1.NETFLIX MOVIE RECOMMENDATION SYSTEM

INDEX

ABSTRACT

In the spread of information, to quickly find one's favorite movie in a large number of movies has become a very important issue. Netflix is a streaming service that allows the members to watch a wide variety of award-winning TV shows, movies, documentaries, and more than thousands of internet-connected devices. Movie recommendation system can play an important role especially when the user has no clear target movie. In this paper, we design and implement a Netflix movie recommendation system using K-Means Clustering Algorithm and Natural Language Processing.

1.1 Introduction to Machine learning:

Machine learning (ML) is the study of computer algorithms that improve automatically through experience.[1] It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than have human programmers

specify every needed stepThe discipline of machine learning employs various approaches to help computers learn to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers

Types of learning algorithms

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised learning

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs.The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix.

Types of Supervised learning algorithms include Active learning , classification and regression.Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range.

Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using

a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. Two of the main methods used in unsupervised learning are principal component and cluster analysis. Cluster analysis is used in unsupervised learning to group, or segment, datasets with shared attributes in order to extrapolate algorithmic relationships. Cluster analysis is the assignment of a set of observations into subsets (called *clusters*) so that observations within the same cluster are similar according to one or more predesignated criteria, while observations drawn from different clusters are dissimilar.

There are many more learning algorithms such as Semi-supervised learning,Reinforcement learning ,Self learning,Feature learning, Anomaly detection *and many more*..

Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.

Artificial neural networks

Artificial neural networks (ANNs), or connectionist systems, are computing systems vaguely inspired by the biological neural networks that constitute

animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is a model based on a collection of connected units or nodes called "artificial neurons", which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit information, a "signal", from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it. In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called "edges". Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

Decision trees

Decision tree learning uses a decision tree as a predictive model to go from observations about an item (represented in the branches) to conclusions

about the item's target value (represented in the leaves). It is one of the predictive modeling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data, but the resulting classification tree can be an input for decision making.

Support vector machines

Support vector machines (SVMs), also known as support vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.[58] An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Regression analysis

Regression analysis encompasses a large variety of statistical methods to estimate the relationship between input variables and their associated features. Its most common form is linear regression, where a single line is drawn to best fit the given data according to a mathematical criterion such as ordinary least squares. The latter is often extended by regularization (mathematics) methods to mitigate overfitting and bias, as in ridge regression. When dealing with non-linear problems, go-to models include polynomial regression (for example, used for trendline fitting in Microsoft Excel [59]), Logistic regression (often used in statistical classification) or even kernel regression, which introduces non-linearity by taking advantage of the kernel trick to implicitly map input variables to higher dimensional space.

Bayesian networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables andtheir conditional independence with a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Why Python?

Artificial intelligence is considered to be the trending technology of the future. Already there are a number of applications made on it. Due to this, many companies and researchers are taking interest in it. But the main question that arises here is that in which programming language can these AI applications be developed? There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI. Among them, Python programming language gains a huge popularity and the reasons are as follows −

Simple syntax & less coding

Python involves very less coding and simple syntax among other programming languages which can be used for developing AI applications. Due to this feature, the testing can be easier and we can focus more on programming.

Inbuilt libraries for AI projects

A major advantage for using Python for AI is that it comes with inbuilt libraries. Python has libraries for almost all kinds of AI projects. For example, NumPy, SciPy, matplotlib, nltk, SimpleAI are some the important inbuilt libraries of Python.

- Open source − Python is an open source programming language. This makes it widely popular in the community.
- Can be used for broad range of programming − Python can be used for a broad range of programming tasks like small shell script to enterprise web applications. This is another reason Python is suitable for AI projects.

Features of Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python's features include the following −

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode − Python has support for an interactive modewhich allows interactive testing and debugging of snippets of code.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable − We can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases − Python provides interfaces to all major commercial databases.
- GUI Programming − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- Scalable − Python provides a better structure and support for large programs than shell scripting.

Important features of Python

Let us now consider the following important features of Python −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

1.2 Objectives of Research:

Our project is about Netflix movie or tv show recommendation through this we can recommend what kind of movie or tv show we can see.In this we have taken the data consists of Show ID,Type,Director,Releaseyear,Dateadded,genre,description,review. This recommendation is very useful for everyone to see what type of movies in it.

1.3 Problem Statement:

- What to watch on Netflix?
- Find Similar movies/ TV shows using text similarity techniques
- How to find the best rated Movies in Netflix Best Movies
- Show me the ratings
- Which Shows are the best

2.Review of Literature

Basically, the idea is to recommend the most popular movies to the users. They could be the more watched ones, or also the ones with the highest ratings. The popularity recommendations can be created based on usage data and item content

Model-based techniques

The ratings are used to implement a model that will improve the results of the collaborative filtering in order to find patterns in the data. To build a model some data mining or machine learning algorithms can be applied. These kinds of models are pretty useful to recommend a set of movies, in the fastest way and show similar results to the Memory-based models. Model-based techniques are based on Matrix factorization (MF), which is very popular because it is an unsupervised learning method for dimensionality reduction

Some of the techniques that might be applied are based on Dimensionality Reduction techniques, for instance, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Probabilistic Matrix Factorization (PMF), Matrix completion Technique, Latent Semantic methods, and Regression and Clustering

Content-based filtering:

The Content-based filtering (CB) aims to recommend items or movies that are alike to movies the user has liked before. The main difference between this approach and the CF is that CB offers the recommendation based not only in similarity by rating, but it is more about the information from the products i.e., the movie title, the year, the actors, the genre.

In order to implement this methodology, it is necessary to possess information describing each item, and some sort of user profile describing what the user likes is also desirable. The task is to learn the user preferences, and then locate or recommend items that are "similar" to the user preferences

There are several techniques that can be implemented to develop a recommendation model, based on the recommendations that can be suggested. For instance, applications of information retrieval such as Term Frequency (TF) or Inverse Document Frequency (IDF) (Salton, 1989), and some machine learning techniques, including Naive Bayes, support vector machine, decision trees, among others.

Term-Frequency - Inverse Document Frequency (TF - IDF):

Tfidf Vectorizer – It converts a collection of raw documents to a matrix of TF-IDF features. Equivalent to Count Vectorizer followed by Tfidf Transformer . If 'filename', the sequence passed is like an argument to fit, it is expected to be a list of filenames that need reading to fetch the raw content to analyze.

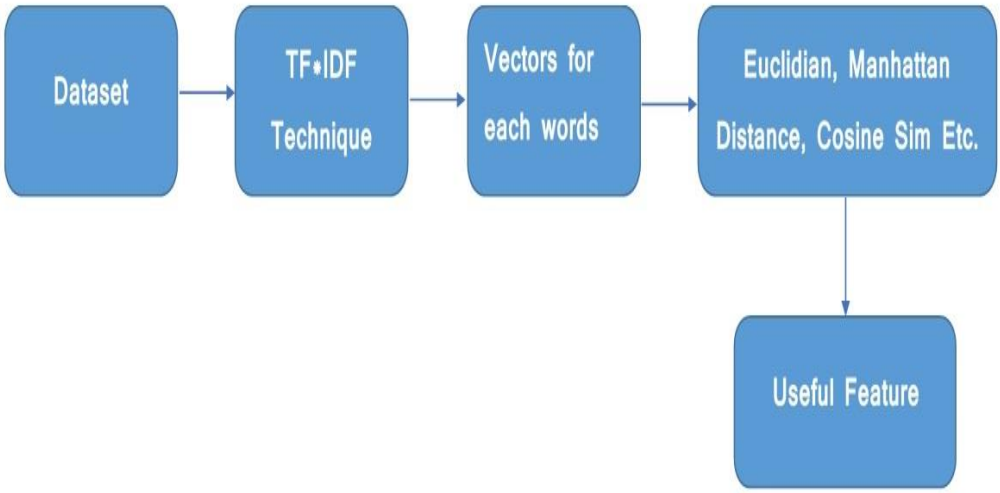
In information retrieval, tf–idf or TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is usually intended to reflect how important a word to a document is, in a collection/corpus. The value of tfidf increases proportionally to the number of times a word appears or repeats in the document and is offset by the number of documents in the corpus that contain the word, which helps to actually adjusts for the fact that some words appear more frequently in general. tf–idf is apperantly one of the most popular term-weighting schemes today. Typically, the weight of tf-idf is composed by two terms: the first computes the normalized Term Frequency (TF),which is the number of times a word appears or repeats in the document, divided by the total number of words in

that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears or repeats.

TF: Term freq. is a measure of how frequently a term occurs in a document. Since the length of every document is different, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is usually divided by the document length (which is the total number of terms in the document) just like a way of normalization:
TF(t) = (No. of times term t appears in a document) / (Total no. of terms in the document).

IDF: Inverse Document Frequency, it basically measures how important a term is. While computing term frequency, all the terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times in a sentence but have very little importance. Hence we are supposed to weigh down the frequent terms while scale up the rare ones, by computing the following:

IDF(t) = log_e(Total number of documents / Number of documents with term t in it).



Cosine Similarity Algorithm:

It's a metric that is used to measure the similarity between documents. It is far more efficient than measuring similarity based on the number of common words. Cosine_similarity calculates the cosine of the angles between the two vectors. So even if in Euclidean distance two vectors are far apart, cosine_similarity could be higher. Cosine similarity is basically the normalised dot product between two vectors. It is the cosine of the angle between 2 points in a multidimensional space. Points with smaller angles are more similar. Points with larger angles are more different.While harder to wrap your head around, cosine similarity solves some problems with Euclidean distance. Namely, magnitude.
We used Cosine Similarity to find the similar descriptions in the data set.

Implementing Cosine Similarity in Python - Cosine similarity is not the angle itself, but the cosine of the angle. So an angle which is snaller (sub 90 degrees) returns a larger similarit

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Adamic Adar Measure:

It is a measure used to compute the closeness of nodes based on their shared neighbors. x and y are 2 nodes (2 Movies) . N(one_node) is a function that return the set of adjacent nodes to one_node Adamic Adar(x,y)=∑u∈N(x)∩N(y)1log(N(u)) Otherwise, for each node u in common to x and y, add to the measure 1/log(N(u)). The quantity 1log(N(u)) determine the importance of u in the measure. If x and y share a node u that has a lot of adjacent nodes, this node is not really relevant. → N(u) is high → 1/log(N(u)) is not high if x and y share a node u that not has a lot of adjacent nodes, this node is really relevant. → N(u) is not high → 1/log(N(u)) is higher

Recommendation engine with a graph:

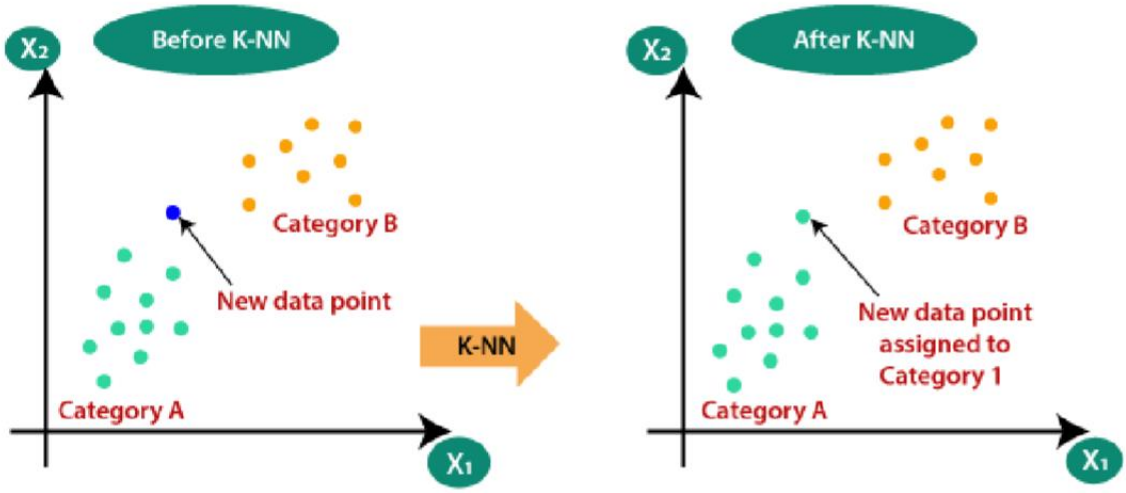
The purpose is to build a recommendation engine based on graph by using the Adamic Adar measure. The more the measure is high, the closest are the two nodes. The measures between all movies are NOT pre-calculated, in order to determine the list of recommendation films, we are going to explore the neighborhood of the target film.

