**Movie Recommender System Using Python**

Project report submitted in partial fulfillment of the Requirements for the

Award of the Degree of B.Tech in

Computer Science and Engineering.

**BY**

**Yatharth Shrivastava**

**(University Roll No. 2017145)**

**B.Tech (CSE)**

****

**Department of Computer Science and Engineering**

**Graphic Era** (**Deemed to** **be University)**

**Dehradun-248002**

**CONTENTS**

* **Acknowledgement**
* **Introduction**

**1.Problem statement**

**2.Motivation**

* **Tools used**

**1.Jupyter notebook**

**2.Pycharm**

* **Methodology**

**1.Dataset identification**

**2.Procuring dataset**

**3.Dataset preprocessing**

**4. Vectorization**

**5.Building recommender system**

**6.Frontend**

* **Testing / Result**
* **Conclusion**
* **Bibliography**

**ACKNOWLEDGEMENT**

Here by I am submitting the project report on **“ Movie Recommender System using Python”** as per the scheme of Graphic Era Deemed University, Dehradun.

I would like to express our sincere gratitude to **Mr.Manish Sharma,** Associate professor, for providing a congenial environment to work in and carry out our project.

I would like to also thanks kaggle for helping me in better understanding each component of topic in an interesting way.

Finally I am very much thankful to all the faculty members of the Department of Computer Science and Technology, friends and our parents for their constant encouragement, support and help throughout the period of project conduction.

Yatharth Shrivastava

**Roll no. 2017145**

**INTRODUCTION**

**Problem Statement : Movie Recommender System Using Python**

An application is required which displays movie titles based on given metadata and similarities

with input data.

**Motivation :** Recommender Systems have a simple objective of displaying most relevant information from a provided dataset to the user on the basis of similarities between the input and present elements of dataframe. They are widely used by many companies and is a backbone to few, for example amazon recommends their users similar products or what’s usually bought alongside that one product under “user also bought this” section, spotify on the other hand recommends music to the user based on their choices and what they’ve been listening to lately either through “you might as well like” or “discover weekly” or “daily mixes” sections respectively. Recommender systems may have similar objective but have different approaches, broadly categorized in 3 categories collaborative filtering recommender system, item based recommender system and content based recommender system.

In collaborative filtering products are recommended to the user on the fact that the product has been liked by user similar to the user, for example if person A and person B have similar taste in movies either of them can recommend one movie to another since they liked it and they having a similar taste the other one will like it as well.

In item based the system identifies similar items based on users previous ratings, for example if person A, B and C gave 5 star ratings to a movie X which is similar to a movie Y they’ll get recommended movie Y based on similar ratings of users A,B and C.

In content based the system uses metadata such as genre, producer, actor etc to recommend items like books movies or music. Its whole concept revolves around the idea that if you like certain item then its highly likely that you’ll like something which is similar to it, for example if you like thor you’ll also like avengers because of them being related to marvel.

Its ability to provide better recommendations to users is what makes recommender system so important, now the user doesn’t have to do a tedious job of deciding what to read/listen/watch next, the recommender system has got their back.

In this project we’ll build a content based recommender system.

**TOOLS USED**

For building our recommender system we require following applications:

1. Jupyter Notebook
2. Pycharm

**Jupyter Notebook**

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that one can also use.

**Pycharm :** PyCharm is a hybrid-platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. It has an intelligent code editor which also helps in identifying errors more efficiently, better code navigation, it helps developers create web applications in Python. It supports popular web technologies such as HTML, CSS, and JavaScript. Developers have the choice of live editing with this IDE. At the same time, they can preview the created/updated web page. It also supports popular frameworks such as Django and has wide variety of libraries at its disposal.

**METHODOLOGY**

**DATASET IDENTIFICATION:** For building our recommender system we need to select exactly what its going to recommend, since this is a content based recommender system picking up movies seemed just the right option to me because it has a diverse metadata.

**PROCURING DATASET:** Since we chose to build a movie recommender system, we require a dataset containing list of movies with attributes like genre, director, actor, plot etc.

I used tmdb (the movie database) list of movies and credits csv files from kaggle which is a great site for learning data science, they also provide datasets for making such projects.

**DATASET PREPROCESSING:** Having downloaded our datasets in form of csv files now we can start working on our project, we boot up Jupyter notebook, create a new folder and upload these datasets there, so that Jupyter notebook can access it later when required. We create a new .ipynb file next, this is where dataset preprocessing will take place, we need the following libraries through the entirety of our code

in Jupyter notebook.

