**BHARTI VIDYAPEETH DEEMED UNIVERSITY**

**AMPLIFY MINDWARE**



**A**

**Major Project-Report**

**On**

**Subject: Content-Based Movie Recommender System**

**SUBMITTED BY**

**Name: Pratik Gaikwad**

**Roll No: BSIT/17/006**

**Exam Seat No: 2015400004**

**2019 – 2020**

Project Completion Certificate

This is to certify that the project report entitled as Content-Based Movie Recommendation System(with Sentiment Analysis).

Submitted by:

Pratik Gaikwad

BSIT/17/006

Academic Year: 2019-2020

The student under the supervision of Mr Bhaskar Anand and Mrs Chanda Hirway , carries out this bonafide work and it is submitted towards the partial fulfillment of the requirement of BVDU-Amplify- DITM, Pune.

Internal Guide Course Coordinator Head of Department

Prof. Prashant Hinduja

Seal/Stamp of College Principal

Place: Pune (BVDU-Amplify- DITM, Pune)

Date:

**ACKNOWLEDGEMENT**

In the accomplishment of this project successfully, many people have best owned upon me their blessing and the heart pledged support, this time I am utilizing to thank all people who have been concerned with this project.

Primarily I would like to thank god for being able to complete this project with success. Then I would like thank my teachers Mr. Bhaskar Anand and Mrs. Chanda for giving me the idea of this excellent topic and helping me throughout the project. I would also like to thank r/machinelearning community, whose valuable guidance has been the ones that helped me patch this project and make it full proof success. Their suggestions and his instructions have served as the major contributor towards the completion of this project.

I greatly acknowledge all the support provided by my friends who helped me to accomplish this project, their valuable suggestions have been very helpful in various phases of the completion of the project. Last but not the least I would like to thank my family members for encouraging and supporting me throughout this Journey.

**ABSTRACT**

Recommender systems have emerged as the essential part of many e-commerce web sites. These systems provide personalized services to assist users in finding favourite items among the huge number of available media on the World Wide Web. Identifying temporal preferences of individuals is one of the major challenges of recommender systems to provide personalization for users. In this project, a content-based movie recommender system is proposed that captures the temporal user preferences in user modelling and predicts the preferred movies. Content Based Recommender System recommends movies similar to the movie user likes and analyses the sentiments on the reviews given by the user for that movie. A movie recommendation is important in our social life due to its strength in providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although, a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the existing users efficiently or to a new user by any means. This is where content based systems come into play.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page no.** |
| 1. | **Introduction** | 7 |
| 2. | 2.1 Overview  2.2 Work Proposal | 8  9 |
| 3. | **Scope of Work**  3.1 Objectives  3.2 Technology and Associated Platforms | 10  12  13 |
| 4. | **Feasibility Analysis** | 14 |
| 5. | **UML/ERD/Process flow/DFD’s** | 16 |
| 6. | **Implementation**  6.1 Theoretical Implementation  6.2 Snippets of Code/Wireframes/Actual System and PYNB ‘s | 19  20  21 |
| 7. | **Testing**  7.1 Case/Plan/Outcome | 36 |
| 8. | **Future Scope** | 37 |
| 9. | **References** | 37 |

|  |  |
| --- | --- |
| Terms | Definitions |
| Recommendation System | Information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. |
| Data science | Utilization of scientific methods, processes, algorithms and systems to extract knowledge and insights from data. |
| Data Cleaning | Process of identifying and removing (or correcting) inaccurate records from a dataset, table, or *database.* |
| Front end (Presentation layer) | Data represented to the user in form of a GUI (Graphical User Interface). |
| Back end (Data access layer) | The underlying structure handling the business logic and data storage. |
| Machine learning | Category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. |
| NLP (Natural Language Processing) | Interactions between computers and human languages, in particular how to process and analyse large amounts of natural data. |
| Sentiment Analysis | Interpretation and classification of emotions (positive, negative and neutral) within text data using text **analysis** techniques |
| API | A computing interface which defines interactions between multiple software intermediaries |

# **Chapter 1**

## 1. Introduction

Recommendation systems are becoming increasingly important in today’s extremely busy world. People are always short on time with the myriad tasks they need to accomplish in the limited 24 hours. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources.

The purpose of a recommendation system basically is to search for content that would be interesting to an individual. Moreover, it involves a number of factors to create personalised lists of useful and interesting content specific to each user/individual. Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. This is achieved through predictive modelling and heuristics with the data available.

# **Chapter 2**

**2.1 Overview**

This system is a content based movie recommendation web application with sentiment analysis that recommends movies similar to the movie user likes and analyses the sentiments on the reviews given by the user for that movie.

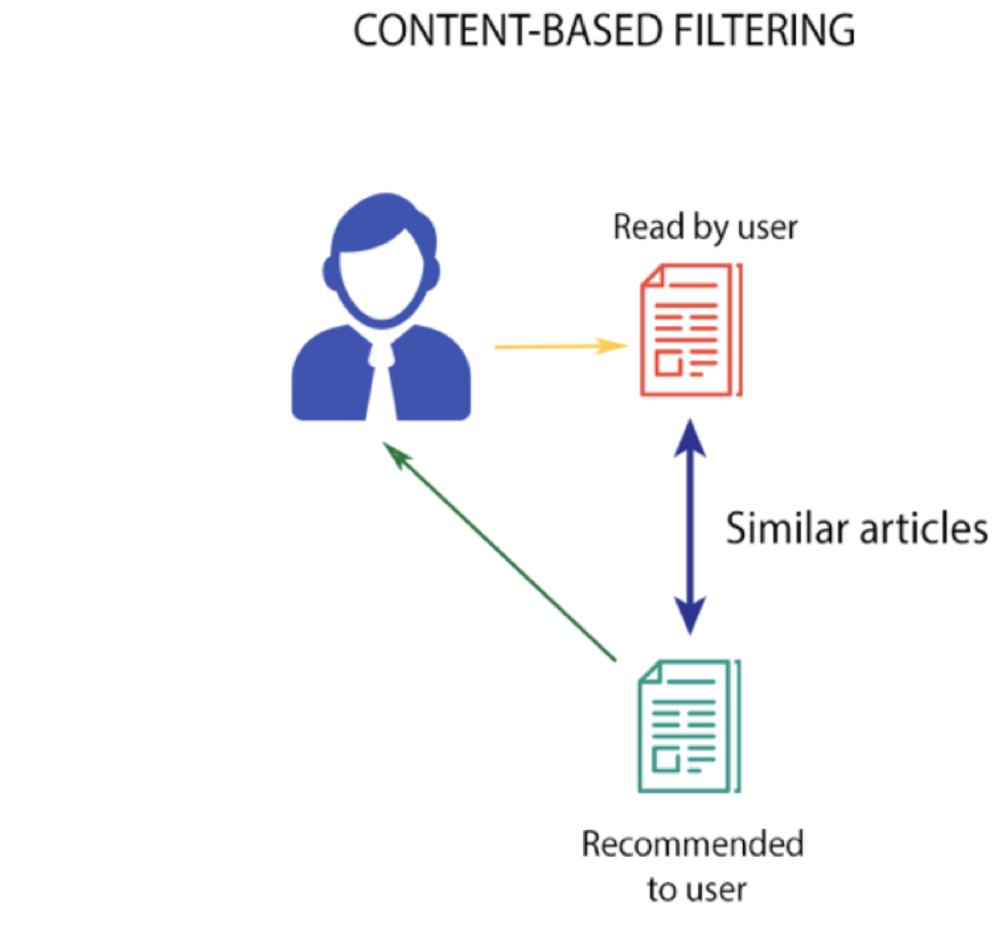
The application is developed using python’s ‘**Flask’** framework. The details of the movies (title, genre, runtime, rating, poster, etc.) were fetched using an API by TMDB(The Movie DB), and using IMDB id of the movie in the API, also **Web Scraping** was done to get reviews given by the user in the IMDB site using ‘beautifulsoup4’, performed sentiment analysis and displayed the corresponding emoji.

The Overview

Our system comprises two separate parts: a front-end and

aback-end. The front-end consists of a Raspberry Pi

‘Cosine Similarity’ is the primary machine learning algorithm used for implementing content-based filtering.

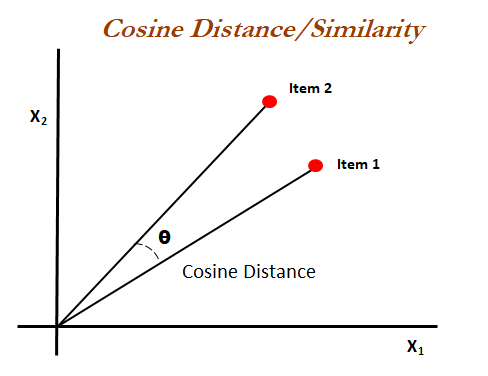
****

**2.2 Work Proposal**

The application uses similarity scores to decide which item is most similar to the item user likes.

