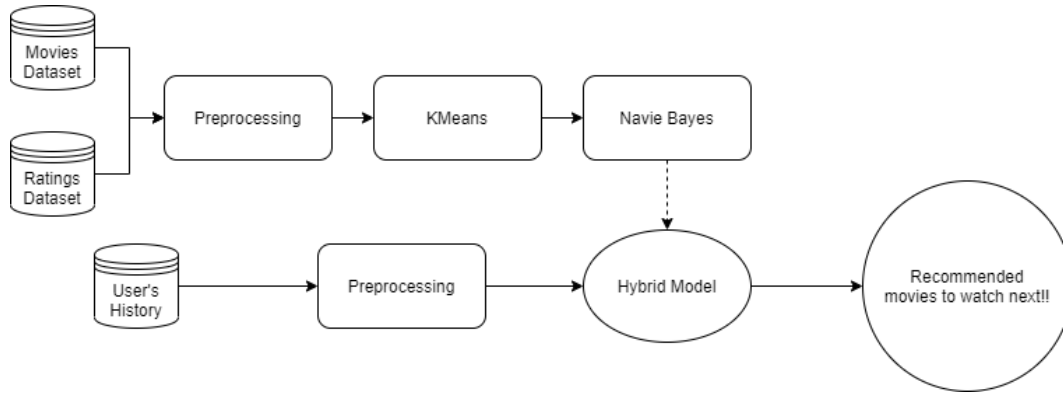


Figure 3.1 Architecture

The trained Navie Bayes model is used in the webservice to provide the movie recommendations in the real time to the users. The webservice consists of two endpoints. One of the endpoints is root which provides a web-based interface for user to provide input and view the output. The other endpoint is a REST API taking input as users watch history and processes it using Navie Bayes model and sends back the output.

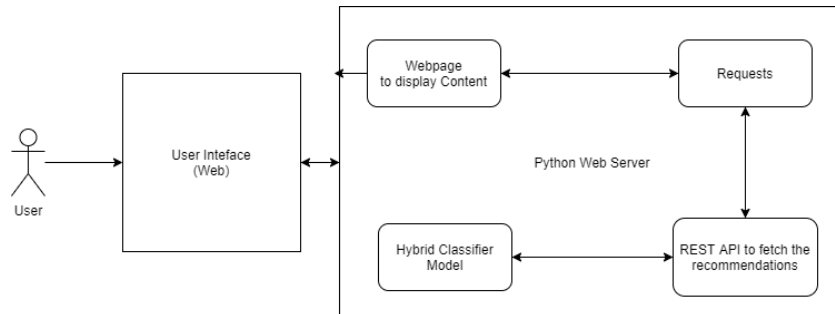


Figure 3.2 Working Model

3.1. Algorithm

3.1.1. KMeans Clustering Algorithm

1. Select the 'k' value i.e., the number of clusters you want to identify.
2. Choose randomly 'k' data points as centroids (c1, c2..., ck) from the vector space.
3. Repeat until convergence:
 - a. for each data point xi:
 - i. find the nearest centroid cj using Euclidian distance.
 - ii. assign xi to that nearest cluster j.
 - b. for each cluster find the new centroid which is the mean of all the xi points assigned to that cluster j.

4. Terminate when none of the cluster assignment change.
- 3.1.2. Navie Bayes Classification Algorithm
1. Convert the data into frequencies
 2. Find the probabilities of the all possible outcomes using the frequencies
 3. Now using Navie Bayesian equation calculate the posterior probability
- 3.1.3. Hybrid K-Means and Navie Bayes Algorithm
1. Preprocess the dataset
 2. Select the 'k' value i.e., the number of clusters you want to identify.
 3. Choose randomly 'k' data points as centroids (c1, c2..., ck) from the vector space.
 4. Repeat until convergence:
 - a. for each data point xi:
 - i. find the nearest centroid cj using Euclidian distance.
 - ii. assign xi to that nearest cluster j.
 - b. for each cluster find the new centroid which is the mean of all the xi points assigned to that cluster j.
 5. Terminate when none of the cluster assignment change.
 6. Convert the data into frequencies
 7. Find the probabilities of the all possible outcomes using the frequencies
 8. Using Navie Bayesian equation calculate the posterior probability.
 9. Use the test data to predict the categories
 10. Find the most repeated category in the predict data.
 11. List top 10 of them as the suggested or recommended
- 3.2. Environment Setup
- 3.2.1. Hardware Requirements
- CPU with minimum of 4 physical cores
 - RAM with minimum of 8GB
 - Hard Disk
 - Operating System (Windows, Linux, Mac)
- 3.2.2. Software Requirements
- Python 3.7.x version
 - Python packages required numpy, sklearn, pandas, jupyter, Flask
 - Chrome Browser
- 3.3. Dataset

Dataset has been gathered from the MovieLens website. There are two files considered from entire sources.