**1)NUMPY**: Which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

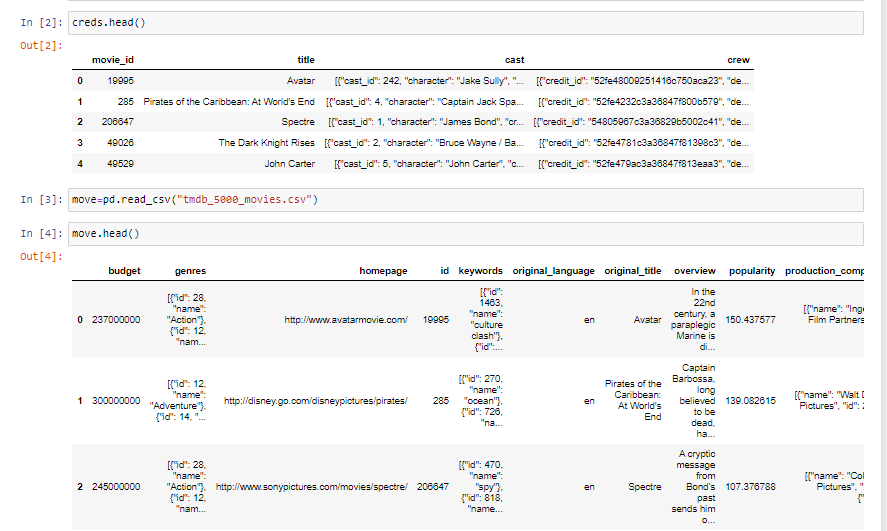
**2)PANDAS:** Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures.

**3)AST:** Responsible for majority of preprocessing ,this module helps to find out programmatically what the current grammar looks like.

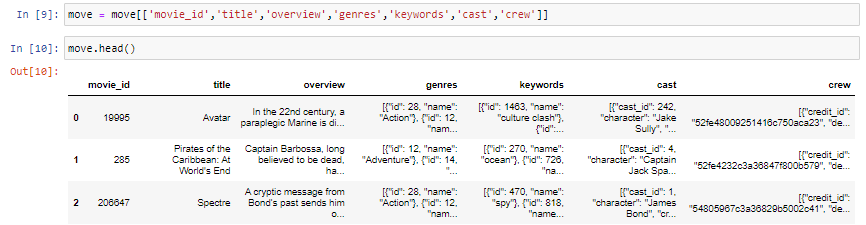
**4)SKLEARN.FEATURE\_EXTRACTION:** Used to extract features in a format supported by machine learning algorithms from datasets consisting of formats such as text and image. We specifically need CountVectorizer which converts a collection of text documents to a matrix of token counts.

**5)SKLEARN.METRICS.PAIRWISE:** Takes either a vector array or a distance matrix, and returns a distance matrix. If the input is a vector array, the distances are computed. If the input is a distances matrix, it is returned instead. We specifically need cosine\_similarity which computes similarity as the normalized dot product of rows and columns.

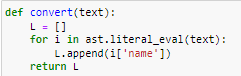
**6)PICKLE:** Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling.

After importing these libraries, we can begin our work by making our raw dataset a processed dataset which not only looks neat but also is sorted while retaining the important information and discarding rest.

Data here as shown is not neat and has a lot of unnecessary punctuation marks, we’ll work on it by first merging both datasets using merge function and then retain those attributes which we actually need.

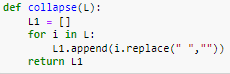


Now we remove rows with null values using dropna command,

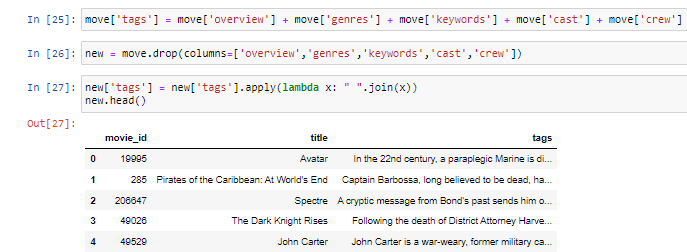


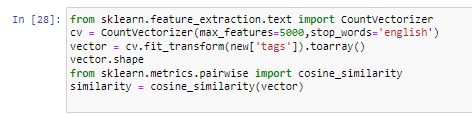


These blocks of code remove all unnecessary punctuation marks for column genres making it easier to access and read. We repeat the same procedure for remaining columns (i.e. keywords, cast, crew).



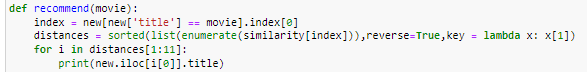
This block of code is used for removing apostrophes from columns. Finally we can include all of our sorted columns into a new dataframe and our processed dataset is now ready for use.

**VECTORIZATION: I**ts a procedure in which we convert our attribute tag into a vector, we’ll use bag of words technique here. This approach is a simple and flexible way of extracting features from documents. A bag of words is a representation of text that describes the occurrence of words within a document. We just keep track of word counts and disregard the grammatical details and the word order. It is called a bag of words because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

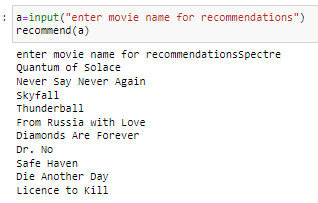


Next step is to get cosine similarity which is nothing but a rating of an item with respect to other items of same dataset, in other words is a cross product between x and x where x= length of dataset which is precisely 4803 in our case.