It is a numerical value ranges between zero to one which helps to determine how much two items are similar to each other on a scale of zero to one. This similarity score is obtained measuring the similarity between the text details of both of the items. So, similarity score is the measure of similarity between given text details of two items. This can be done by cosine-similarity.

**Cosine similarity** is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.



**Chapter 3**

**3.1 Scope of Work**

The project is divided into two halves which include:

1. **Building the Machine Learning Model**

In the proposed system content-based filtering model was implemented for making recommendations based on the similarity score.

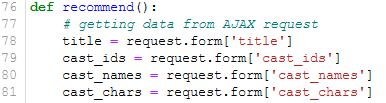
The algorithm **‘Cosine Similarity’** is implemented upon parameters of the final dataset for producing accurate suggestion of similar movies.

In an collaborative model the algorith would be K-NN(Nearest Neighbours).

1. **Consumption on front-end**

The framework used for consumption of ML model on web is Flask. It is a micro-web framework written in Python which helps in implementing a machine learning application that can be easily plugged, extended and deployed as a web application.

Final implementation of the web app is done by embedding flask into the main.py file and using AjaxRequest to call the function.

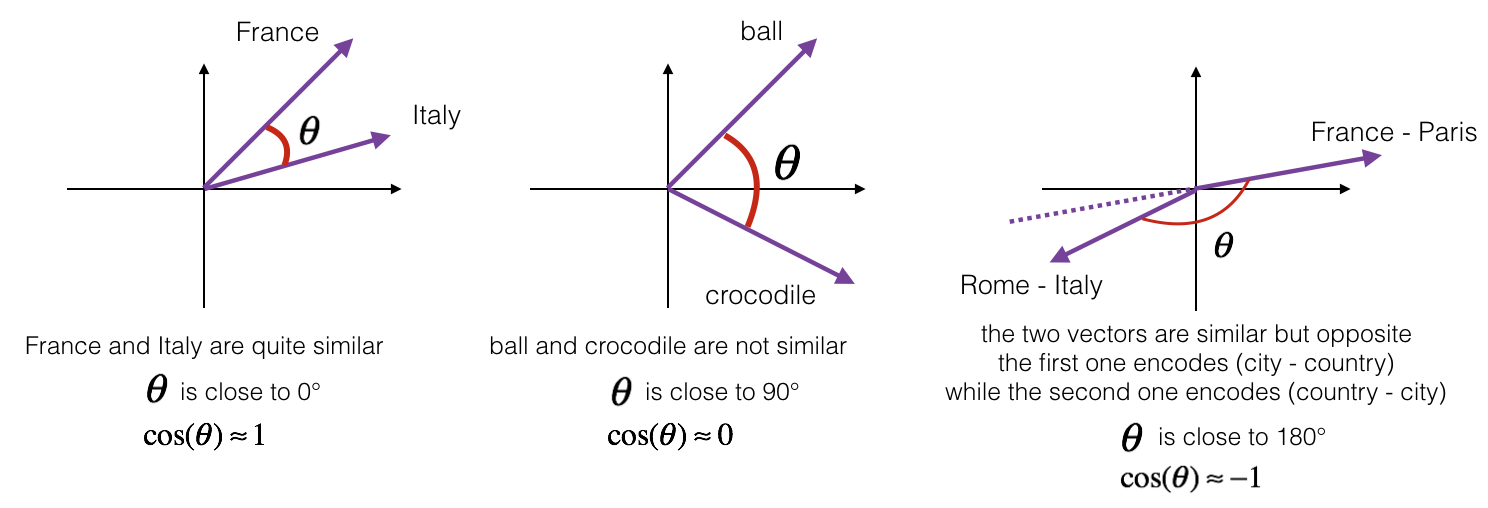


## Similarity Score:

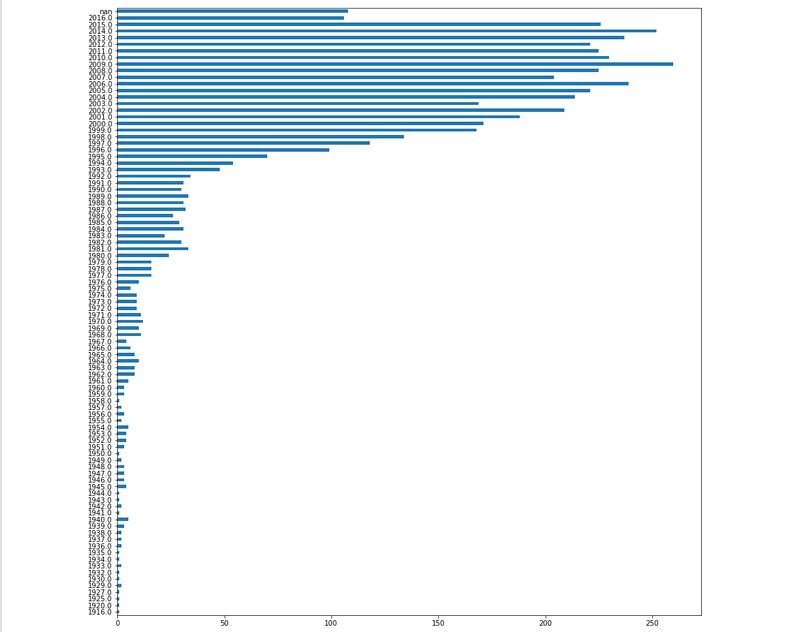
How does it decide which item is most similar to the item user likes? Here we use the similarity scores.

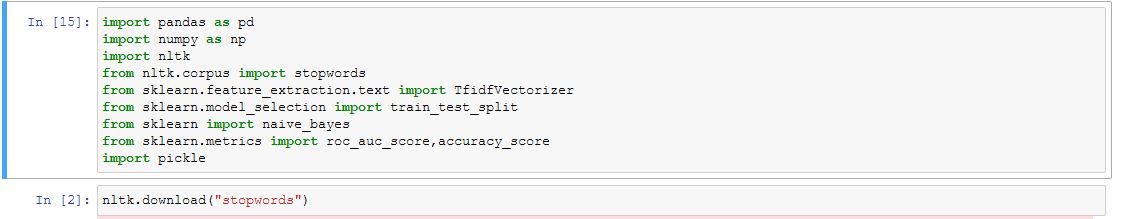
It is a numerical value ranges between zero to one which helps to determine how much two items are similar to each other on a scale of zero to one. This similarity score is obtained measuring the similarity between the text details of both of the items. So, similarity score is the measure of similarity between given text details of two items. This can be done by cosine-similarity

Example:



## 3.1 Objectives

* The primary objective is finding suitable datasets which consist historical information of movies from all time, cleaning them (dropping NaN’s, remove nulls and special chars, filter further for essential information) and applying the cosine similarity algorithm which will help analyze and suggest similar movies based on the similarity score which the user might like.
* The secondary objective is to train the NLP model to carry out sentiment Analysis based on user reviews.

****

**NLTK library for training** Natural Language Processing Model to perform Sentiment Analysis.



Converting all the information into vector form using TFIDF Vectorizer to create NLP Model.

## 3.2 Technology and Associated Platforms

* **HARDWARE REQUIREMENTS(minimum)**

Processor Intel Pentium/Core –

1.7GHz and above

Memory 1GB and above

Storage 2GB minimum fee space Graphics 1GB and above

* **SOFTWARE REQUIREMENTS**

**Framework(s):** Flask**,** Jupyter Notebook

**Libraries:** numpy, scipy, nltk, scikit-learn, pandas, beautifulsoup4, tmdbv3api, lxml, requests.

**Operating System(s):** Windows 7,8,10 / Linux

**Programming Languages:** Python, HTML, Javascript(JSON, AJAX\_REQUEST)

**Terminal:** cmd/gnome **, API:** TheMovieDB (TMDB API)

**Chapter 4**

**4.1 Feasibility Analysis**

• Types of Feasibility Conducted:

1. Technical feasibility:

The project was experimental in nature as I was not familiar with the Python programming language and its extensive range of open source libraries.

However due to the robust and flexible nature of Python aided by the open source nature of the respective software libraries we were able to utilize the tools effectively to a good extent.