KNN Algorithm:

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between

the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K-NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

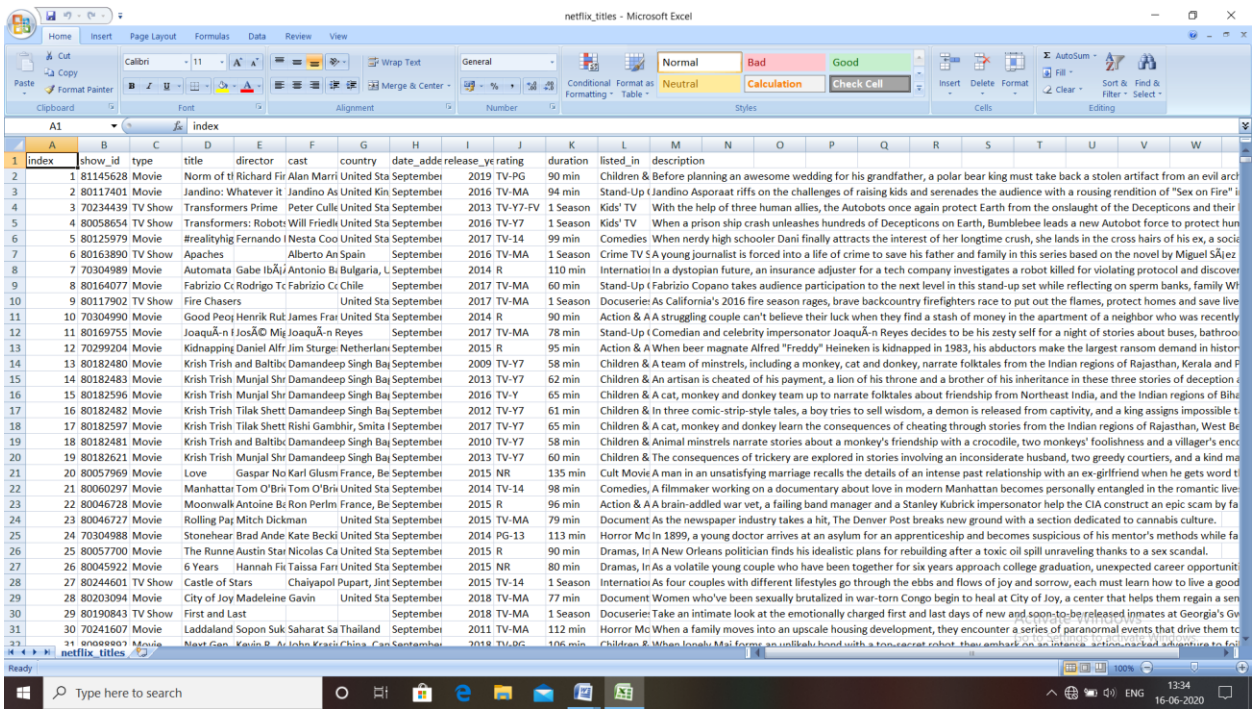
Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



3.Data preparation:

It was noticeable that there is a group of movies that have been rated by a few users, this implies that their ratings might be biased. In addition, there is a group of users that have rated few movies, then their ratings could be biased as well

In order to prepare the data to be used in recommender models, and based on the information described above. It is important to (i) Select the relevant data, which means reducing the data volume by improving the data quality and (ii) Normalize the data, eliminating some extreme values in the ratings per user. Having above benchmark will help us to improve not only the quality of the data but also the efficiency



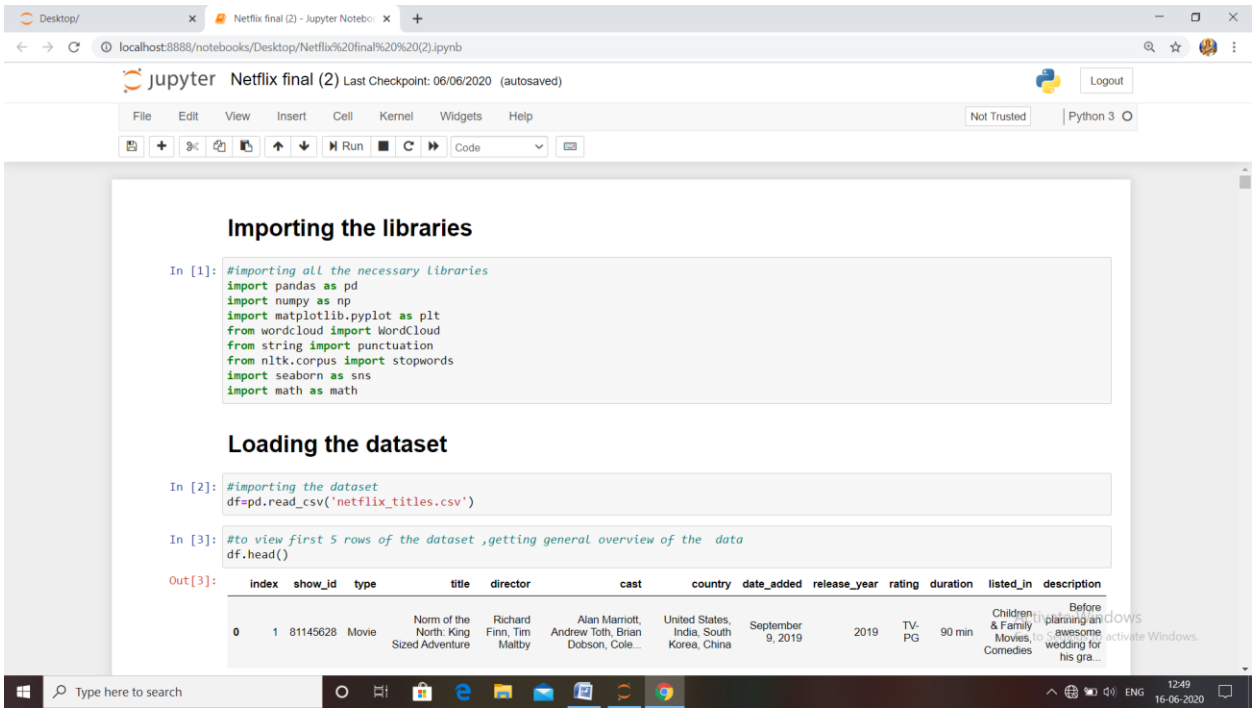
- The recommendations methodologies can be applied. When choosing between the implementation of Popularity, Collaborative Filtering, Content-

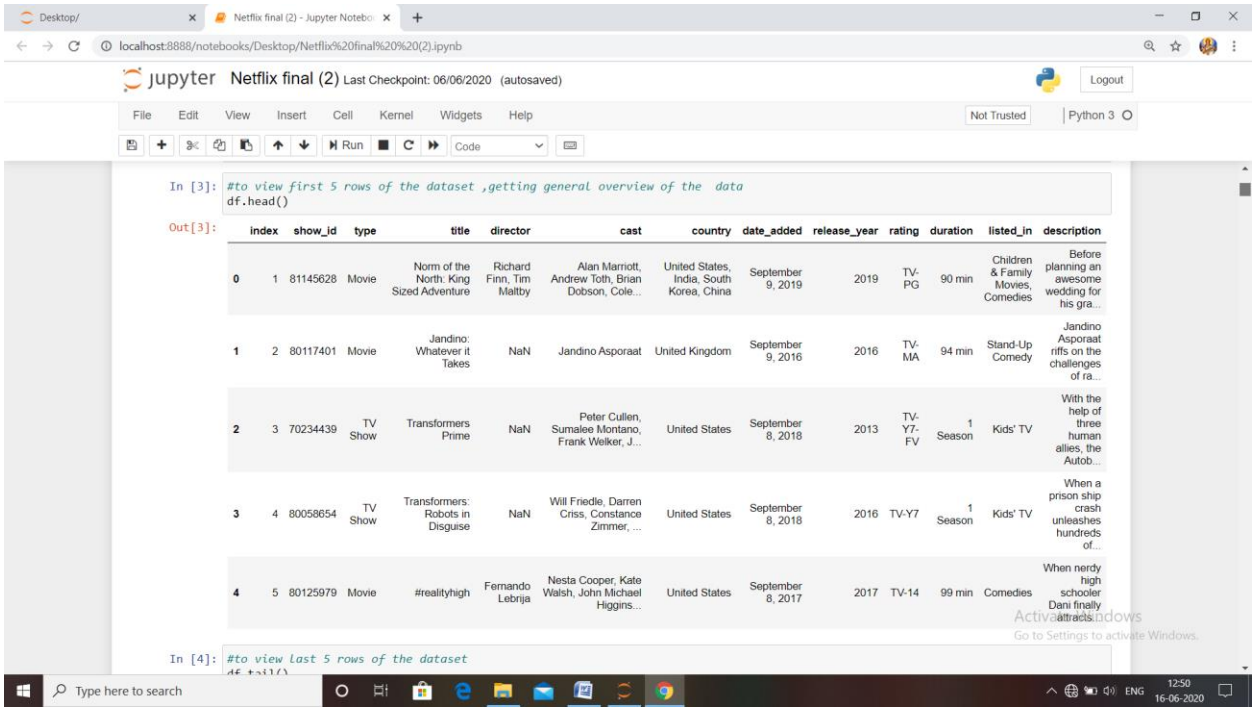
based filtering or Hybrid filtering, several criteria should be considered. For instance, the available information, because we just count with a data set of ratings, and the description of the movies

- Now, for the Content-based filtering, both approaches could be implemented the Memory-based techniques and the Model based-techniques. However, it is indispensable to choose the approaches that best suit our needs and the datasetcorrespond just to the titles, thus it is not possible to apply either Content-based filtering or Hybrid filtering for lack of information

4.Methodology:

4.1 Exploratory Data Analysis:





Desktop/ | Netflix final (2) - Jupyter Notebo | localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3

Run Code

In [3]:
```
#to view first 5 rows of the dataset ,getting general overview of the  data
df.head()
```

