NETFLIX MOVIE RECOMMENDATION SYSTEM

Table of content

Abstract

My project will be the Netflix movies and TV Show dataset, Netflix is an application that keeps growing exponentially whole around the world and it is the most famous streaming platform. It given a large number of movies and series available on the platform, it is a perfect opportunity to flex our data manipulation skills and dive into the entertainment industry.

The goal of this project was to Recommends content based on movie description. Here I would recommend Movie based on Movie titles only. Similar movies would have similar names thus having a high cosine similarity. Then the movies that are most likely to be similar are recommended.

I generates TF-IDF matrix and finds cosine similarity of each movie with other movies and displays top 10 similar movies, and to analysis and visualization of content is available in different countries, analysis of Actors / Directors and find interesting insights.

Dataset

This dataset contains data collected from Netflix of different TV shows and movies from the year 2008 to 2021.

- type: Gives information about 2 different unique values one is TV Show and another is Movie
- title: Gives information about the title of Movie or TV Show
- director: Gives information about the director who directed the Movie or TV Show
- cast: Gives information about the cast who plays role in Movie or TV Show
- release_year: Gives information about the year when Movie or TV Show was released
- rating: Gives information about the Movie or TV Show are in which category (eg like the movies are only for students, or adults, etc)
- duration: Gives information about the duration of Movie or TV Show
- listed_in: Gives information about the genre of Movie or TV Show
- description: Gives information about the description of Movie or TV Show

EDA

Importing library

```
import pandas as pd
import numpy as np
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import pandas as pd
import seaborn as sns
```

See the Netflix data:

```
In [4]: # imports a CSV file (netflix_titles) to DataFrame format
        df =  pd.read_csv("netflix_titles.csv")
        df.head(10)
```

Out[4]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of

Data Exploration

Exploring the data

Data Exploration

```
In [5]: #print a summary of a DataFrame
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
```

Finding if the dataset contains null values

```
In [6]: #find the nulls value
        df.isnull().sum()
```

```
Out[6]: show_id         0
        type            0
        title           0
        director     2634
        cast          825
        country       831
        date_added     10
```

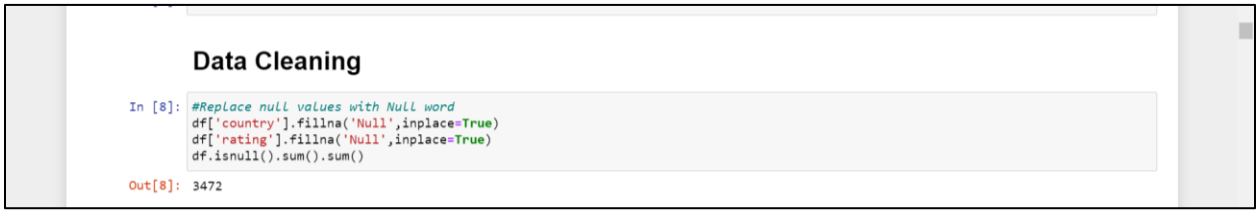
Finding how many unique values are there in the dataset

```
In [7]: #find unique value
        df.nunique()
```

```
Out[7]: show_id         8807
        type               2
        title           8807
        director        4528
        cast            7692
        country          748
        date_added      1767
        release_year      74
        rating            17
```

Data Cleaning

Replacing null values with ‘NULLs’



Data Cleaning

```
In [8]: #Replace null values with Null word
        df['country'].fillna('Null',inplace=True)
        df['rating'].fillna('Null',inplace=True)
        df.isnull().sum().sum()
```

```
Out[8]: 3472
```

Remove missing value

```
df['rating'].fillna('Null',inplace=True)
df.isnull().sum().sum()
```

Out[8]: 3472

```
In [56]: df = df.dropna( how='any',subset=['cast', 'director'])

In [57]: df.isnull().sum()
```

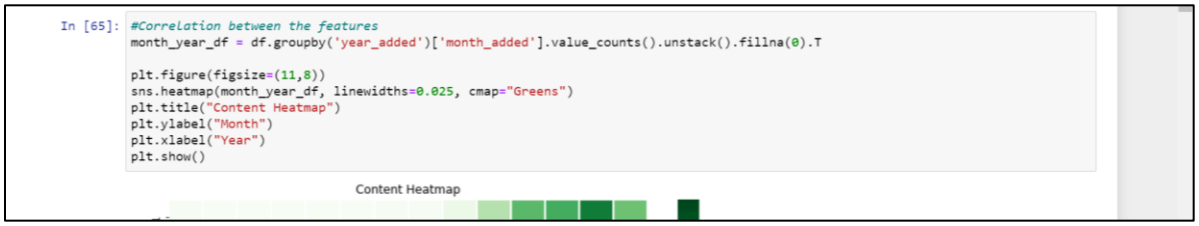
Converting into a proper date-time format

```
dtype: int64

In [10]: #Converting into date-time format and adding two more features year and month.

df["date_added"] = pd.to_datetime(df['date_added'])
df['year_added'] = df['date_added'].dt.year
df['month_added'] = df['date_added'].dt.month

In [65]: #Correlation between the features
```