2. Scheduling feasibility:

The time frame allotted for data gathering, cleaning and integration for final use creeped into the time frame allotted to Front-End and UI development, yet somehow I managed to come up with a pretty decent design and flow. The tools and libraries which made the job easier were found along and really made a lot of difference in keeping up with the schedule.

While Content based filtering uses: Cosine Similarity,

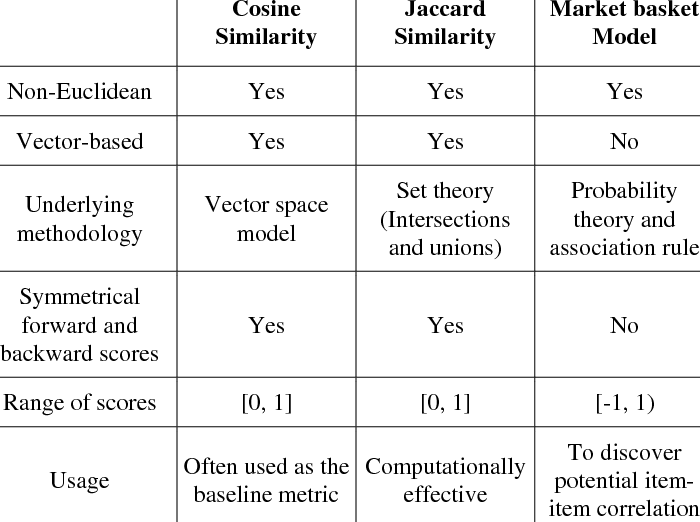
Collaborative Filtering can use: K-Nearest Neighbours,

Both are incomparable as they are different methods of recommendation.

Alternative to k-NN is K-Means:



Alternatives to Cosine similarity are:



**Chapter 5**

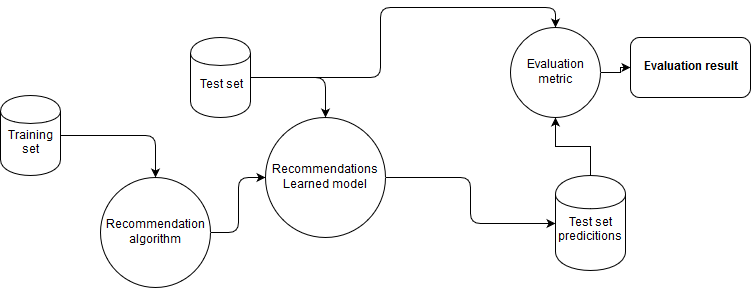
**System Architecture and Design**

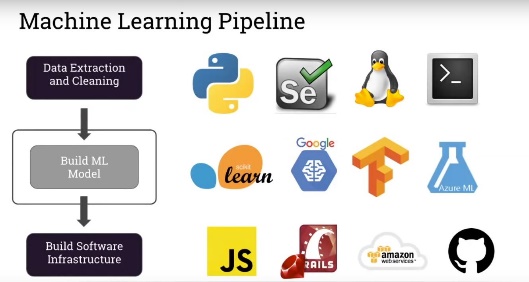
**5.1 Model**

In Fig.1 the UML diagram for our proposed system.

**Training data**: A training data set is a set of data used to discover predictive relationships. Training set is used in intelligent systems, machine learning, genetic programming and statistics.

**Testing data**: Test data is the input given to a software program. It represents data that affects or is affected by the execution of the specific module.



**5.2 Process Flow**

**5.3 Flow Diagram (Sentiment Analysis)**

****

**Input:** An entire dataset main\_data.csv made by combining multiple datasets is made to work at the backend so that when a user inputs the desired movie name a dropdown list comes with autocomplete suggestions to select from.

**API Response:** On-click the details of the movie will be loaded with the help of TMDB API and get-request. The cast and movie information will be displayed along with actor name, picture and personal information.



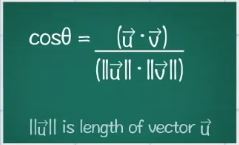
**Data Processing and Output:** The ML model will then based on similarity score of the movie requested by the user will compare it to other data columns of the dataset and return suggestions in descending order of similarity.

**Chapter 6**

**IMPLEMENTATION**

**6.1 Theoretical Implementation**

The project is undertaken using machine learning, natural language processing and makes suggestions based on similarity score of items using cosine similarity algorithm and shows reviews based on sentiment analysis. The cosine similarity algorithm provides better suggestions as it uses content based filtering instead of collaborative technique which helps suggest things similar to the user’s taste.

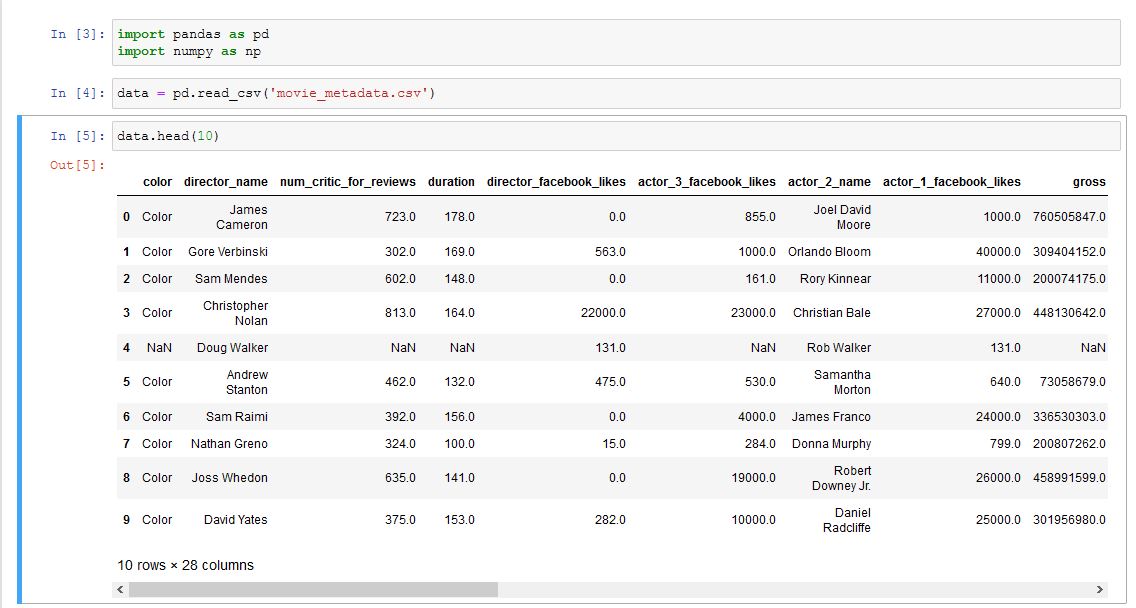


The implementation of the above system increases the user involvement in the streaming applications and helps the viewer experience.

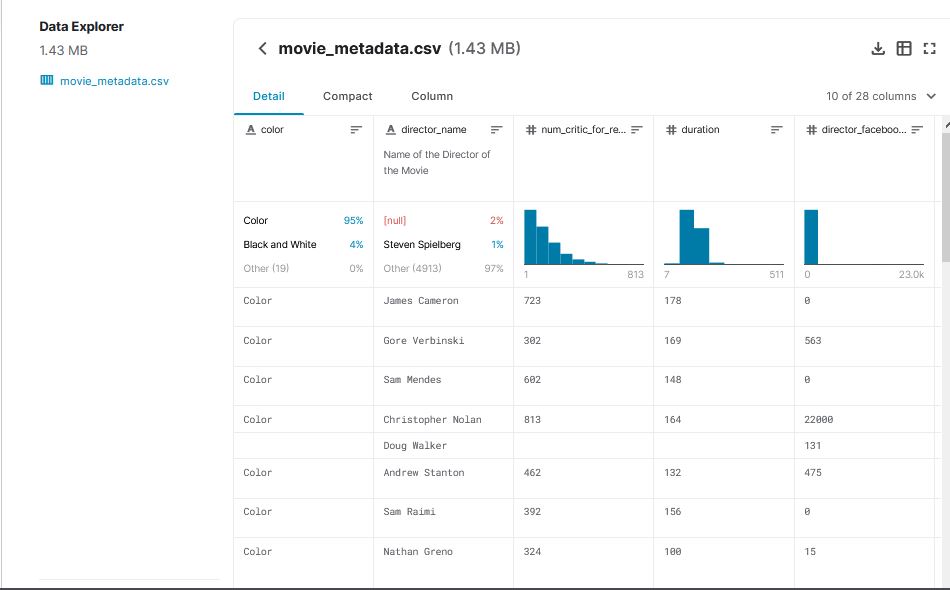
**6.2** **Snippets of Code/Wireframes/Actual System and PYNB’s**