3.3.1. movies.csv

This file consists of three fields movieId, title, genres.

Table 3.1 Movies Dataset

movieId	title	genres
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	Jumanji (1995)	Adventure Children Fantasy
3	Grumpier Old Men (1995)	Comedy Romance
4	Waiting to Exhale (1995)	Comedy Drama Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action Crime Thriller

3.3.2. ratings.csv

This file consists of four fields userId, movieId, rating, timestamp.

Table 3.2 Ratings Dataset

userId	movieId	rating	timestamp
1	1	4	964982703
1	3	4	964981247
1	6	4	964982224
1	47	5	964983815
1	50	5	964982931
1	70	3	964982400

3.4. Analysis

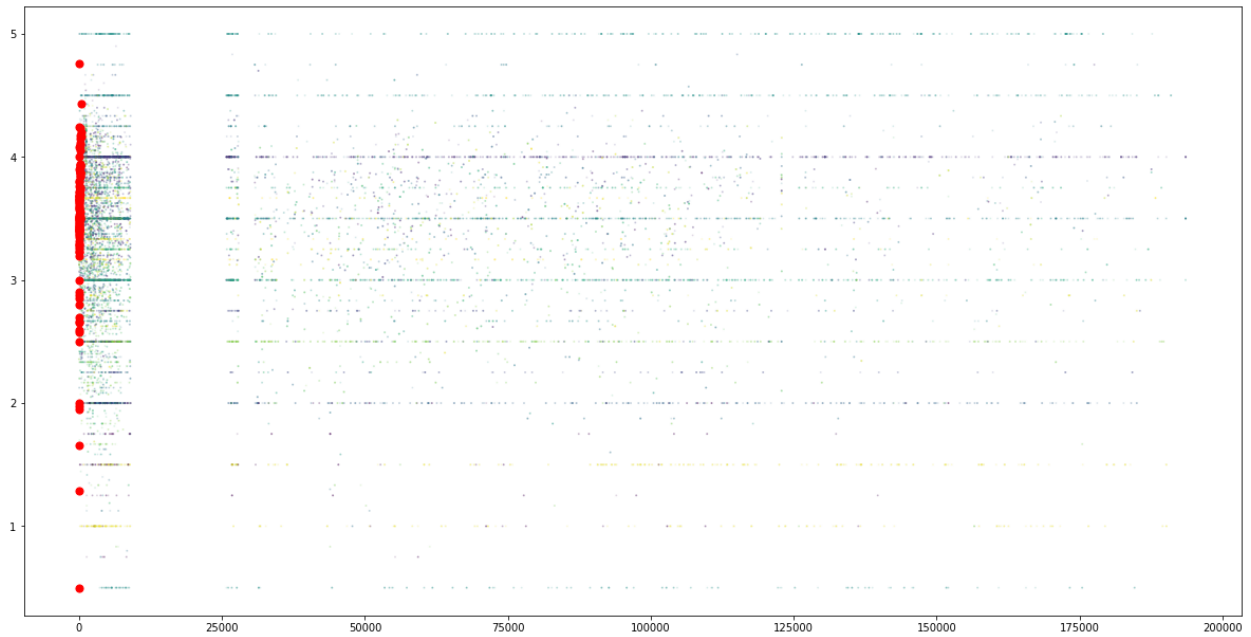


Figure 3.3 Scatter plot between movieId and rating fields

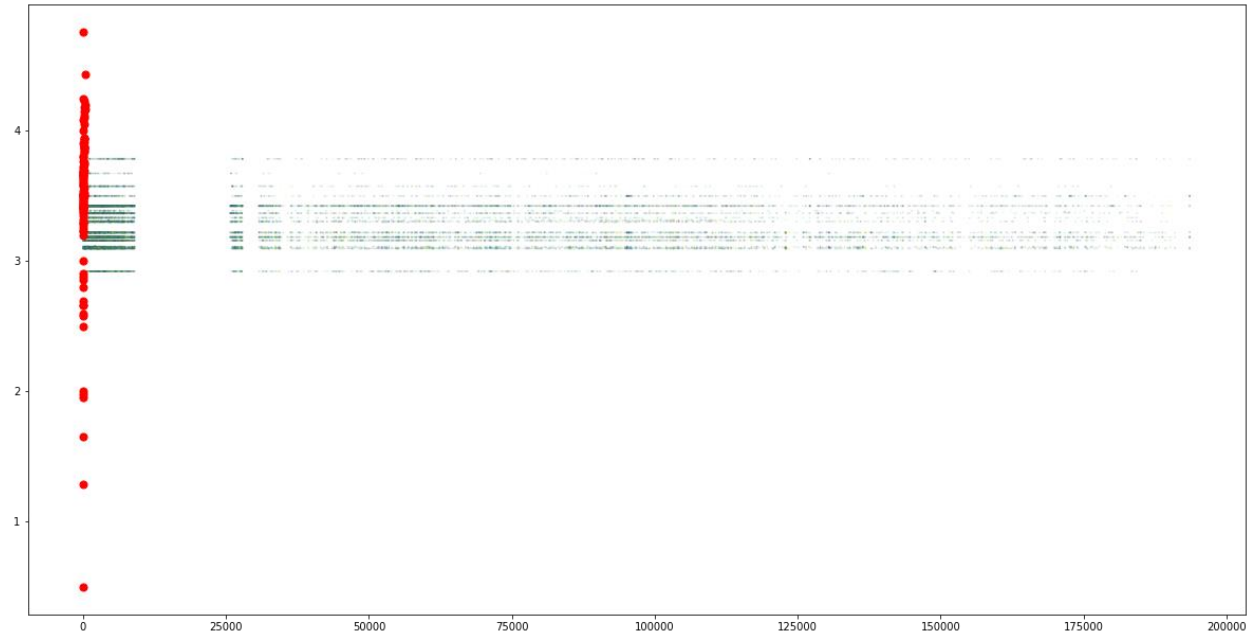


Figure 3.4 Scatter plot between movieId and genre_rating fields

