**Pune Institute of Computer Technology,**

# Dhankawadi, Pune - 43

Academic year :2021-22

**MINI PROJECT REPORT ON**

MOVIE RECOMMONDATION

**Aryan Chatterjee 31105**

**Vedant Bothikar 31115**

**Shubham Chemate 31118**

# Under the guidance of

Prof. A. A. Chandorkar



DEPARTMENT OF COMPUTER ENGINEERING

Academic Year 2021-22

# Table of Contents -

1. Title
2. Problem Statement
3. Date
4. Objectives
5. Theory
6. System Architecture
7. Methodology
8. Results
9. Conclusion

1. **Title -** movie recommendation model

## 2. Problem Statement -

Develop a movie recommendation model using the scikit-learn library in python. Refer dataset

<https://github.com/rashida048/Some-NLProjects/blob/master/movie_dataset.csv>

3. **Date –** 29/04/22

### 4. Objectives -

* To describe the dataset and draw analysis.
* To recommend list of movies based on users input.
* Display the similarity matrix score.

### 5. Theory -

* Libraries Used - - Pandas
  + Numpy
  + Seaborn
  + Matplotlib
  + Scikitlearn

* Functions Used -
  + **Pandas.read\_csv()** - A civ file path is passed as the parameter to this function and a comma-separated values (csv) file is returned as two-dimensional data structure with labeled axes in form of Dataframe or TextParser.

* + **dataframe.info()** - This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.

* + **df.set\_index()** – this function is used to set particular column as index of dataframe

* + **df.drop(columns = [##col name],axis = 1,inplace = True)-** this function is used to drop columns mentioned in columns parameter axis = 1means columnwise and inplace = True means in the same dataframe.

* + **dataframe.isnull()** - Return a boolean same-sized object indicating if the values are NA. NA values, such as None or numpy.NaN, gets mapped to True values. Everything else gets mapped to False values.

* + **dataframe.columns** - Returns a list of all column names present in the dataframe.

* + **Convert-string-dict-to-dict** : this function is used to convert string dictionaries in certain columns and extract only useful data from it.

* + **Select\_top\_4()** – As all actors contribution not matters that much so this function selects only top 4 actors from df.cast column

* + **CountVectorizer()** – this is sklearn library function used to convert document of words to vector format so that our model can do further calculations like cosine similarity.

* + **Recommendation\_model** – This is final class were we will take final dataframe named as final and transform it using class methods and return the new dataframe named as new\_df.

* + **Main\_Recommendation \_model(movie\_name)** – This function is our main function which takes movie name as input find its id as key calculate cosine similarity with all other movies sort it and present top 10 movies with similarity scores.

### 6. System Architecture -

Jupiter notebook, Python, Windows/Linux operating systems, i5 intel processor , 8 GB RAM

### 7. Methodology -

**Step 1 - Importing all the required libraries**

*import pandas as pd import numpy as np import seaborn as sns*

*import matplotlib.pyplot as plt*

**Step 2 - Reading the dataset**

Dataset link –

[*https://github.com/rashida048/Some-NLProjects/blob/master/movie\_dataset.csv*](https://github.com/rashida048/Some-NLProjects/blob/master/movie_dataset.csv) *df = pd.read\_csv(“movies\_dataset.csv”)*

**Step 3 – Exploring the data**

*#shape of the dataset df.shape*

*#information of all the columns in the dataset df.info()*

*#total null values in the dataset df.isnull.sum()*

**Step 4 - Dataset dividing in int\_df and obj\_df**

*-int\_cols = [col for col in df.columns if f[col].dtype != 'object'] -int\_df = df[int\_cols]*

* *obj\_cols = [col for col in df.columns if df[col].dtype == 'object']*
* *obj\_df = df[ obj\_cols ]*

**Step 5 - clean the data and convert it into suitable format** *def convert\_string\_dict\_to\_dict(sr): lis = [] word = "" i = 0 while i < (len(sr)-3):*

*if sr[i:i+4] == "name": i = i + 8 while sr[i]!='"': word += sr[i] i+=1 lis.append(word) word = "" else: i += 1 return lis*

*obj\_df.production\_countries =*

*obj\_df.production\_countries.apply(convert\_string\_dict\_to\_dict)*

**step 6- clean the data by removing stopwords and lemmatize** *import nltk from nltk.corpus import stopwords from nltk.stem.wordnet import WordNetLemmatizer*

*def remove\_stopwords(sent): sent = sent.split() stop\_words = set(stopwords.words('english')) sent = [w for w in sent if not w in stop\_words] return sent*

*def lemmatize(sent):*

*lemmatizer = WordNetLemmatizer() sent = [lemmatizer.lemmatize(w) for w in sent] return sent*

*dfn.keywords = dfn.keywords.apply(lambda x : remove\_stopwords(x)) dfn.keywords = dfn.keywords.apply(lambda x : lemmatize(x))*

**step 7- import countvectorizer model and convert keywords to vectorized form** """

*## countvectorizer with stopwords and lemmatize and remove stopwords and n features 200 from sklearn.feature\_extraction.text import CountVectorizer tfidf\_vectorizer = CountVectorizer(stop\_words="english",max\_features=100) tfidf\_matrix = tfidf\_vectorizer.fit\_transform(dfn.keywords)"""*

**step 8- main function to recommend top 10 movies**

*def main\_recommendation( name ):*

*id = dfn.loc[dfn.title == name].index.values[0] model = recommondation\_model( final.copy() ) new\_df = model.recommond\_operations(id)#.iloc[:,3:] cos = cosine\_similarity(new\_df,new\_df) indices = {} cnt = 0 for i in new\_df.index:*

*indices[i] = cnt cnt += 1*

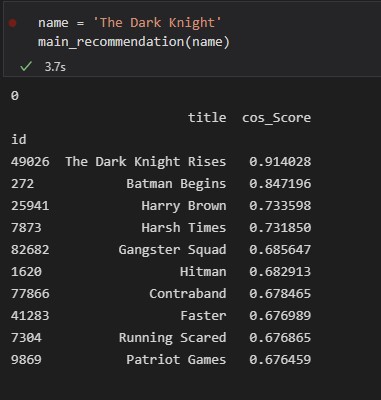
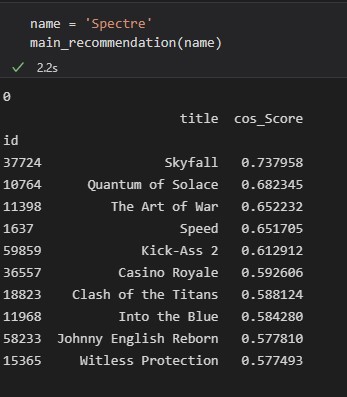
*new\_df['cos\_Score'] = cos[indices[id],:] new\_df['title'] = dfn.title*

*## sort the according to one column*

*new\_df.sort\_values(by = 'cos\_Score',ascending = False,inplace = True) print(new\_df[['title','cos\_Score']].iloc[1:11,:])*

*#return new\_df*

### 8. Results -



### 9. Conclusion -

Successfully completed movie recommendation project using Scikitlearn Library.