* **Pre-Processing File 1:**

Reading the initial Dataset(movie\_metadata.csv)



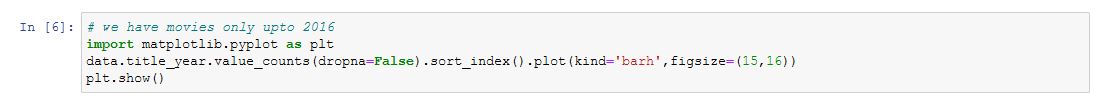
Source of the Dataset1:



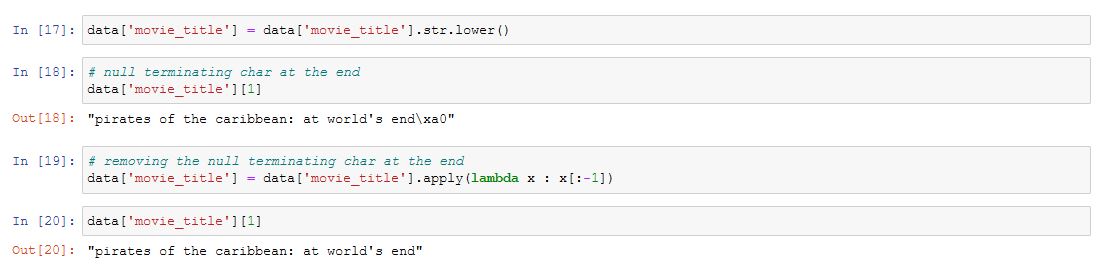
Extracting required information from DS1:



Dropping NaN values:



Removing Nulls and \xa0 chars at the end:



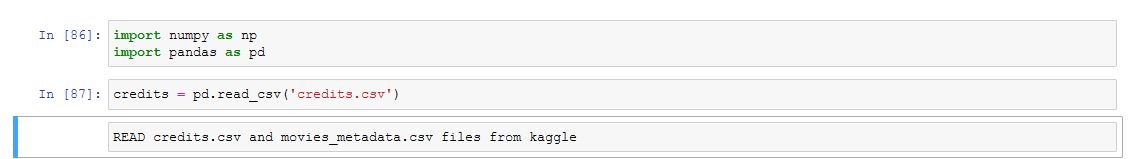
Saving the files as data.csv



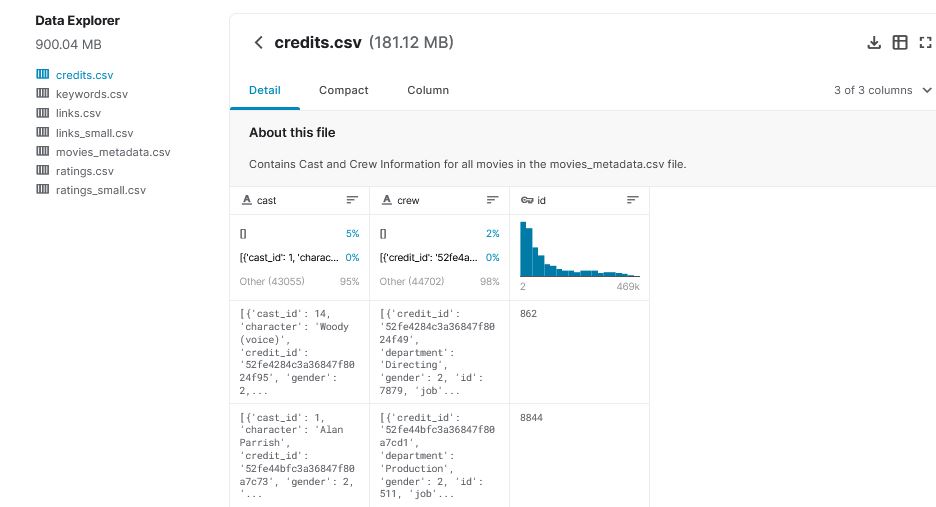
This Kaggle data set only has movies data upto 2016.

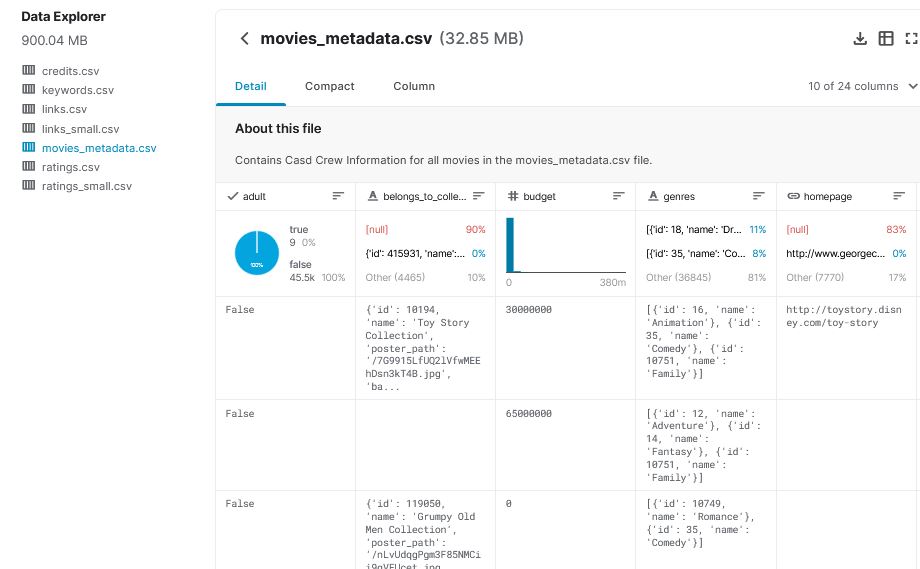
* **Pre-Processing File 2**

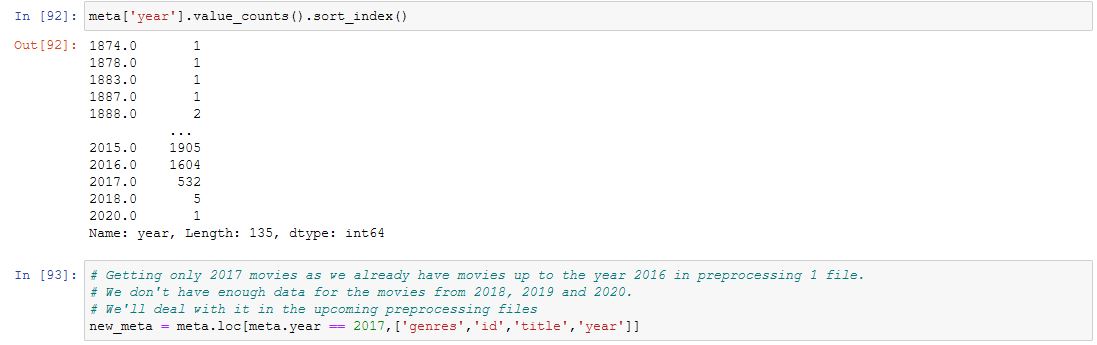
Reading credits.csv and movies\_metadata.csv dataset from kaggle for 2017-2020 movie information.



Source:



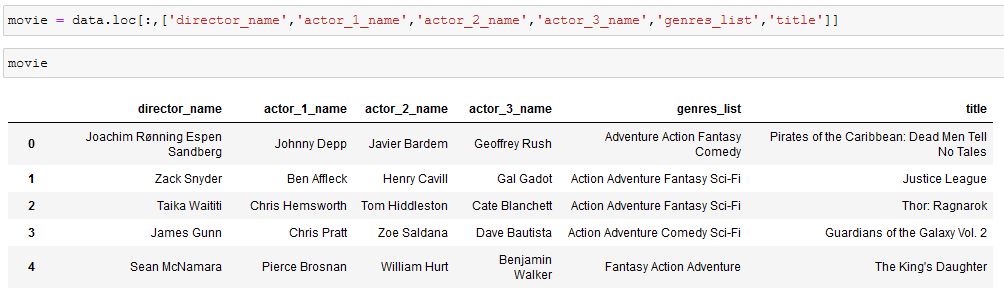




Combining New\_meta with credits to get filtered information like genres, id, title, cast, etc.