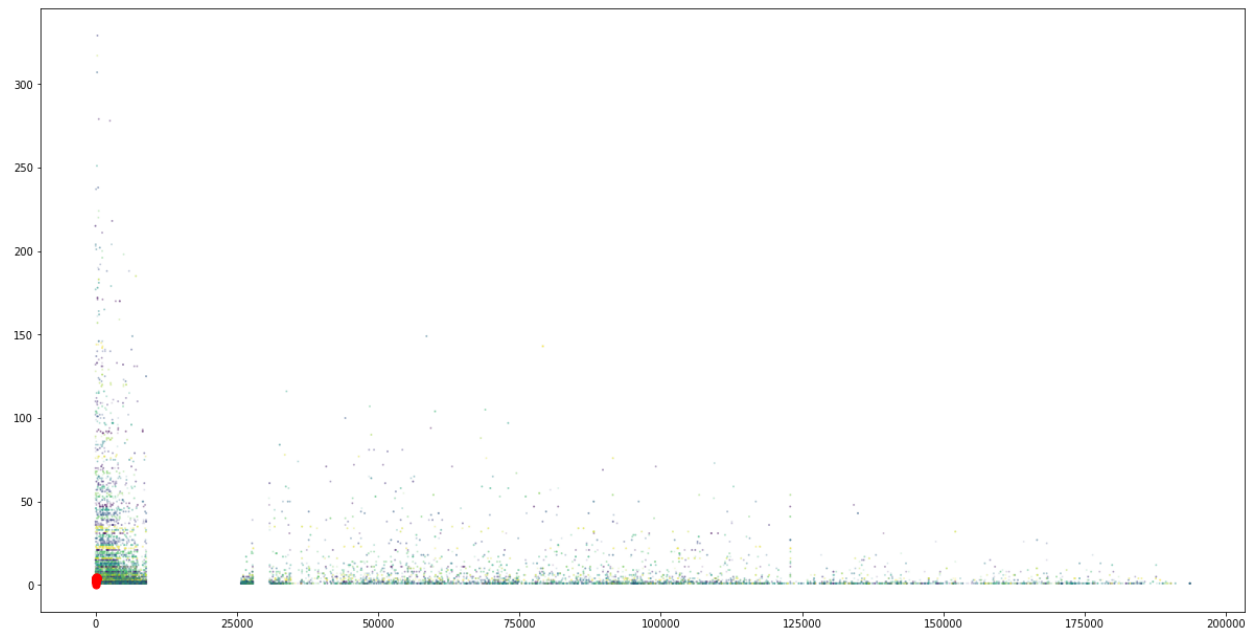
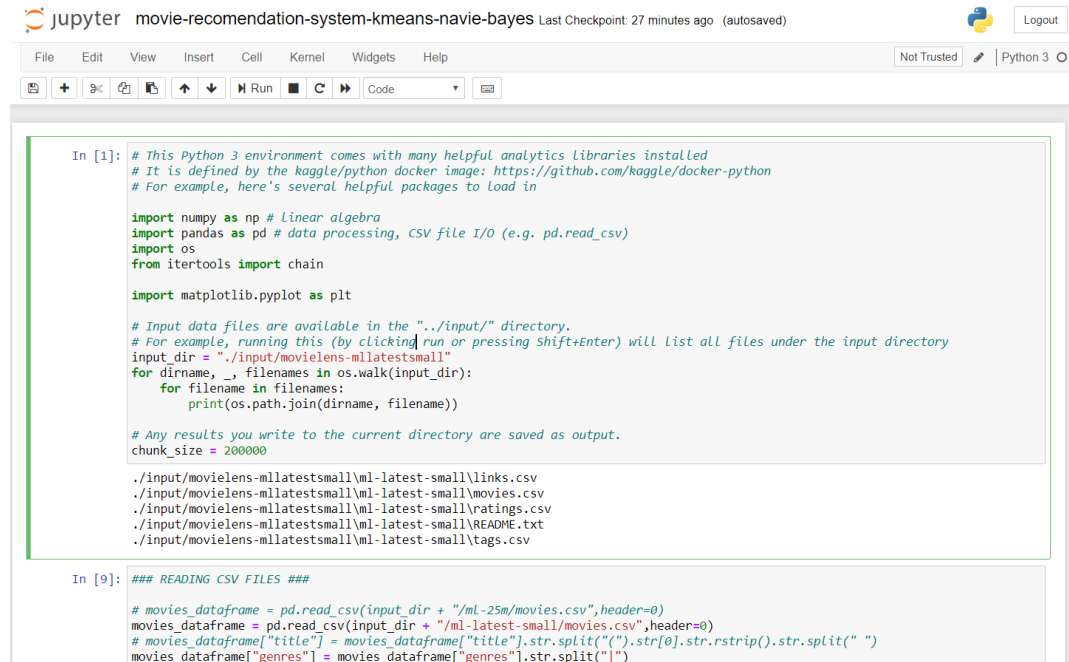


Figure 3.5 Scatter plot between movieId and score fields

4. Experimental Results

4.1. Implementation

- Open a command prompt in the directory where the notebook file is located and run the command “*jupyter notebook*”. This starts a jupyter server where you can run the python code.



```
In [1]: # This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
from itertools import chain

import matplotlib.pyplot as plt

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory
input_dir = "../input/movielens-mlatestsmall"
for dirname, _, filenames in os.walk(input_dir):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
chunk_size = 200000

./input/movielens-mlatestsmall/ml-latest-small\links.csv
./input/movielens-mlatestsmall/ml-latest-small\movies.csv
./input/movielens-mlatestsmall/ml-latest-small\ratings.csv
./input/movielens-mlatestsmall/ml-latest-small\README.txt
./input/movielens-mlatestsmall/ml-latest-small\tags.csv

In [9]: ### READING CSV FILES ###

# movies_dataframe = pd.read_csv(input_dir + "/ml-25m/movies.csv",header=0)
movies_dataframe = pd.read_csv(input_dir + "/ml-latest-small/movies.csv",header=0)
# movies_dataframe["title"] = movies_dataframe["title"].str.split("(").str[0].str.rstrip().str.split(" ")
movies_dataframe["genres"] = movies_dataframe["genres"].str.split("|")
```

Figure 4.1 Jupyter Notebook Running

- Run the cells to run the code in them. Once everything is completed a file named “./output/movies.csv” will be generated. Which is the output generated from the KMeans clustering process
- Provide the “./output/movies.csv” as input for the python webserver program using the command “*flask run*”.