Data Visualization

Chart 1

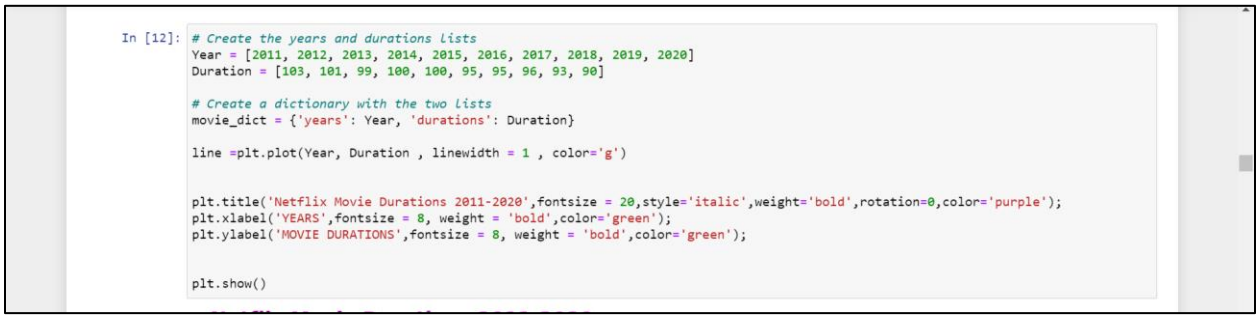
Viewing the correlation between the features.



```
In [65]: #Correlation between the features
month_year_df = df.groupby('year_added')['month_added'].value_counts().unstack().fillna(0).T

plt.figure(figsize=(11,8))
sns.heatmap(month_year_df, linewidths=0.025, cmap="Greens")
plt.title("Content Heatmap")
plt.ylabel("Month")
plt.xlabel("Year")
plt.show()
```


Chart 2

Movie duration over years



```
In [12]: # Create the years and durations lists
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Duration = [103, 101, 99, 100, 100, 95, 95, 96, 93, 90]

# Create a dictionary with the two lists
movie_dict = {'years': Year, 'durations': Duration}

line =plt.plot(Year, Duration , linewidth = 1 , color='g')


plt.title('Netflix Movie Durations 2011-2020',fontsize = 20,style='italic',weight='bold',rotation=0,color='purple');
plt.xlabel('YEARS',fontsize = 8, weight = 'bold',color='green');
plt.ylabel('MOVIE DURATIONS',fontsize = 8, weight = 'bold',color='green');


plt.show()
```

Chart3

Pie chart to show the percentage of content type

```
In [70]: #plot Pie chart to show the percentage of content type
         df['type'].value_counts().plot(kind='pie', autopct='%1.0f%%', legend=dict(x=0.1, y=1.1))
         plt.title("Persentage of Content Type", y=1.02 , fontsize = 10,style='italic',weight='bold',rotation=0)

Out[70]: Text(0.5, 1.02, 'Persentage of Content Type')
```

Chart 4

Bar charts of the number of Movies per Country.(descending)

```
In [46]: #creates bar charts of the number of Movies per Country.(descending)
         data = df.set_index('title').country.str.split(', ', expand=True).stack().reset_index(level=1, drop=True);

         plt.figure(figsize=(7,9))
         N = sns.countplot(y = data, order=data.value_counts().index[:20] , palette= ["#7fcdbb","#edf8b1"])
         plt.title('Top 20 Countries on Netflix' , family='serif',fontsize = 15,loc='center',color='r');
         plt.xlabel('Titles')
         plt.ylabel('Country')
         plt.show()
```