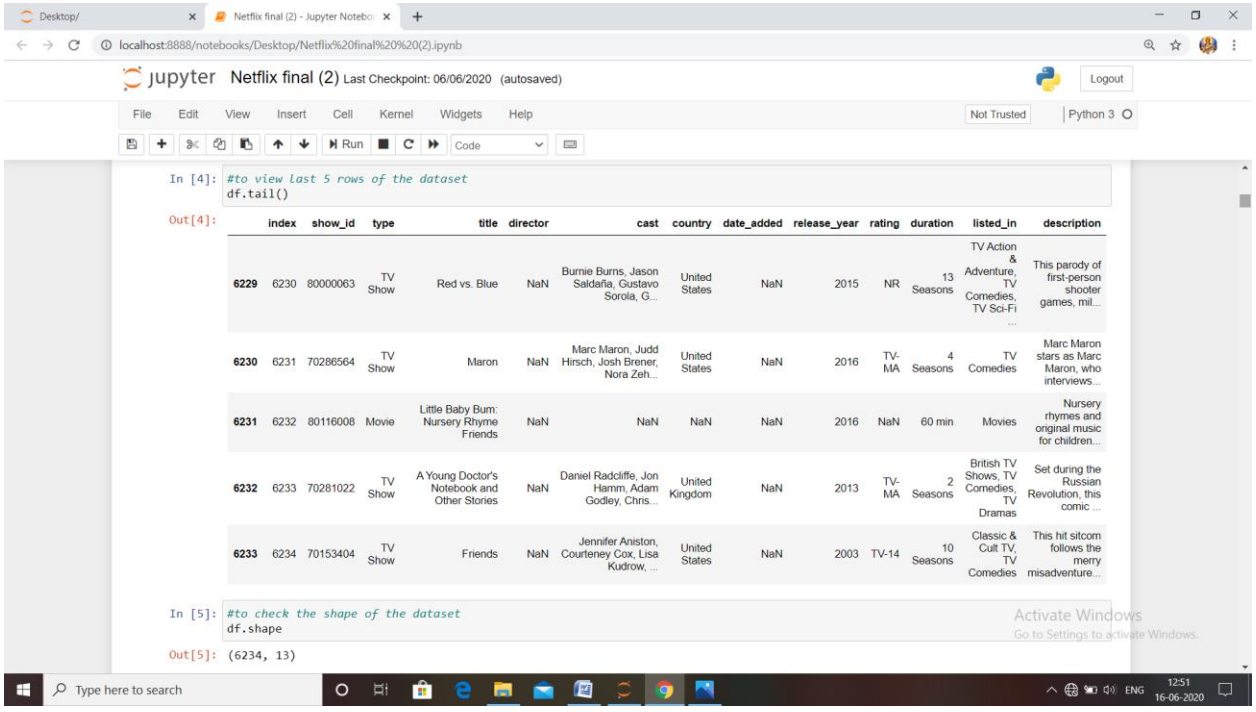
Out[3]:

	index	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	1	81145628	Movie	Norm of the North: King Sized Adventure	Richard Finn, Tim Maltby	Alan Marriott, Andrew Toth, Brian Dobson, Cole...	United States, India, South Korea, China	September 9, 2019	2019	TV-PG	90 min	Children & Family Movies, Comedies	Before planning an awesome wedding for his gra...
1	2	80117401	Movie	Jandino: Whatever it Takes	NaN	Jandino Asporaat	United Kingdom	September 9, 2016	2016	TV-MA	94 min	Stand-Up Comedy	Jandino Asporaat riffs on the challenges of ra...
2	3	70234439	TV Show	Transformers Prime	NaN	Peter Cullen, Sumalee Montano, Frank Welker, J...	United States	September 8, 2018	2013	TV-Y7-FV	1 Season	Kids' TV	With the help of three human allies, the Autob...
3	4	80058654	TV Show	Transformers: Robots in Disguise	NaN	Will Friedle, Darren Criss, Constance Zimmer, ...	United States	September 8, 2018	2016	TV-Y7	1 Season	Kids' TV	When a prison ship crash unleashes hundreds of...
4	5	80125979	Movie	#realityhigh	Fernando Lebrija	Nesta Cooper, Kate Walsh, John Michael Higgins...	United States	September 8, 2017	2017	TV-14	99 min	Comedies	When nerdy high schooler Dani finally attracts...

In [4]:
```
#to view last 5 rows of the dataset
```

Activate Windows Go to Settings to activate Windows.

Type here to search ENG 12:50 16-06-2020



Desktop/ | Netflix final (2) - Jupyter Notebo | localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3

Run Code

In [4]:
```
#to view last 5 rows of the dataset
df.tail()
```

Out[4]:

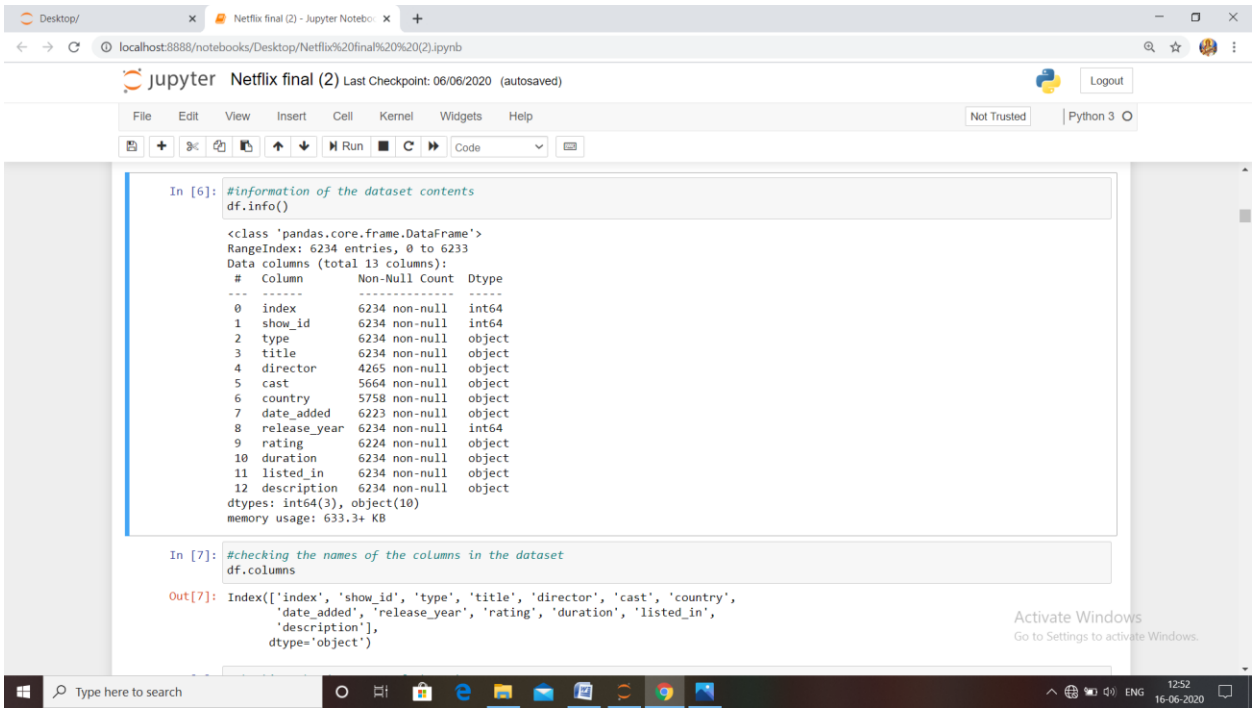
	index	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
6229	6230	80000063	TV Show	Red vs. Blue	NaN	Burnie Burns, Jason Saldaña, Gustavo Sorola, G...	United States	NaN	2015	NR	13 Seasons	TV Action & Adventure, TV Comedies, TV Sci-Fi ...	This parody of first-person shooter games, mil...
6230	6231	70286564	TV Show	Maron	NaN	Marc Maron, Judd Hirsch, Josh Brener, Nora Zeh...	United States	NaN	2016	TV-MA	4 Seasons	TV Comedies	Marc Maron stars as Marc Maron, who interviews...
6231	6232	80116008	Movie	Little Baby Bum: Nursery Rhyme Friends	NaN	NaN	NaN	NaN	2016	NaN	60 min	Movies	Nursery rhymes and original music for children...
6232	6233	70281022	TV Show	A Young Doctor's Notebook and Other Stories	NaN	Daniel Radcliffe, Jon Hamm, Adam Godley, Chris...	United Kingdom	NaN	2013	TV-MA	2 Seasons	British TV Shows, TV Comedies, TV Dramas	Set during the Russian Revolution, this comic ...
6233	6234	70153404	TV Show	Friends	NaN	Jennifer Aniston, Courteney Cox, Lisa Kudrow, ...	United States	NaN	2003	TV-14	10 Seasons	Classic & Cult TV, TV Comedies	This hit sitcom follows the merry misadventure...

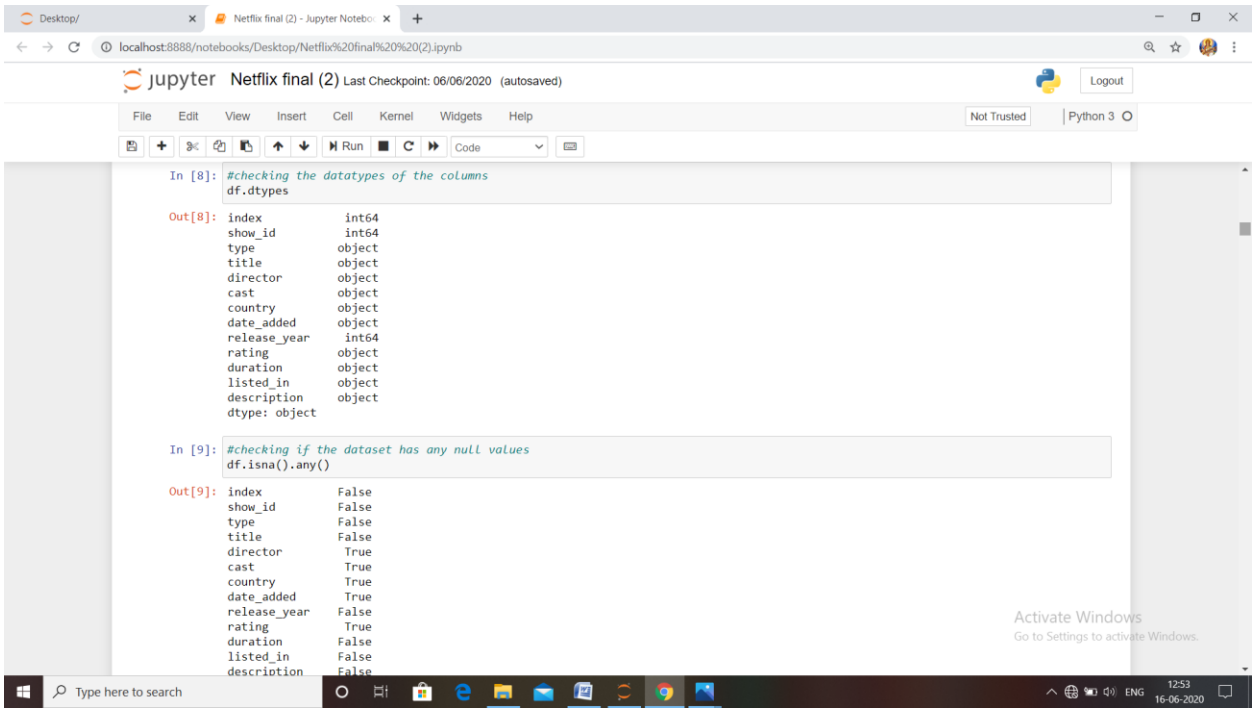
In [5]:
```
#to check the shape of the dataset
df.shape
```