4.2. Results

4.2.1. movies.csv

Table 4.1 Output Movies Dataset

movieId	score	rating	title	genre	genre_rating	category
1	215	3.920930233	Toy Story (1995)	Adventure	3.215229808	9
2	110	3.431818182	Jumanji (1995)	Adventure	3.215229808	13
8	8	2.875	Tom and Huck (1995)	Adventure	3.215229808	68
10	132	3.496212121	GoldenEye (1995)	Adventure	3.215229808	7
13	8	3.125	Balto (1995)	Adventure	3.215229808	68
15	13	3	Cutthroat Island (1995)	Adventure	3.215229808	41
			City of Lost Children, The (CitÃ© des			
29	38	4.013157895	enfants perdus, La) (1995)	Adventure	3.215229808	20

4.2.2. Web UI

- Open the url “localhost:5000” on any browser

(500) Days of Summer (2009)

ADD

PREVIOUSLY WATCHED!!

Movie Id	Title	Genres
----------	-------	--------

GET RECOMMENDATIONS

NEXT TO WATCH!!

Figure 4.2 Webpage initial

- Select the movies user have previously watched and click on “ADD” button

I, Robot (2004)

ADD

PREVIOUSLY WATCHED!!

Movie Id	Title	Genres
73741	Ninja (2009)	Action Crime Drama Thriller
32031	Robots (2005)	Adventure Animation Children Comedy Fantasy Sci-Fi IMAX
5803	I Spy (2002)	Action Adventure Comedy Crime
8644	I, Robot (2004)	Action Adventure Sci-Fi Thriller

GET RECOMMENDATIONS

Figure 4.3 Webpage after adding watch history

- Then click on “GET RECOMMENDATIONS” to fetch the suggestions

NEXT TO WATCH!!

- Fight Club (1999)
- Dark Knight, The (2008)
- Princess Bride, The (1987)
- Star Wars: Episode IV - A New Hope (1977)
- Apocalypse Now (1979)
- Star Wars: Episode V - The Empire Strikes Back (1980)
- Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)
- Matrix, The (1999)
- Monty Python and the Holy Grail (1975)
- Saving Private Ryan (1998)

Figure 4.4 Webpage movie recommendations

5. Conclusions and Suggestions for Future Work

In this paper a hybrid recommendation system is being developed using K-Means Clustering and Navie Bayes. In the proposed approach, metrics like rating of movie, genre and total ratings for movie are used. This makes the approach independent of the user and focus on the patterns. Since the recommendations are based on the frequencies of the rating provided on movies and genres they are not always satisfied to the user. Past based recommendation is not always better approach because there are different factors which alter user's interest.

The efficiency can be boosted if the attributes like gender, age, location etc., are added in the analysis. Link with social media platforms can help the system to better understand the patterns of the user and provide better recommendations. If the proposed can be evolved to find the possible interests not just based on watch history then it might increase the efficiency.

6. Bibliography

1. MovieLens Datasets, <https://grouplens.org/datasets/movielens/>
2. K-Means Clustering, https://en.wikipedia.org/wiki/K-means_clustering
3. Navie Bayes Classification, https://en.wikipedia.org/wiki/Naive_Bayes_classifier
4. Python Flask, <https://pypi.org/project/Flask/>
5. Developers@Work, <https://developerswork.online/>
6. Wang, Z., Yub, X., Feng, N. and Wang, Z. (2014) ‘An improved collaborative movie recommendation system using computational intelligence’, Journal of Visual Languages and Computing, Vol. 25, No. 6, pp.667–675.
7. McNee, S.M., Riedl, J. and Konstan, J.A. (2006) ‘Being accurate is not enough: how accuracy metrics have hurt recommender systems’, ACM CHI ‘06 Extended Abstracts, ACM, pp.1103–1108.
8. Fields, B., Rhodes, C. and d’Inverno, M. (2010) ‘Using song social tags and topic models to describe and compare playlists’, Workshop on Music Recommendation and Discovery 2010, CEUR, September, p.633.
9. Bennett, J. and Lanning, S. (2007) ‘The netflix prize’, KDD Cup and Workshop ‘07.
10. Guy, I., Zwerdling, N., Carmel, D., Ronen, I., Uziel, E., Yogev, S. and OfekKoifman, S. (2009) ‘Personalized recommendation of social software items based on social relations’, ACM RecSys ‘09, ACM, pp.53–60.
11. Martin, F.J. (2009) ‘RecSys ‘09 industrial keynote: top 10 lessons learned deploying and operating real-world recommender systems developing’, ACM RecSys ‘09, ACM, pp.1–2.