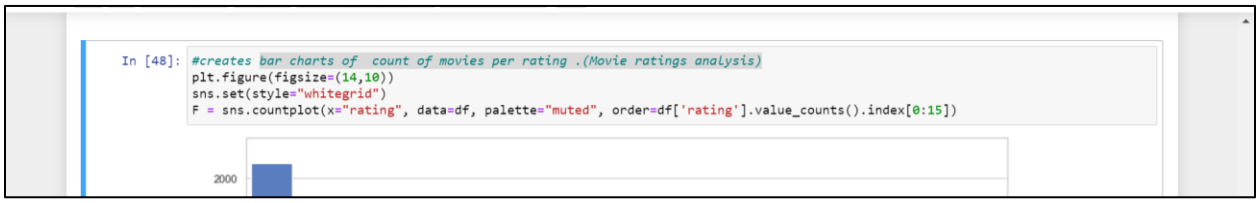
Chart 5

The content added for Movies compared to TV Shows over years

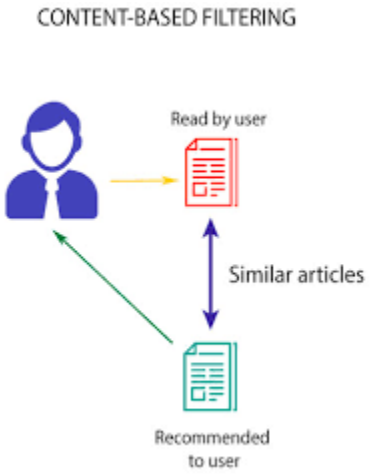
```
In [14]: #the content added for  Movies compared to TV Shows over years
         d1 = df[df["type"] == "TV Show"]
         d2 = df[df["type"] == "Movie"]

         col = "year_added"

         X1 = d1[col].value_counts().reset_index()
         X1 = X1.rename(columns = {col : "count", "index" : col})
         X1['percent'] = X1['count'].apply(lambda x : 100*x/sum(X1['count']))
         X1 = X1.sort_values(col)


         Y2 = d2[col].value_counts().reset_index()
         Y2 = Y2.rename(columns = {col : "count", "index" : col})
         Y2['percent'] = Y2['count'].apply(lambda x : 100*x/sum(Y2['count']))
         Y2 = Y2.sort_values(col)


         new_x = go.Scatter(x=X1[col], y=X1["count"], name="TV Shows", marker=dict(color="#EC7063"))
         new_y = go.Scatter(x=Y2[col], y=Y2["count"], name="Movies", marker=dict(color="#1D8348"))
         data = [new_x, new_y]
         layout = go.Layout(title="Content added over the years",legend=dict(x=0.1, y=1.1, orientation="h"))
         fig = go.Figure(data, layout=layout)
         fig.show()
```

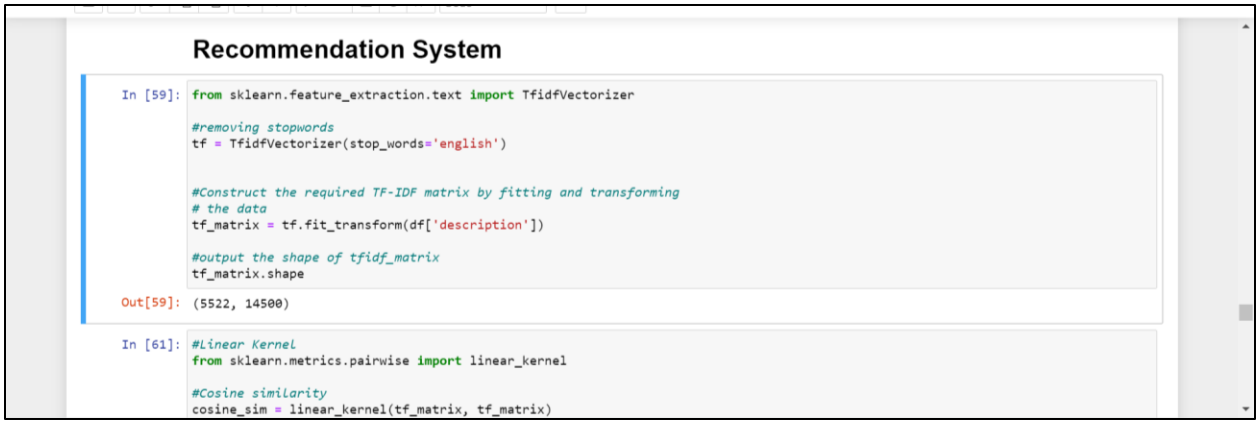
Chart 6

Bar charts of count of movies per rating .(Movie ratings analysis)

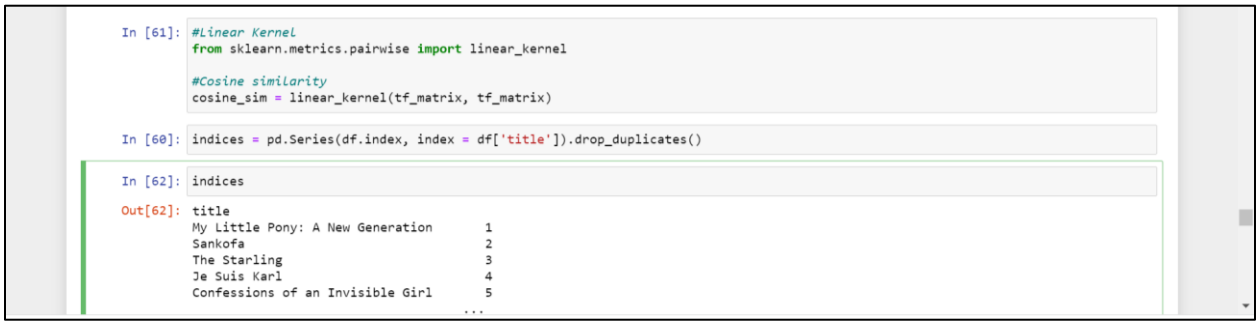


Recommendation System (TF-IDF)





There are about 14500 words described for the 5522 movies in this dataset.



- Initialize Tfidf vectorizer & fit into title column
- Calculate cosine similarity
- Convert all titles into a Series associated with movie index numbers
- Function that gets movie recommendations based on the cosine similarity score of movie titles

Tools

- Numpy and Pandas for data manipulation
- Matplotlib and Seaborn for plotting