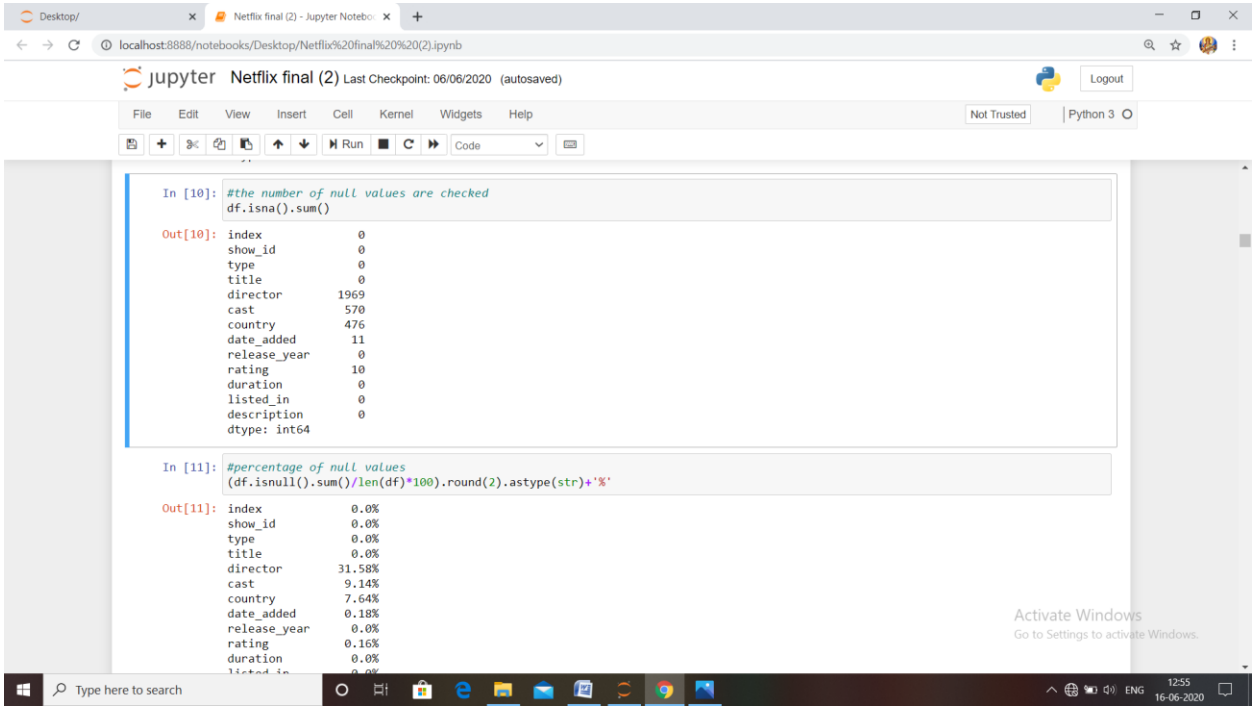
Out[5]: (6234, 13)

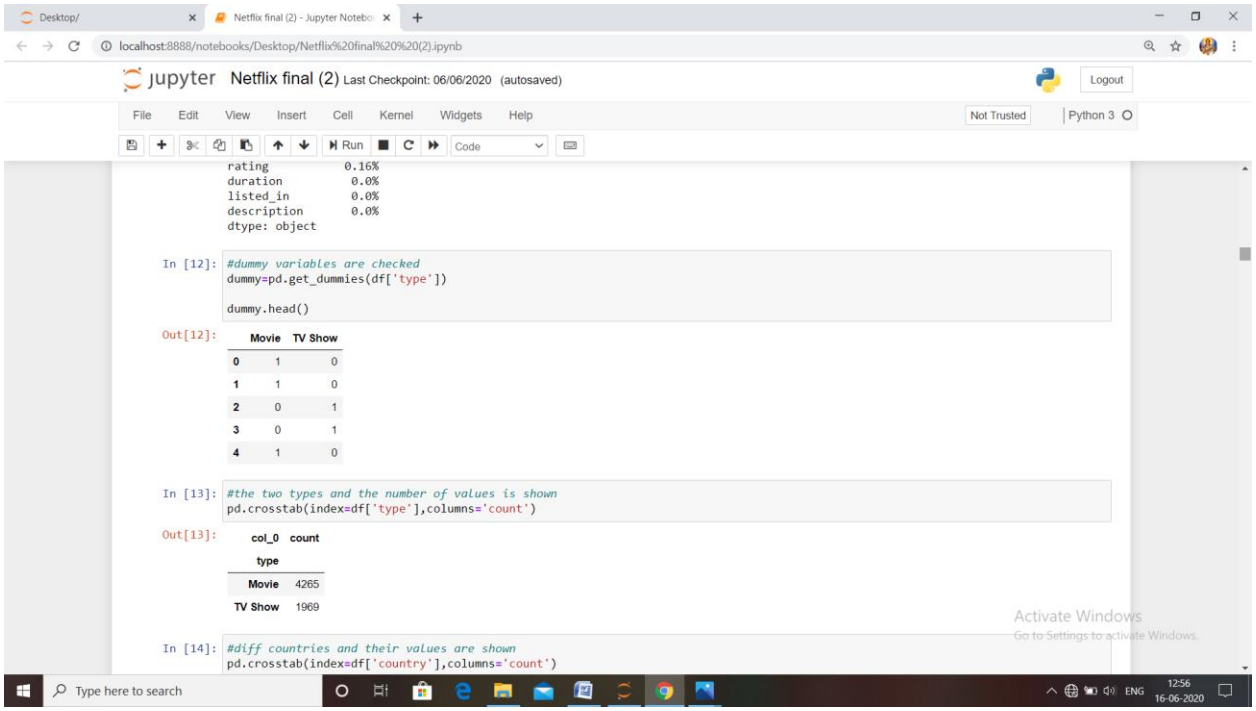
Activate Windows Go to Settings to activate Windows.

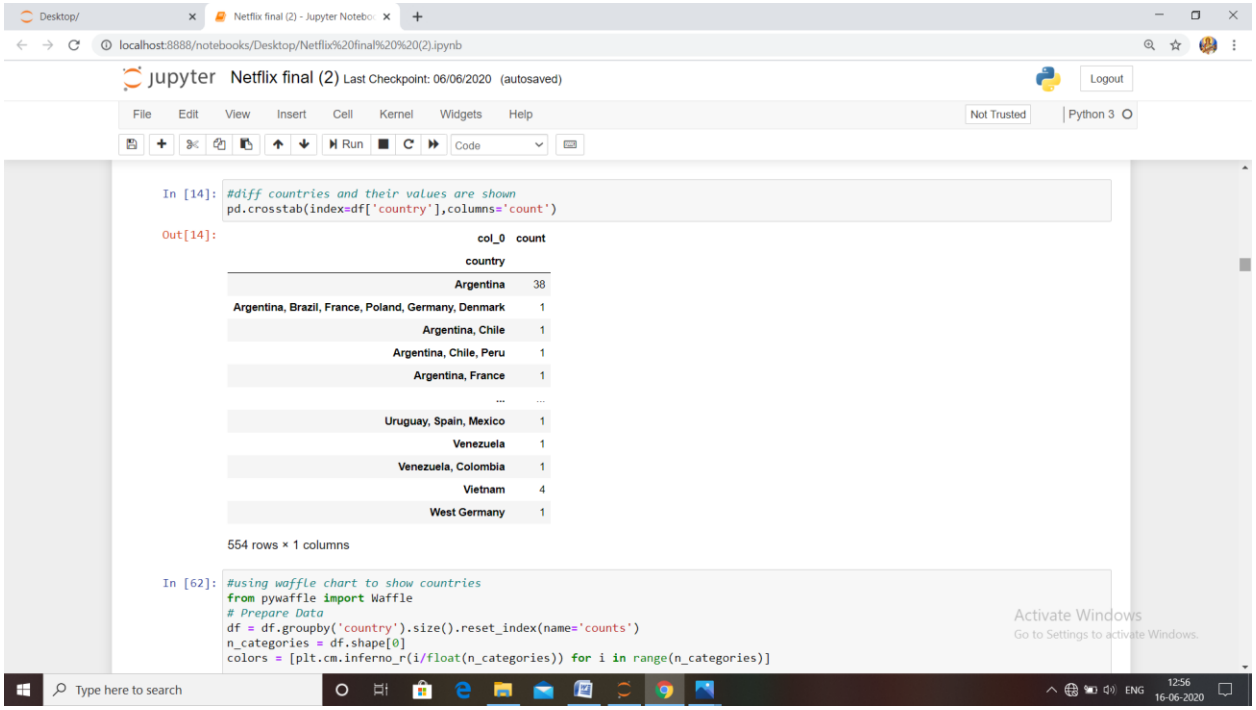
Type here to search ENG 12:51 16-06-2020