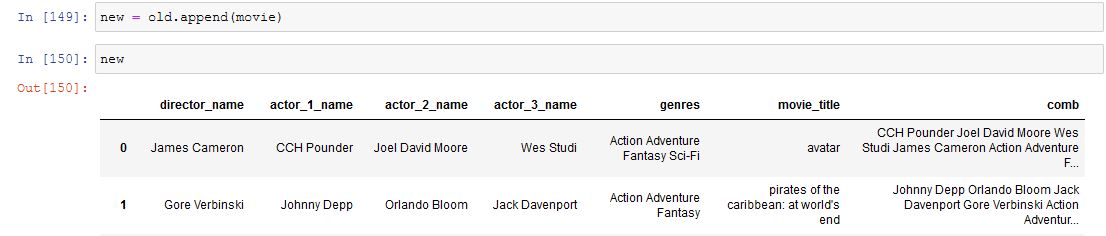
Similarly, we extract few other items and combine them and store them into a movie variable.



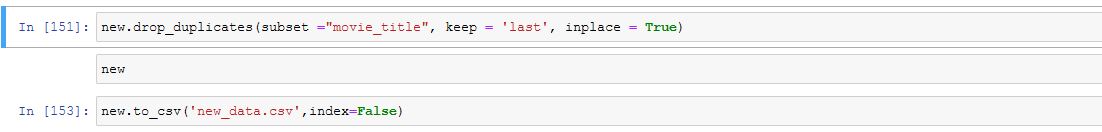
We read the older dataset



And now we combine this (2017-2018) with the dataset which contains up to 2016,



Dropping duplicates and creating a new dataset,



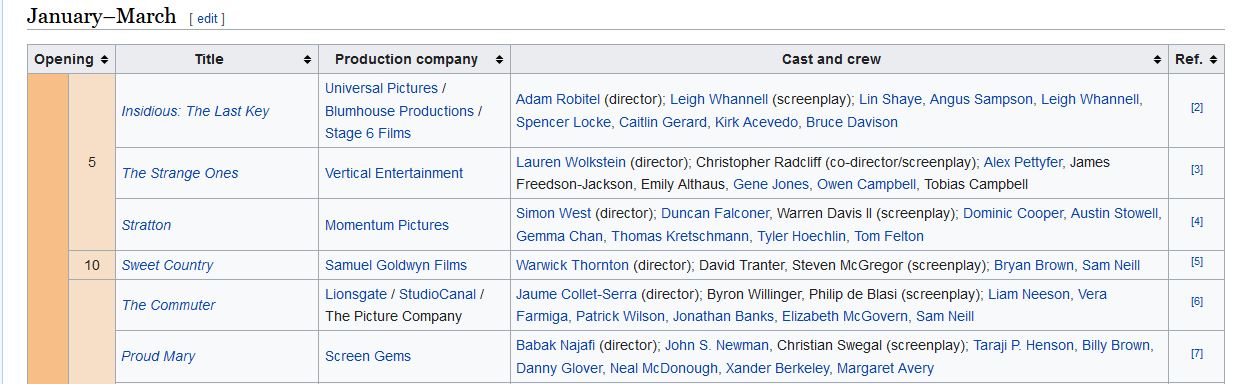
* **Pre-Processing File 3:**

2018-19 data isn’t available on kaggle hence taken from Wikipedia:

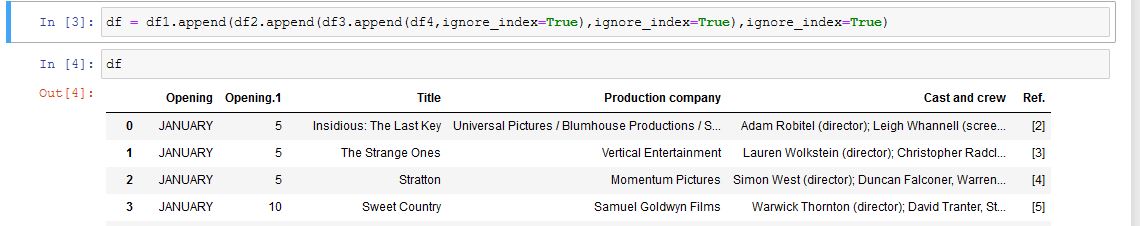




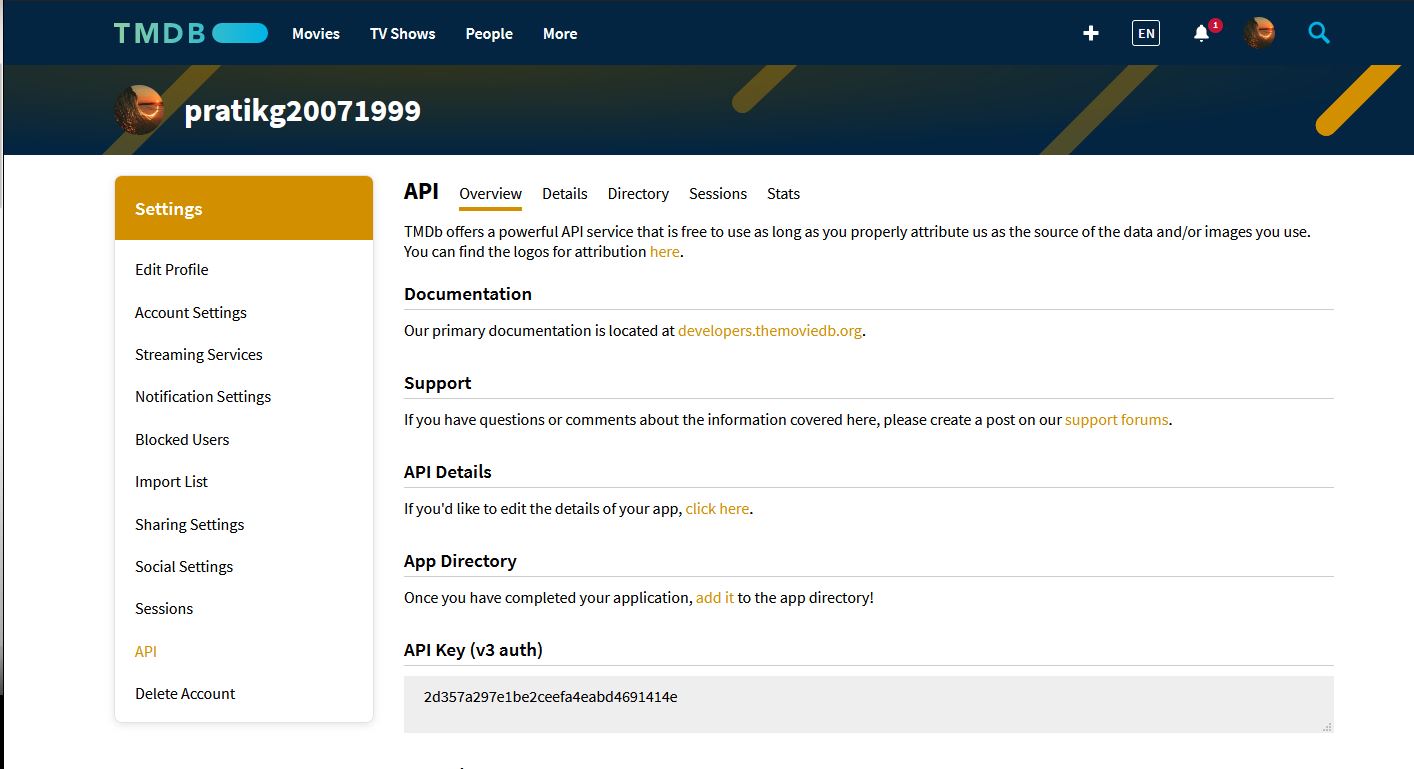
Source:



Appending data:



The Movie DB API(tmdbv3api) for appending genres:



This API helps communicate with URL to pick-up data.

Get request from the URL:



NOTE: Response generated in JSON format.

Get genres, append and pass in form of string.

The API was used as a data collection strategy used to create dataset for years 2018-2020.