**BUILDING RECOMMENDER SYSTEM:** Finally we’ll make a function to see how well our dataset is processed and whether the recommendations are up to expectations or not.

We actually need to sort our list in descending order so that we get cosine similarity closest to the one which user has entered as input, use of enumerate function is to ensure we don’t lose our original indexing, this function is basically our recommender system which will be used in Pycharm later.

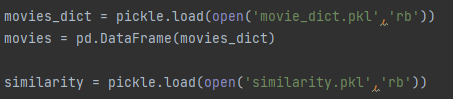
We test our recommender system by manually giving an input to check whether the results are satisfactory or not,



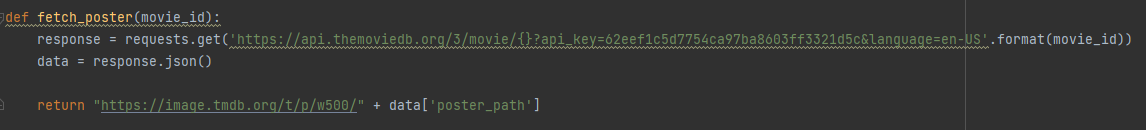
The results are quite accurate and we can say its running, before proceeding towards next step first we need to convert our dataframe into a dictionary and then use dump command for pickling both converted into dictionary dataframe and similarity matrix.

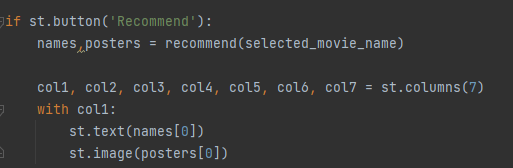
**FRONTEND:** Now the only thing left to do is to create a frontend for our recommender system which makes it more interactive with the user, we’ll be using Pycharm for this, libraries required are as follows:  
**1)PANDAS(explained above)  
2)PICKLE(explained above)  
3)STREAMLIT:** Used for making frontend here, widely known for making web apps with less code.  
**4)REQUESTS:** The de facto standard for making HTTP requests in Python.

First things first upload all pickle files on this project over Pycharm, after importing these libraries we simply require the same function recommend from Jupyter but instead it’ll store all the values in an array and will be displayed whenever we input a movie title and hit button to recommend.



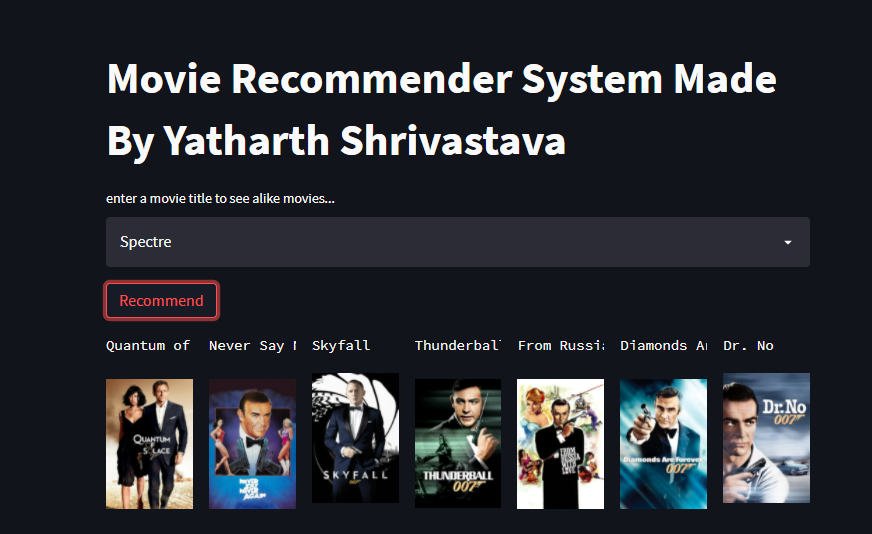
In addition to movie titles I decided to add movie posters as well, I used tmdb api keys for accessing these posters from their database.

Streamlit library had variety of functions such as button to make button, text to add text, image to add our posters, title to create one, selectbox to choose items from, it reduced code length and made everything lot more easier to work around with.



This is how we’re going to display our movie titles and respective poster, what ‘with’ basically does is ensure text or title of movie comes above its respective title and theres no mix and match. We repeat the same for remaining 6 columns and our frontend is completed.

**TESTING:** Results are same as the one from Jupyter and its working well

****

**CONCLUSION**

The completion of the project went quiet well, I learned much new things while I was building up it, and I get up to know various platforms which help us to learn. Building it opened up many paths for me as I realised that there’s a lot to learn.  
Datasets and help in code by kaggle was invaluable as well, overall working on this project was great fun as I acquired knowledge and understood working of recommender system in depth while enjoying it.

**BIBLIOGRAPHY**

* https://www.kaggle.com
* https://developers.themoviedb.org/3
* https://docs.python.org
* https://scikit-learn.org/stable/index.html
* https://www.tutorialspoint.com
* https://docs.streamlit.io/library
* Python for data analysis by Wes McKinney