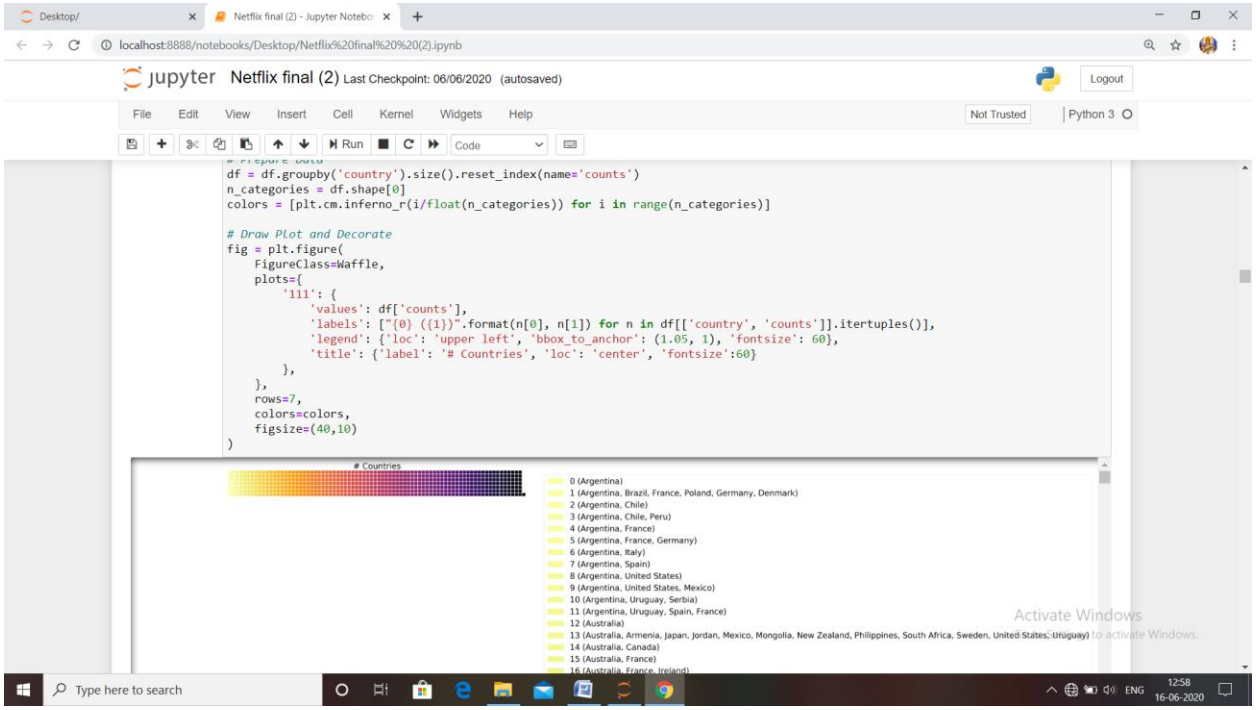

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [6]:
#information of the dataset contents
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6234 entries, 0 to 6233
Data columns (total 13 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 index 6234 non-null int64
1 show_id 6234 non-null int64
2 type 6234 non-null object
3 title 6234 non-null object
4 director 4265 non-null object
5 cast 5664 non-null object
6 country 5758 non-null object
7 date_added 6223 non-null object
8 release_year 6234 non-null int64
9 rating 6224 non-null object
10 duration 6234 non-null object
11 listed_in 6234 non-null object
12 description 6234 non-null object
dtypes: int64(3), object(10)
memory usage: 633.3+ KB
In [7]:
#checking the names of the columns in the dataset
df.columns
Out[7]:
Index(['index', 'show_id', 'type', 'title', 'director', 'cast', 'country',
'date_added', 'release_year', 'rating', 'duration', 'listed_in',
'description'],
dtype='object')
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:52
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [8]:
#checking the datatypes of the columns
df.dtypes
Out[8]:
index int64
show_id int64
type object
title object
director object
cast object
country object
date_added object
release_year int64
rating object
duration object
listed_in object
description object
dtype: object
In [9]:
#checking if the dataset has any null values
df.isna().any()
Out[9]:
index False
show_id False
type False
title False
director True
cast True
country True
date_added True
release_year False
rating True
duration False
listed_in False
description False
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:53
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [10]:
#the number of null values are checked
df.isna().sum()
Out[10]:
index 0
show_id 0
type 0
title 0
director 1969
cast 570
country 476
date_added 11
release_year 0
rating 10
duration 0
listed_in 0
description 0
dtype: int64
In [11]:
#percentage of null values
(df.isnull().sum()/len(df)*100).round(2).astype(str)+'%'
Out[11]:
index 0.0%
show_id 0.0%
type 0.0%
title 0.0%
director 31.58%
cast 9.14%
country 7.64%
date_added 0.18%
release_year 0.0%
rating 0.16%
duration 0.0%
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:55
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
rating 0.16%
duration 0.0%
listed_in 0.0%
description 0.0%
dtype: object
In [12]:
#dummy variables are checked
dummy=pd.get_dummies(df['type'])
dummy.head()
Out[12]:
Movie TV Show
0 1 0
1 1 0
2 0 1
3 0 1
4 1 0
In [13]:
#the two types and the number of values is shown
pd.crosstab(index=df['type'],columns='count')
Out[13]:
col_0 count
type
Movie 4265
TV Show 1969
In [14]:
#diff countries and their values are shown
pd.crosstab(index=df['country'],columns='count')
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:56
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [14]:
#diff countries and their values are shown
pd.crosstab(index=df['country'],columns='count')
Out[14]:
col_0 count
country
Argentina 38
Argentina, Brazil, France, Poland, Germany, Denmark 1
Argentina, Chile 1
Argentina, Chile, Peru 1
Argentina, France 1
... ...
Uruguay, Spain, Mexico 1
Venezuela 1
Venezuela, Colombia 1
Vietnam 4
West Germany 1
554 rows × 1 columns
In [62]:
#using waffle chart to show countries
from pywaffle import Waffle
# Prepare Data
df = df.groupby('country').size().reset_index(name='counts')
n_categories = df.shape[0]
colors = [plt.cm.inferno_r(i/float(n_categories)) for i in range(n_categories)]
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:56
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
df = df.groupby('country').size().reset_index(name='counts')
n_categories = df.shape[0]
colors = [plt.cm.inferno_r(i/float(n_categories)) for i in range(n_categories)]
# Draw Plot and Decorate
fig = plt.figure(
FigureClass=Waffle,
plots={
'111': {
'values': df['counts'],
'labels': ["{0} ({1})".format(n[0], n[1]) for n in df[['country', 'counts']].itertuples()],
'legend': {'loc': 'upper left', 'bbox_to_anchor': (1.05, 1), 'fontsize': 60},
'title': {'label': '# Countries', 'loc': 'center', 'fontsize':60}
},
},
rows=7,
colors=colors,
figsize=(40,10)
)
Countries
0 (Argentina)
1 (Argentina, Brazil, France, Poland, Germany, Denmark)
2 (Argentina, Chile)
3 (Argentina, Chile, Peru)
4 (Argentina, France)
5 (Argentina, France, Germany)
6 (Argentina, Italy)
7 (Argentina, Spain)
8 (Argentina, United States)
9 (Argentina, United States, Mexico)
10 (Argentina, Uruguay, Serbia)
11 (Argentina, Uruguay, Spain, France)
12 (Australia)
13 (Australia, Armenia, Japan, Jordan, Mexico, Mongolia, New Zealand, Philippines, South Africa, Sweden, United States, Uruguay)
14 (Australia, Canada)
15 (Australia, France)
16 (Australia, France, Ireland)
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
12:58
16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [15]:
```
#number of movies/tvshows released in each year is shown
pd.crosstab(index=df['release_year'],columns='count')
```

Out[15]:

release_year	col_0 count
1925	1
1942	2
1943	3
1944	3
1945	3
...	...
2016	830
2017	959
2018	1063
2019	843
2020	25

72 rows × 1 columns

In [16]:
```
#plotting a bar graph to show distribution of rating categories
plt.figure(figsize=(8,6))
df['rating'].value_counts(normalize=True).plot.bar(color=(0.1, 0.1, 0.1, 0.1),  edgecolor='blue')
plt.title('Distribution of rating categories')
plt.xlabel('rating')
plt.ylabel('relative frequency')
plt.show()
```


Type here to search

ENG 12:58 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)

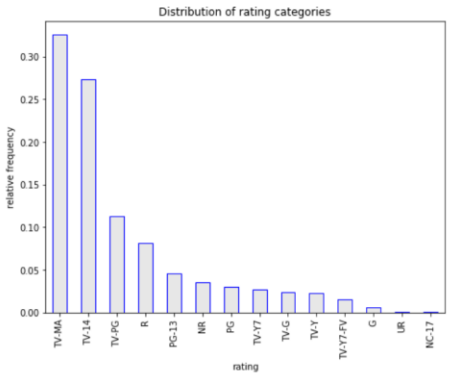
Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [16]:
```
#plotting a bar graph to show distribution of rating categories
plt.figure(figsize=(8,6))
df['rating'].value_counts(normalize=True).plot.bar(color=(0.1, 0.1, 0.1, 0.1),  edgecolor='blue')
plt.title('Distribution of rating categories')
plt.xlabel('rating')
plt.ylabel('relative frequency')
plt.show()
```



Activate Windows
Go to Settings to activate Windows.

Type here to search

ENG 12:59 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebo | localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)

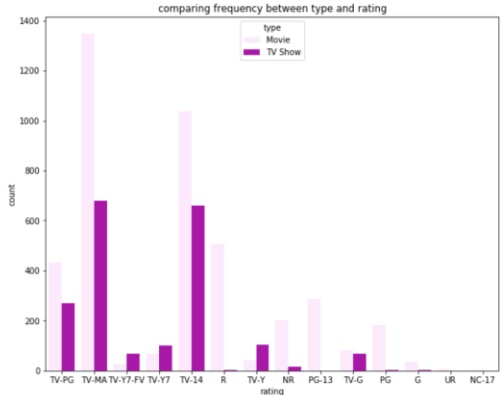
Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

rating

In [17]:
```
#plotting a countplot to show comparision between frequency of the type and rating
plt.figure(figsize=(10,8))
sns.countplot(x='rating',hue='type',color='m',data=df)
plt.title('comparing frequency between type and rating')
plt.legend="upper left corner"
plt.show()
```



Type here to search | ENG 13:00 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebo | localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

rating

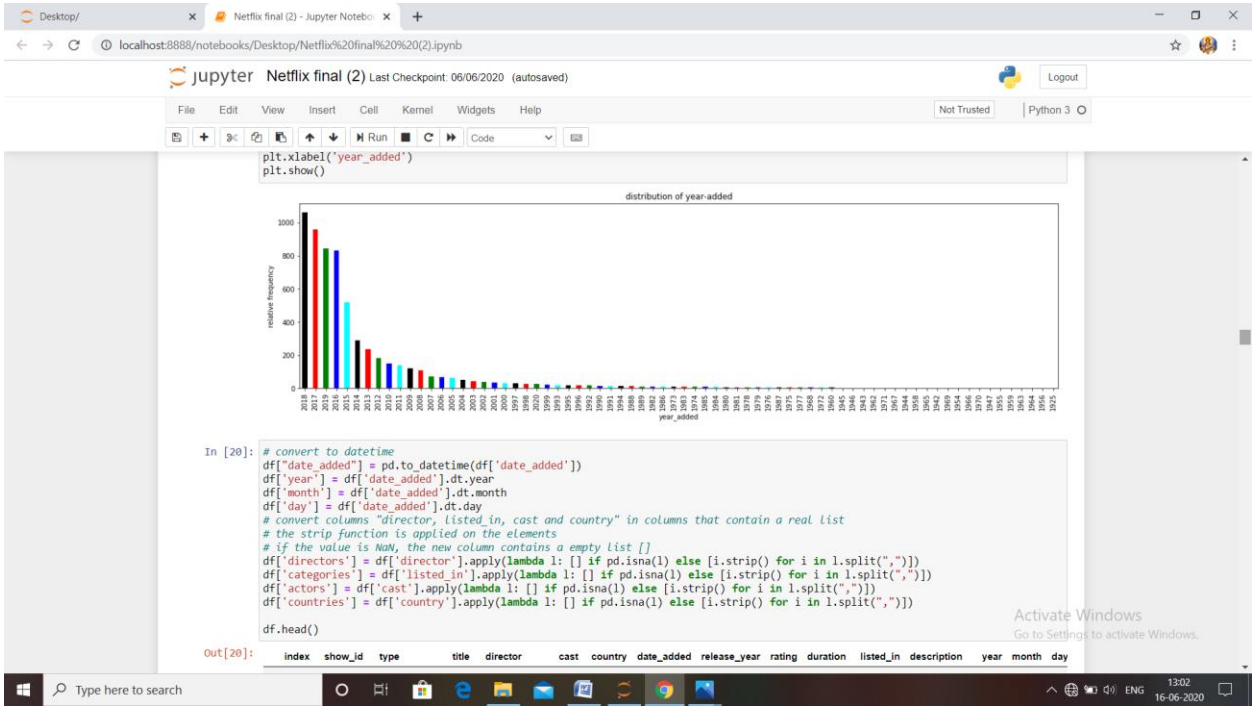
In [18]:
```
#to check each countries rating probabilities and normalizing it
for i in df['country'].unique():
    print(i)
    print(df[df['country']==i]['rating'].value_counts(normalize=True)*100)
    print('-'*40)
```

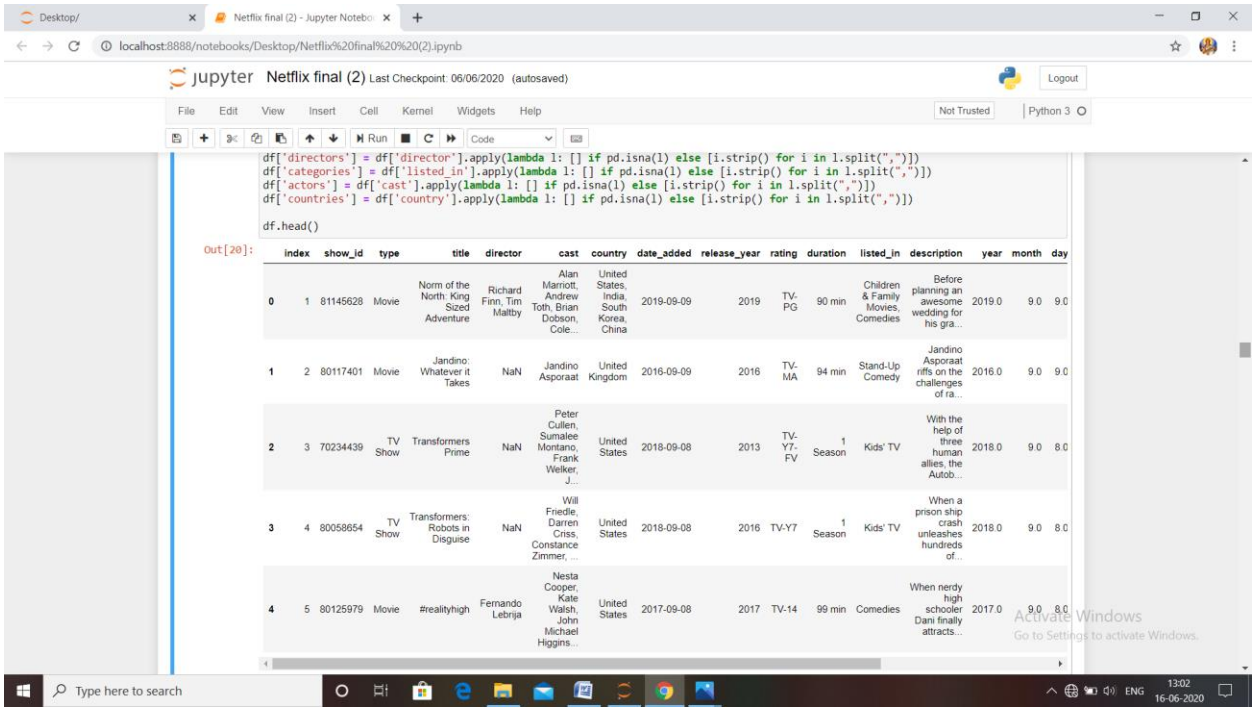
```
United States, India, South Korea, China
TV-PG    100.0
Name: rating, dtype: float64
----------------------------------------
United Kingdom
TV-MA       39.942529
TV-14       23.275862
TV-PG       17.816092
R            7.471264
TV-G         6.034483
NR           2.298851
TV-Y7        0.862069
PG           0.862069
TV-Y         0.574713
G            0.287356
TV-Y7-FV     0.287356
PG-13        0.287356
Name: rating, dtype: float64
----------------------------------------
United States
```