Same steps followed for 2019 data:



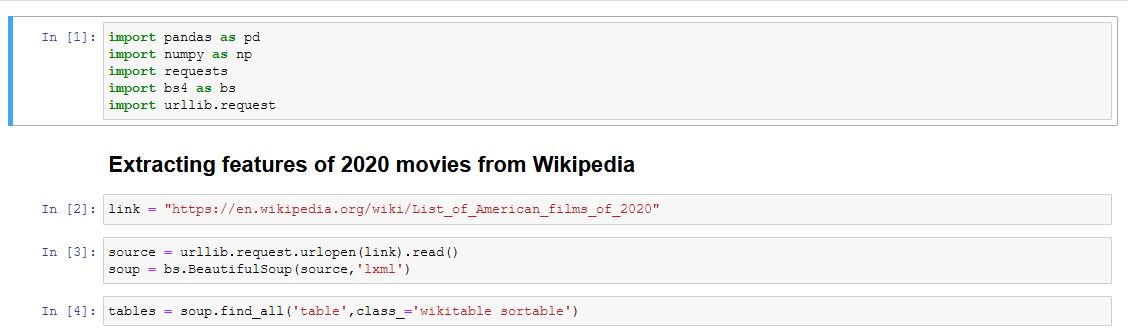
**2018**



**2019**

* **Pre-Processing 4: (WEB SCRAPING)**

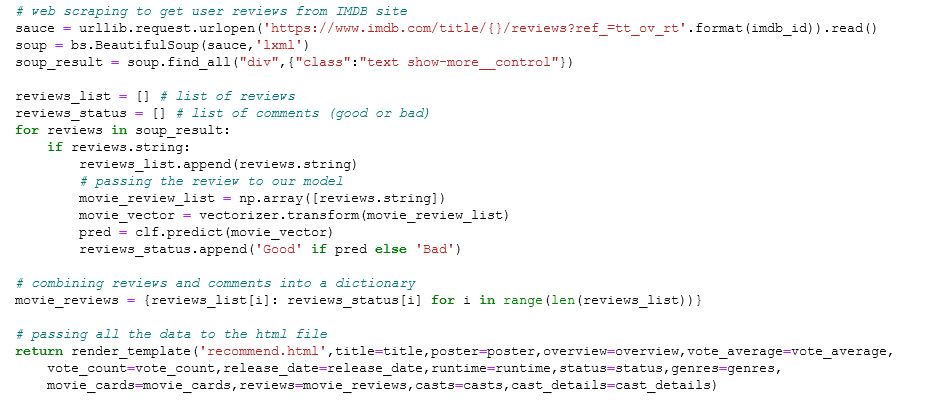
**Beautiful Soup** library is used here for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.



Similarly all the data for 2020 movies was extracted in stored in new\_df20.

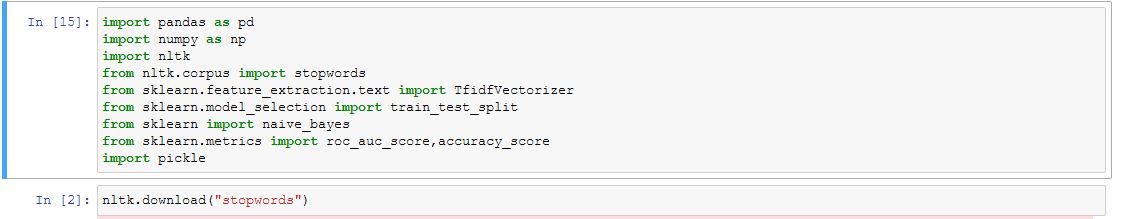
All .csv files old\_df, new\_df, new\_df20 were combined into the **FINAL DATASET: main\_data******

Getting 2018 onwards data into same features as before and finally appending into the dataset. Many steps for data collection have been repeated which could have been easily avoided by using OOPS but the project gradually moved forward hence the complex method was kept intact to maintain integrity of the project.

* **WEB-SCRAPING(for reviews)**
* **Sentiment Analysis**

Review.txt is basically a review sentiment analysis dataset, sentiment.ipynb is NLP model file for sentiment analysis.

Importing NLTK (Natural Language Toolkit) to remove ‘stopwords’.



In this file all the information from review.text is converted into vectors using **TFIDFVectorizer** to create a NLP model.



**NLP model.pkl:** is the file used for sentiment analysis.

**Algorithm used is: Multinomial Naïve Bayes**

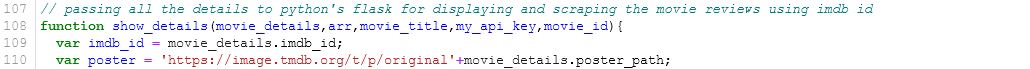
Alternatives to this are:

Hyper-parameter tuning can be used for more accuracy.

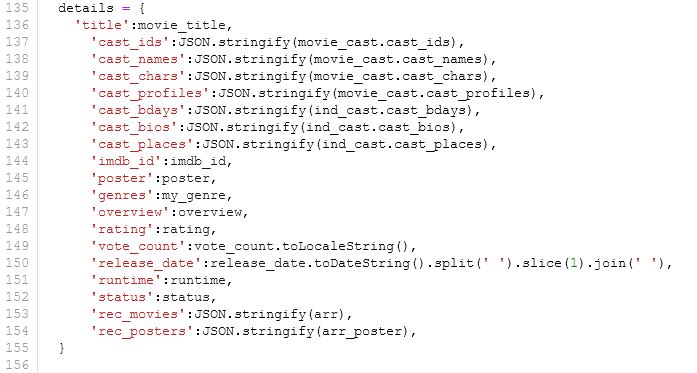
* **Recommend.js**



On-click, load details function will be called and it will do a get-request on the TMDB API.



Details getting passed to main.py:

  
Then /recommend is called by AJAX REQUEST:

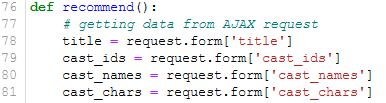


Which calls recommend function in main.py

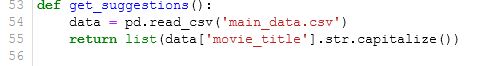
* **Main.py**

**Primary functions:**

**1:** def recommend()

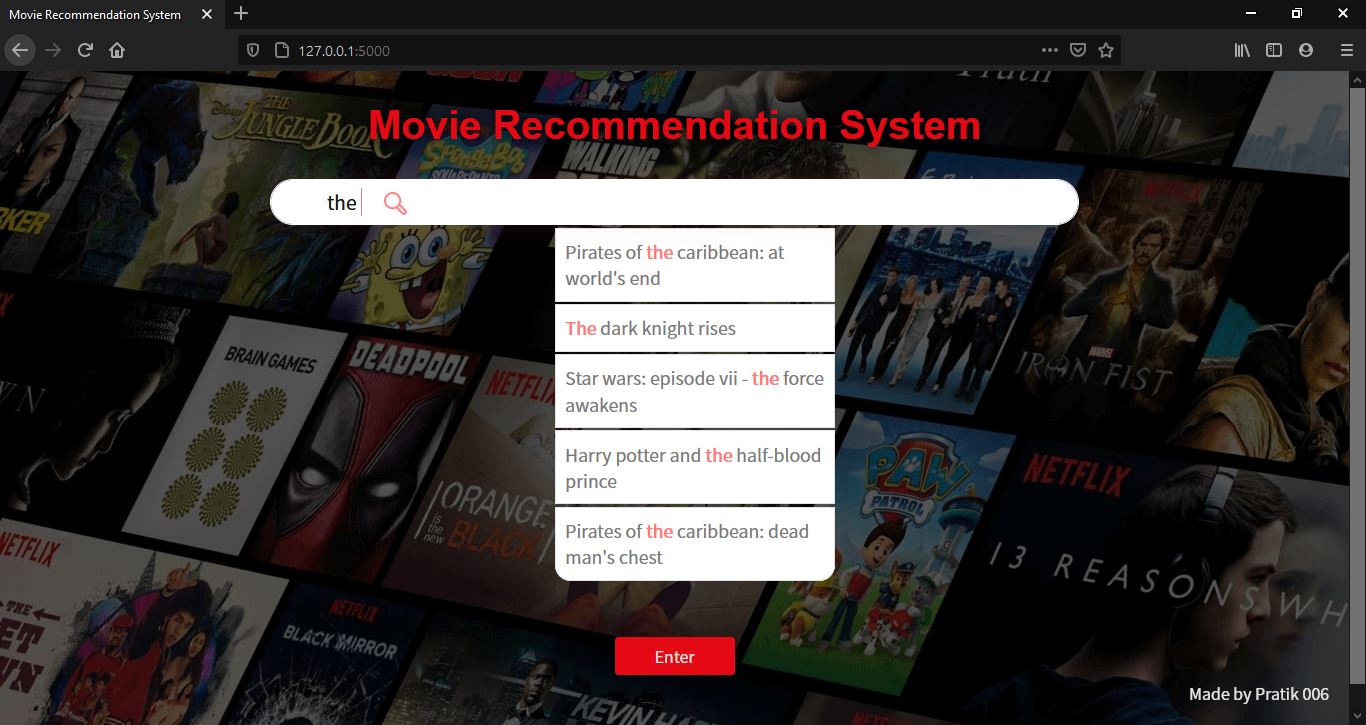


**2:** def get\_suggestions()



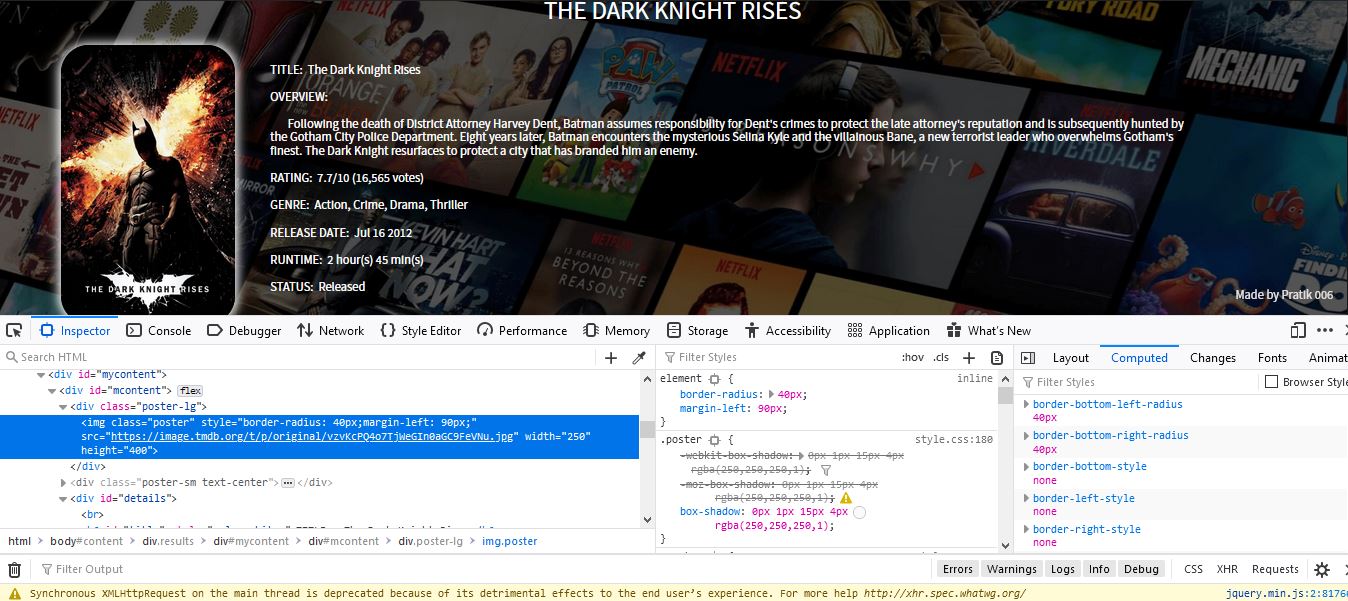
* **FINAL SYSTEM**

**Home Page:**

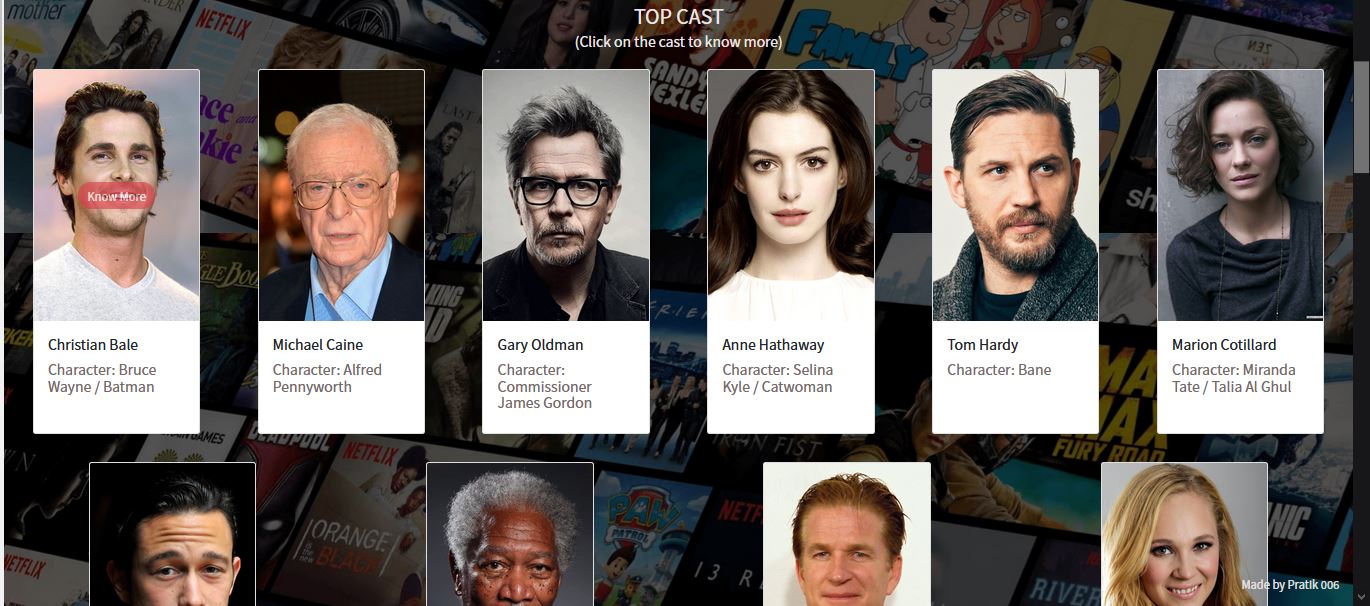


Dropdown’s are generated by Autocomplete.js

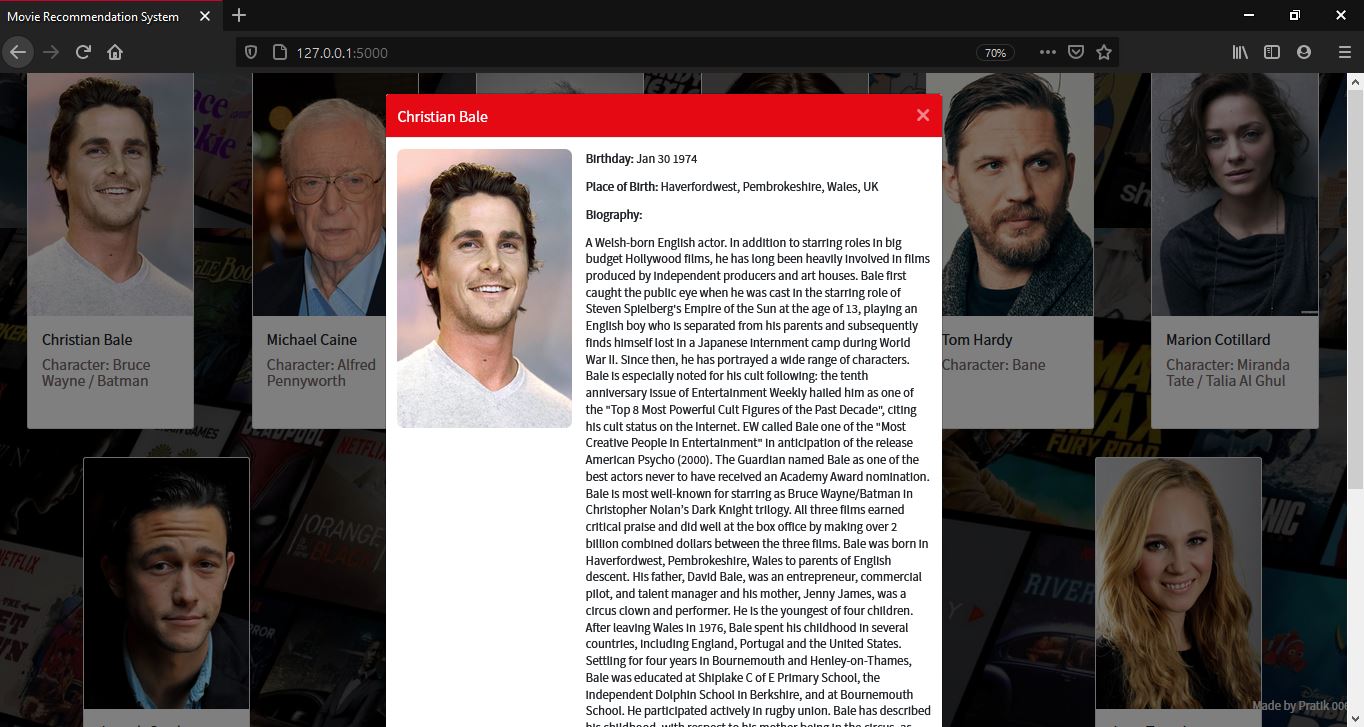
**Click to Fetch Movie Details:**



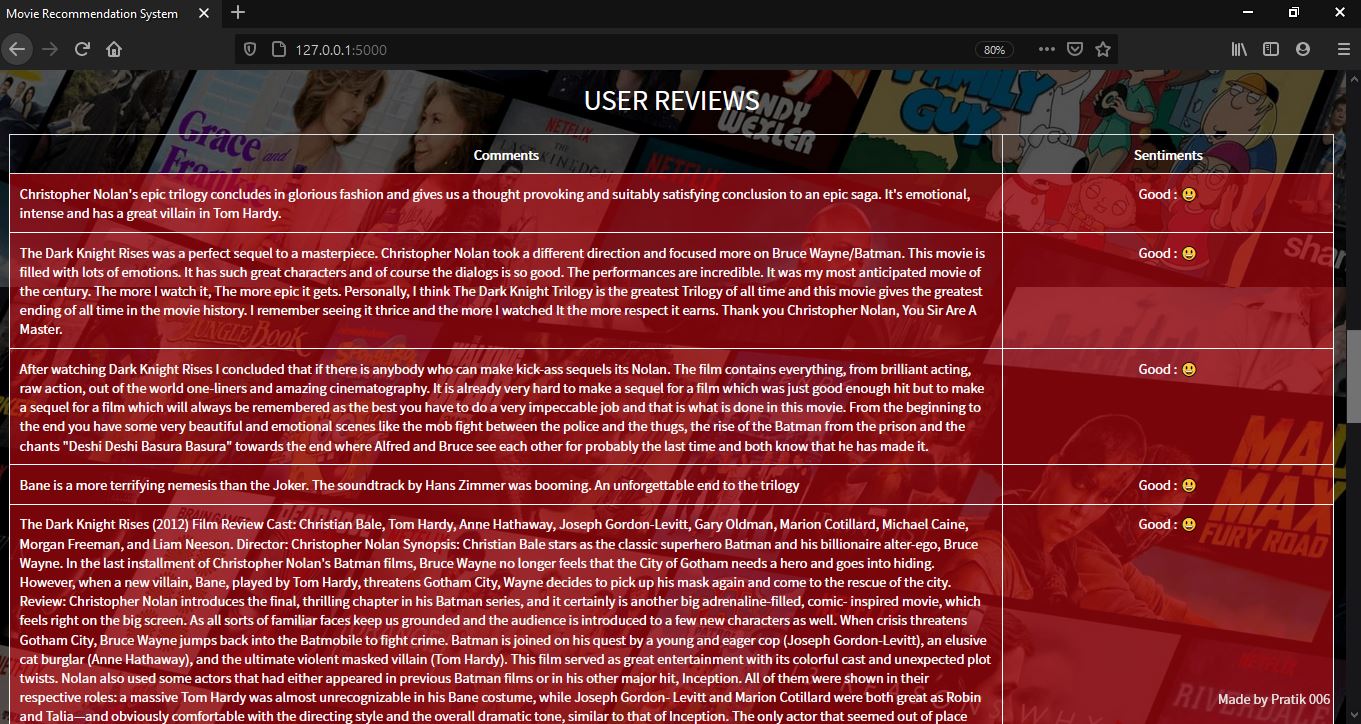
Movie poster, details and cast information fetched using TMDBV3API.



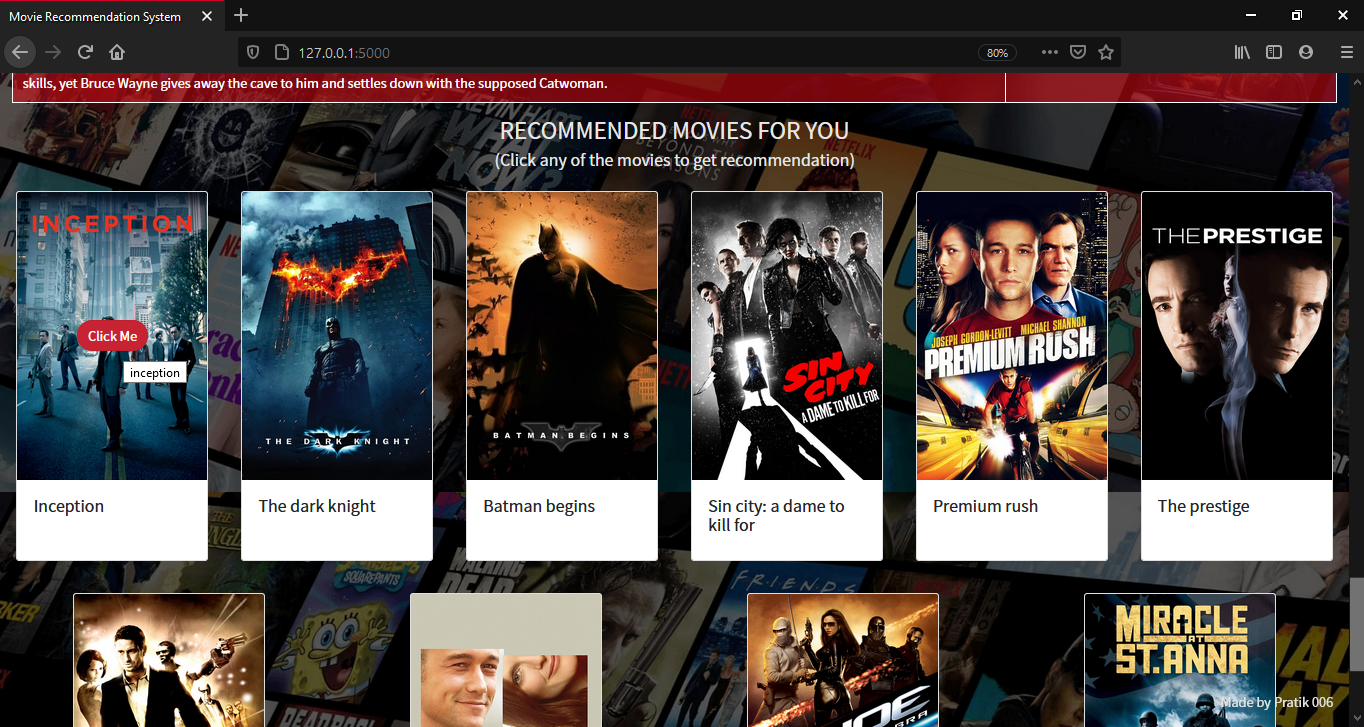
Actor photos and information fetched using TMDB API.



**Reviews based on sentiment analysis:**



**Recommendations of movies based on entry:**



**Chapter 7 (Testing)**

Metrics of testing were:

1. **Precision:**

The portion of recommendations that were successful. (Selected by the algorithm and by the user)

1. **Recall:**

The portion of relevant items selected by algorithm compared to a total number of relevant items.

**7.1 Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S No.** | **Action** | **Inputs** | **Expected Output** | **Actual Output** | **Test Result** |
| **1.** | Data extraction and assembling | Extract common fields from multiple data sets | Final Dataset with values in required fields | Dataset with Nulls, Characters and duplicate values. | Pass  (After Cleaning the obtained dataset) |
| **2.** | Embedding API | Request movies | Display details of requested movie. | Movie information and pictures fetched from the API | Pass |
