**Pune Institute of Computer Technology,**
**Dhankawadi, Pune - 43**

Academic year : 2021-22

**MINI PROJECT REPORT ON**

# MOVIE RECOMMENDATION

**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** - 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
*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 -**

```
name = 'Spectre'
main_recommendation(name)
```
✓ 2.2s

```
0
                      title  cos_Score
id
37724                Skyfall   0.737958
10764      Quantum of Solace   0.682345
11398         The Art of War   0.652232
1637                   Speed   0.651705
59859              Kick-Ass 2   0.612912
36557           Casino Royale   0.592606
18823    Clash of the Titans   0.588124
11968           Into the Blue   0.584280
58233  Johnny English Reborn   0.577810
15365     Witless Protection   0.577493
```

```
name = 'The Dark Knight'
main_recommendation(name)
```
✓ 3.7s

```
0
                       title  cos_Score
id
49026  The Dark Knight Rises   0.914028
272            Batman Begins   0.847196
25941            Harry Brown   0.733598
7873             Harsh Times   0.731850
82682         Gangster Squad   0.685647
1620                  Hitman   0.682913
77866             Contraband   0.678465
41283                 Faster   0.676989
7304          Running Scared   0.676865
9869           Patriot Games   0.676459
```

## 9. **Conclusion -**

Successfully completed movie recommendation project using Scikitlearn Library.