In [19]:
```
#plotting a bar graph to show distribution of year added
plt.figure(figsize=(20,5))
df['release_year'].value_counts().plot.bar(color=['black', 'red', 'green', 'blue', 'cyan'])
plt.title('distribution of year-added')
plt.ylabel('relative frequency')
plt.xlabel('year_added')
plt.show()
```

distribution of year-added

Type here to search | ENG 13:01 16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
plt.xlabel('year_added')
plt.show()
distribution of year-added
relative frequency
year_added
In [20]:
convert to datetime
df["date_added"] = pd.to_datetime(df['date_added'])
df['year'] = df['date_added'].dt.year
df['month'] = df['date_added'].dt.month
df['day'] = df['date_added'].dt.day
convert columns "director, listed_in, cast and country" in columns that contain a real list
the strip function is applied on the elements
if the value is NAN, the new column contains a empty list []
df['directors'] = df['director'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df['categories'] = df['listed_in'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df['actors'] = df['cast'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df['countries'] = df['country'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df.head()
Out[20]:
index show_id type title director cast country date_added release_year rating duration listed_in description year month day
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:02
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
df['directors'] = df['director'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df['categories'] = df['listed_in'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df['actors'] = df['cast'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df['countries'] = df['country'].apply(lambda l: [] if pd.isna(l) else [i.strip() for i in l.split(",")])
df.head()
Out[20]:
index show_id type title director cast country date_added release_year rating duration listed_in description year month day
0 1 81145628 Movie Norm of the North: King Sized Adventure Richard Finn, Tim Maltby Alan Marriott, Andrew Toth, Brian Dobson, Cole... United States, India, South Korea, China 2019-09-09 2019 TV-PG 90 min Children & Family Movies, Comedies Before planning an awesome wedding for his gra... 2019.0 9.0 9.0
1 2 80117401 Movie Jandino: Whatever it Takes NaN Jandino Asporaat United Kingdom 2016-09-09 2016 TV-MA 94 min Stand-Up Comedy Jandino Asporaat riffs on the challenges of ra... 2016.0 9.0 9.0
2 3 70234439 TV Show Transformers Prime NaN Peter Cullen, Sumalee Montano, Frank Welker, J... United States 2018-09-08 2013 TV-Y7-FV 1 Season Kids' TV With the help of three human allies, the Autob... 2018.0 9.0 8.0
3 4 80058654 TV Show Transformers: Robots in Disguise NaN Will Friedle, Darren Criss, Constance Zimmer, ... United States 2018-09-08 2016 TV-Y7 1 Season Kids' TV When a prison ship crash unleashes hundreds of... 2018.0 9.0 8.0
4 5 80125979 Movie #realityhigh Fernando Lebrija Nesta Cooper, Kate Walsh, John Michael Higgins... United States 2017-09-08 2017 TV-14 99 min Comedies When nerdy high schooler Dani finally attracts... 2017.0 9.0 8.0
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:02
16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter **Netflix final (2)** Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run Code

In [21]:
```
#dc=description_cleaning
dc = " ".join(df.description.values)
```

In [22]:
```
#using wordcloud to show movie titles
movie_titles = " ".join(df.title.values)
word_cloud = WordCloud(width=800,height=800,background_color='black',max_words=150).\
generate_from_text(movie_titles)
plt.figure(figsize=[8,8])
plt.imshow(word_cloud)
plt.show()
```

Activate Windows
Go to Settings to activate Windows.

Type here to search | ENG | 13:03 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

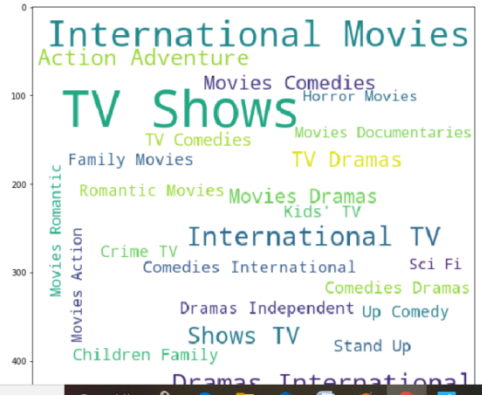
jupyter **Netflix final (2)** Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

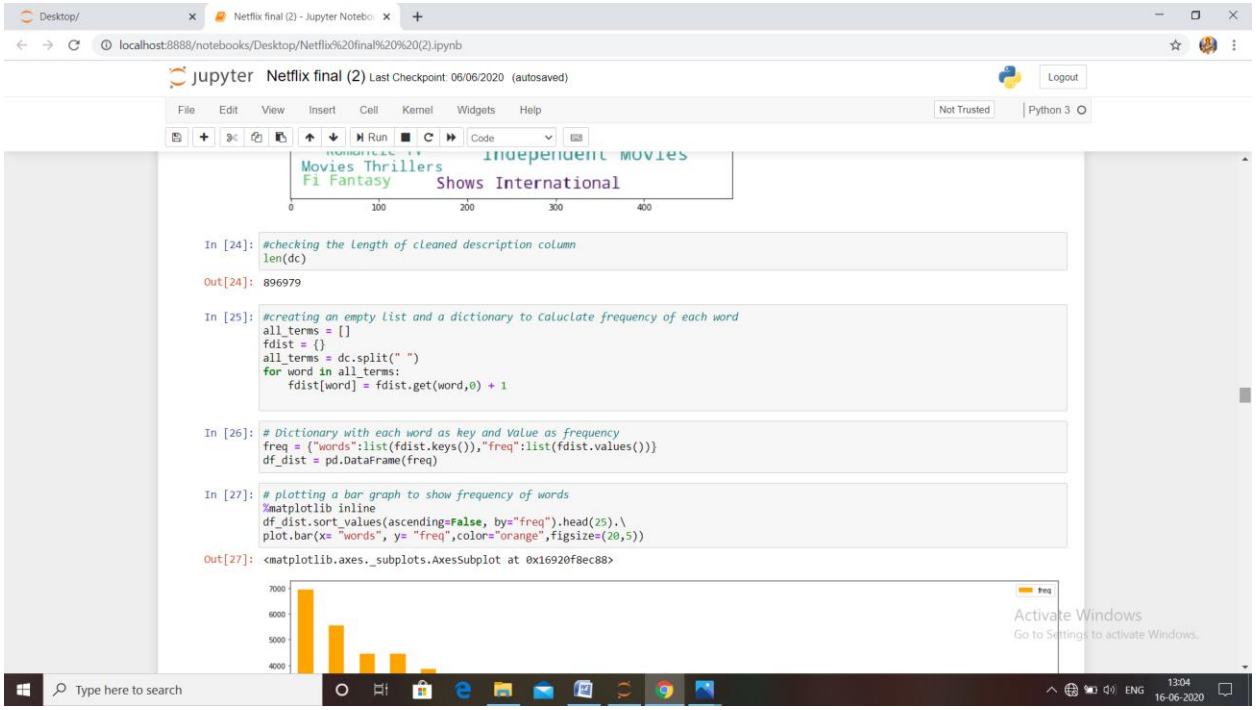
Run Code

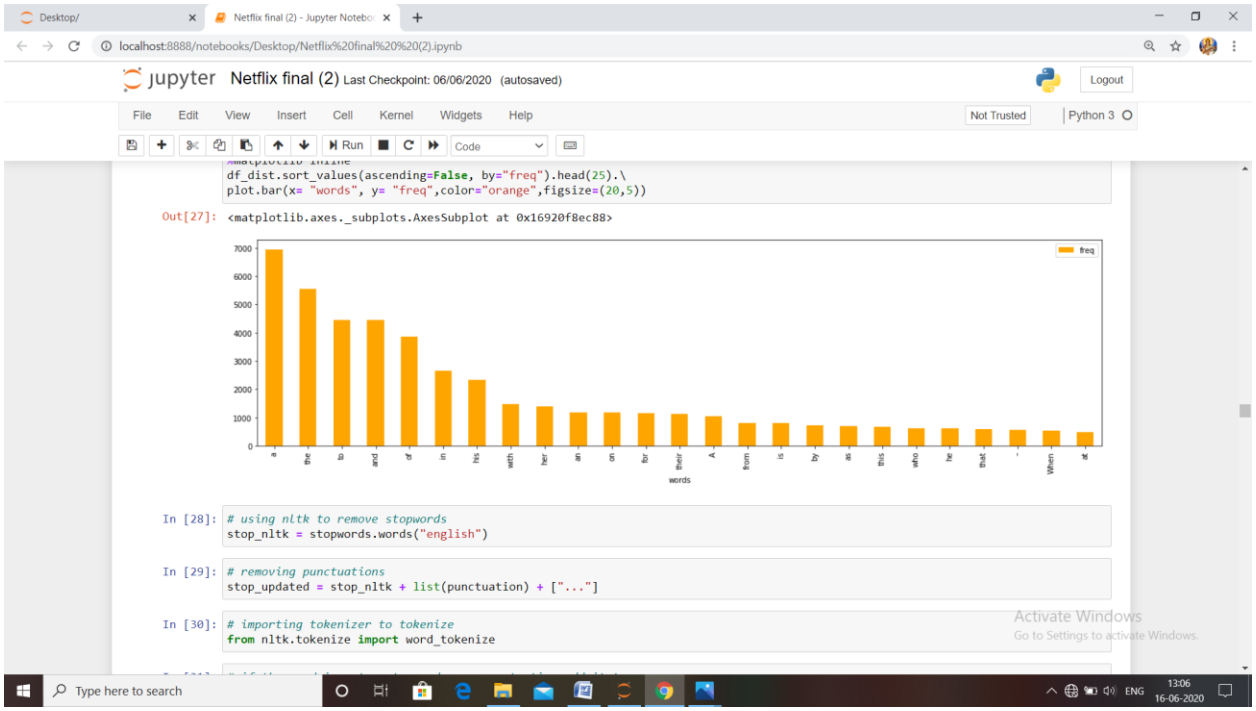
In [23]:
```
# using wordcloud to show genres
movie_genre = " ".join(df.listed_in.values)
word_cloud = WordCloud(width=500,height=500,background_color='white',max_words=30).\
generate_from_text(movie_genre)
plt.figure(figsize=[10,10])
plt.imshow(word_cloud)
plt.show()
```



Activate Windows
Go to Settings to activate Windows.