| **3.** | Reviews | Training NLP Model to display reviews | Display reviews by sentiment analysis | Balanced review based on MNB algorithm | Pass |
| **4.** | Make Recommendations based on similarity score | Compare fields of input to make recommendations | Accurate Movie suggestions with respect to input. | Accurate movie suggestions based on similarity scores. | Pass |

**8. Future Scope**

The future scope can be extended using hybrid methods of filtering to make recommendations. As this project is only a recommendation engine, instead of what can be a full-fledged movie streaming platform where both collaborative and content-based approaches can be used to put together what we call as hybrid filtering. This advanced method eliminates the requirement of two different machine learning methods and despite being difficult to integrate, performs significantly well as compared to traditional recommendation methods, which is with more accuracy.

Moreover, instead of scrambling for data all over the web one good real time data repository of movies would be much better to carry out analysis.

**9. References**

* 5000 movie dataset - <https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset>
* <https://www.kaggle.com/rounakbanik/the-movies-dataset>
* <https://en.wikipedia.org/wiki/List_of_American_films_of_2020>
* API - <https://www.themoviedb.org/documentation/api>
* Flask tutorials: <https://www.youtube.com/watch?v=MwZwr5Tvyxo&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH>
* JavaScript and Ajax Request: <https://www.youtube.com/watch?v=h0ZUpPiV1ac>
* <https://www.reddit.com/r/MachineLearning/>