Type here to search | ENG | 13:04 16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Movies Thrillers
Fi Fantasy
Shows International
0 100 200 300 400
In [24]:
#checking the length of cleaned description column
len(dc)
Out[24]: 896979
In [25]:
#creating an empty list and a dictionary to Caluclate frequency of each word
all_terms = []
fdist = {}
all_terms = dc.split(" ")
for word in all_terms:
fdist[word] = fdist.get(word,0) + 1
In [26]:
# Dictionary with each word as key and Value as frequency
freq = {"words":list(fdist.keys()),"freq":list(fdist.values())}
df_dist = pd.DataFrame(freq)
In [27]:
plotting a bar graph to show frequency of words
%matplotlib inline
df_dist.sort_values(ascending=False, by="freq").head(25).\
plot.bar(x= "words", y= "freq",color="orange",figsize=(20,5))
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x16920f8ec88>
7000
6000
5000
4000
freq
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:04
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
df_dist.sort_values(ascending=False, by="freq").head(25).\
plot.bar(x= "words", y= "freq",color="orange",figsize=(20,5))
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x16920f8ec88>
7000
6000
5000
4000
3000
2000
1000
0
freq
a
the
to
and
of
in
his
with
her
an
on
for
their
A
from
is
by
as
this
who
he
that
-
When
at
words
In [28]:
# using nltk to remove stopwords
stop_nltk = stopwords.words("english")
In [29]:
# removing punctuations
stop_updated = stop_nltk + list(punctuation) + ["..."]
In [30]:
# importing tokenizer to tokenize
from nltk.tokenize import word_tokenize
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:06
16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter **Netflix final (2)** Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

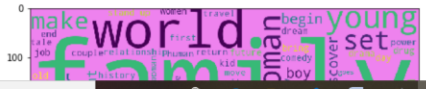
In [30]:
```
# importing tokenizer to tokenize
from nltk.tokenize import word_tokenize
```

In [31]:
```
# if the word is not a stopword or a punctuation add it to res
def  clean_txt(sent):
    tokens = word_tokenize(sent.lower())
    res=""
    for term in tokens :
                if (term not in stop_updated and len(term) > 2):
                    res = res+" "+term
    res=res.strip()
    return res
```

In [32]:
```
#applying clean text function to description
df['cd'] = df.description.apply(clean_txt)
```

In [33]:
```
#adding the values to reviews_combined column
reviews_combined = " ".join(df.cd.values)
```

In [34]:
```
# Word cloud on cleaned dataset
reviews_combined = " ".join(df.cd.values)
word_cloud = WordCloud(width=800,height=800,background_color='violet',max_words=150).\
generate_from_text(reviews_combined)
plt.figure(figsize=[8,8])
plt.imshow(word_cloud)
plt.show()
```



Activate Windows
Go to Settings to activate Windows.

Type here to search | ENG | 13:06 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

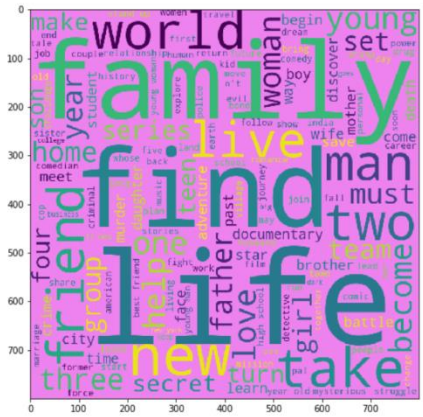
jupyter **Netflix final (2)** Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

```
plt.show()
```



In [35]:
```
#creating an empty list and a dictionary to Caluclate frequency of each word in reviews_combined
all_terms = []
fdist = {}
```

Activate Windows
Go to Settings to activate Windows.

Type here to search | ENG | 13:07 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter **Netflix final (2)** Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Code

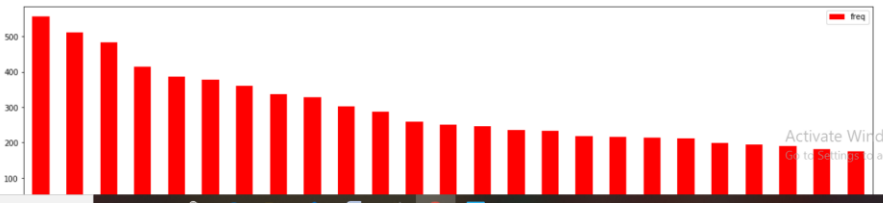
0 100 200 300 400 500 600 700

In [35]:
```
#creating an empty list and a dictionary to Caluclate frequency of each word in reviews_combined
all_terms = []
fdist = {}
all_terms = reviews_combined.split(" ")
for word in all_terms:
    fdist[word] = fdist.get(word,0) + 1
```

In [36]:
```
# Dictionary with each word as key and Value as frequency
freq = {"words":list(fdist.keys()),"freq":list(fdist.values())}
df_dist = pd.DataFrame
```

In [37]:
```
#plotting a bar graph between words and frequency
%matplotlib inline
df_dist.sort_values(ascending=False, by="freq").head(25).\
plot.bar(x= "words", y= "freq",color='red',figsize=(20,5))
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x16920b31860>



Type here to search

ENG 13:07 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebook

localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter **Netflix final (2)** Last Checkpoint: 06/06/2020 (autosaved)

Logout

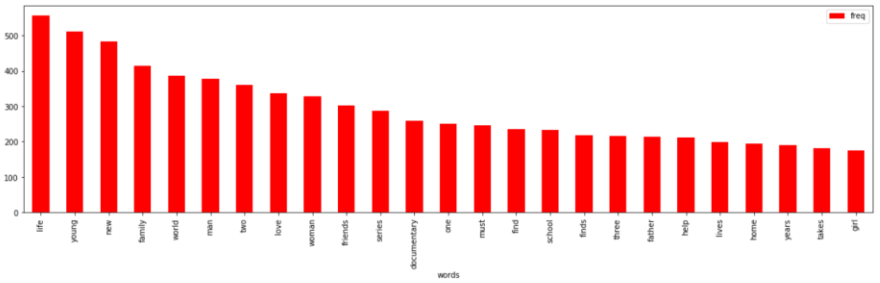
File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run | Code

```
df_dist.sort_values(ascending=False, by="freq").head(25).\
plot.bar(x= "words", y= "freq",color='red',figsize=(20,5))
```

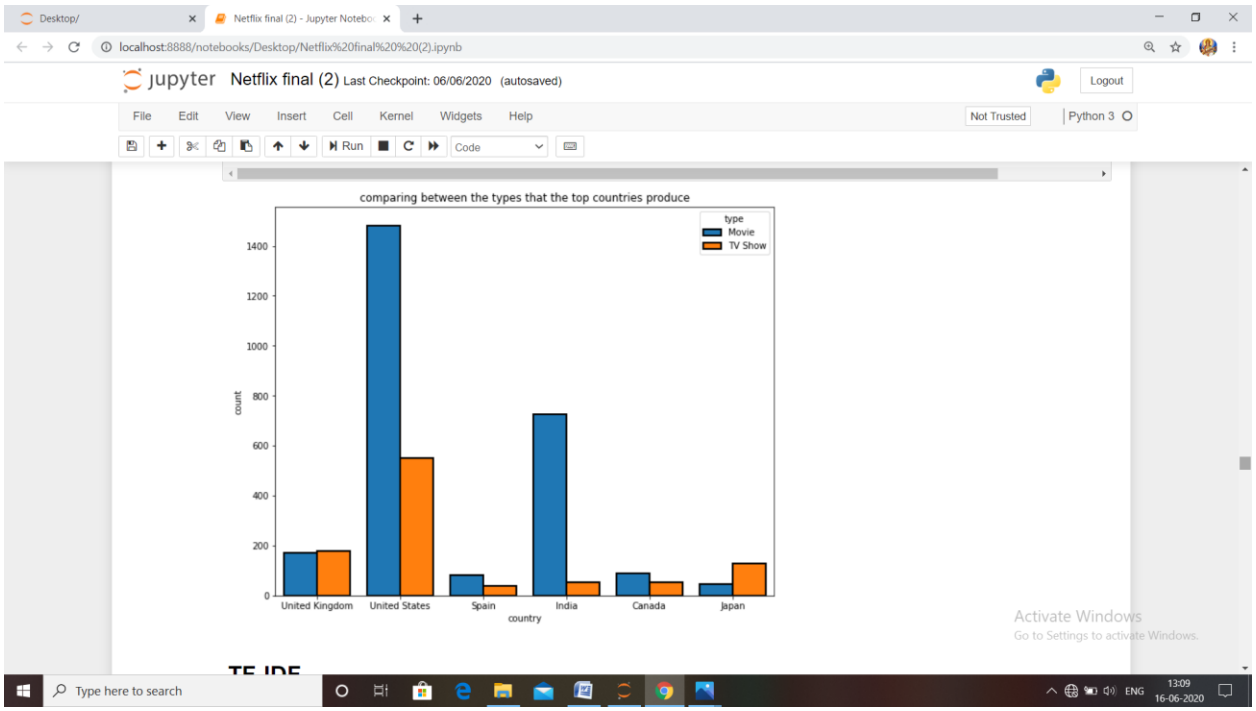
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x16920b31860>

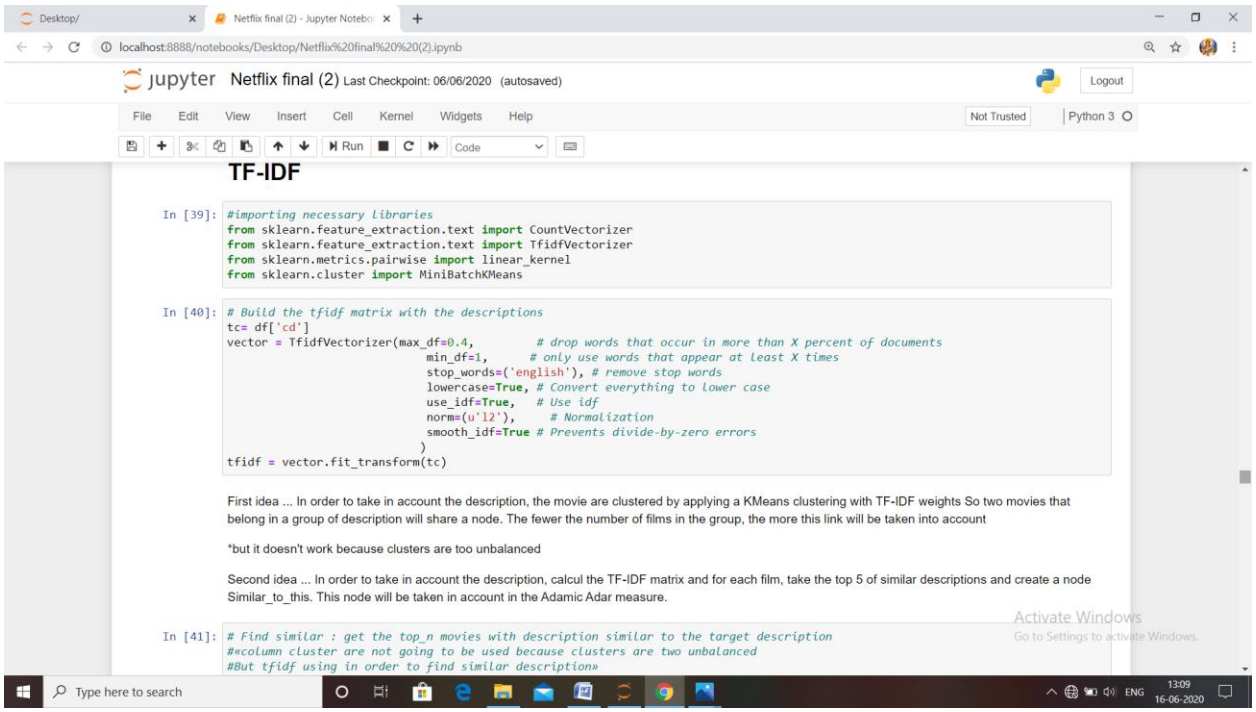


In [38]:
```
#plotting a bar graph to compare countries and their production of movies an tv shows
top_productive_countries=df[(df['country']=='United States')|(df['country']=='India')|(df['country']=='United Kingdom')|(df['cou
                            (df['country']=='Canada')|(df['country']=='Spain')]
plt.figure(figsize=(10,8))
sns.countplot(x='country',hue='type',saturation=1,edgecolor=(0,0,0),linewidth=2,data=top_productive_countries)
plt.title('comparing between the types that the top countries produce')
plt.show()
```


Type here to search

ENG 13:08 16-06-2020





TF-IDF

In [39]:
```
#importing necessary libraries
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.cluster import MiniBatchKMeans
```

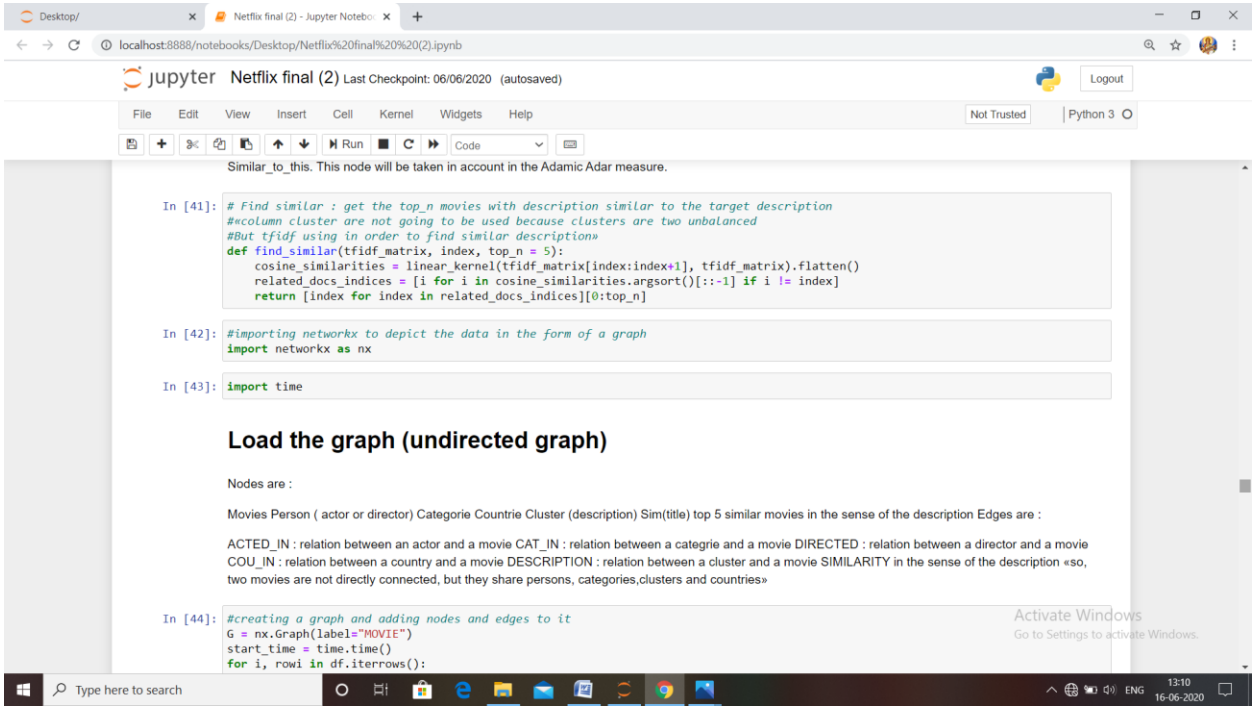
In [40]:
```
# Build the tfidf matrix with the descriptions
tc= df['cd']
vector = TfidfVectorizer(max_df=0.4,         # drop words that occur in more than X percent of documents
                         min_df=1,      # only use words that appear at least X times
                         stop_words=('english'), # remove stop words
                         lowercase=True, # Convert everything to lower case
                         use_idf=True,   # Use idf
                         norm=(u'l2'),     # Normalization
                         smooth_idf=True # Prevents divide-by-zero errors
                        )
tfidf = vector.fit_transform(tc)
```

First idea ... In order to take in account the description, the movie are clustered by applying a KMeans clustering with TF-IDF weights So two movies that belong in a group of description will share a node. The fewer the number of films in the group, the more this link will be taken into account

*but it doesn't work because clusters are too unbalanced

Second idea ... In order to take in account the description, calcul the TF-IDF matrix and for each film, take the top 5 of similar descriptions and create a node Similar_to_this. This node will be taken in account in the Adamic Adar measure.

In [41]:
```
# Find similar : get the top_n movies with description similar to the target description
#«column cluster are not going to be used because clusters are two unbalanced
#But tfidf using in order to find similar description»
```



Similar_to_this. This node will be taken in account in the Adamic Adar measure.

In [41]:
```
# Find similar : get the top_n movies with description similar to the target description
#«column cluster are not going to be used because clusters are two unbalanced
#But tfidf using in order to find similar description»
def find_similar(tfidf_matrix, index, top_n = 5):
    cosine_similarities = linear_kernel(tfidf_matrix[index:index+1], tfidf_matrix).flatten()
    related_docs_indices = [i for i in cosine_similarities.argsort()[::-1] if i != index]
    return [index for index in related_docs_indices][0:top_n]
```

In [42]:
```
#importing networkx to depict the data in the form of a graph
import networkx as nx
```

In [43]:
```
import time
```

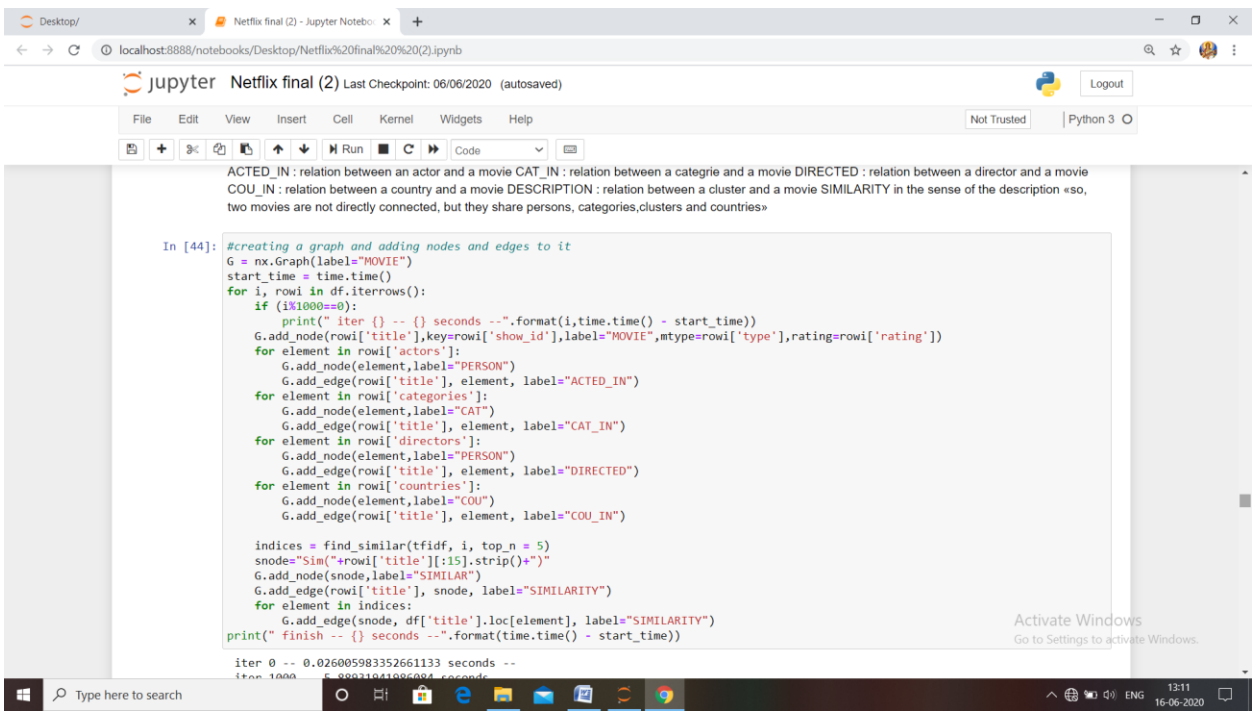
Load the graph (undirected graph)

Nodes are :

Movies Person (actor or director) Categorie Countrie Cluster (description) Sim(title) top 5 similar movies in the sense of the description Edges are :

ACTED_IN : relation between an actor and a movie CAT_IN : relation between a categrie and a movie DIRECTED : relation between a director and a movie COU_IN : relation between a country and a movie DESCRIPTION : relation between a cluster and a movie SIMILARITY in the sense of the description «so, two movies are not directly connected, but they share persons, categories,clusters and countries»

In [44]:
```
#creating a graph and adding nodes and edges to it
G = nx.Graph(label="MOVIE")
start_time = time.time()
for i, rowi in df.iterrows():
```

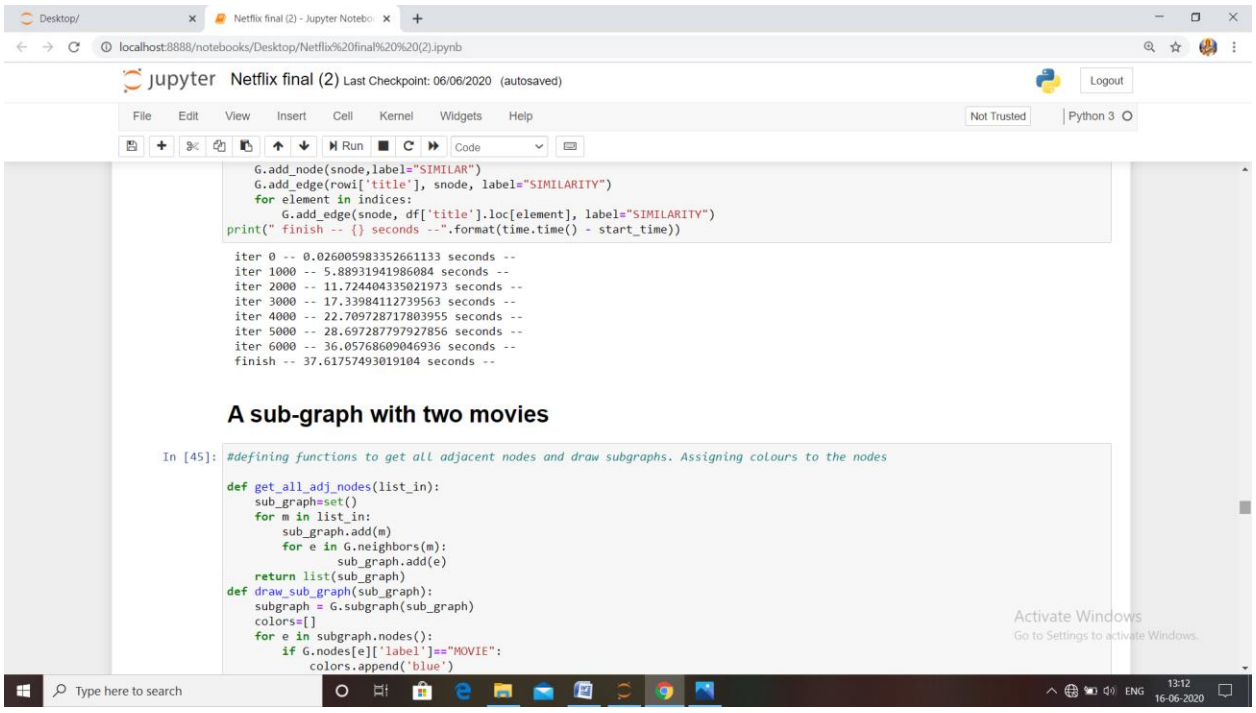



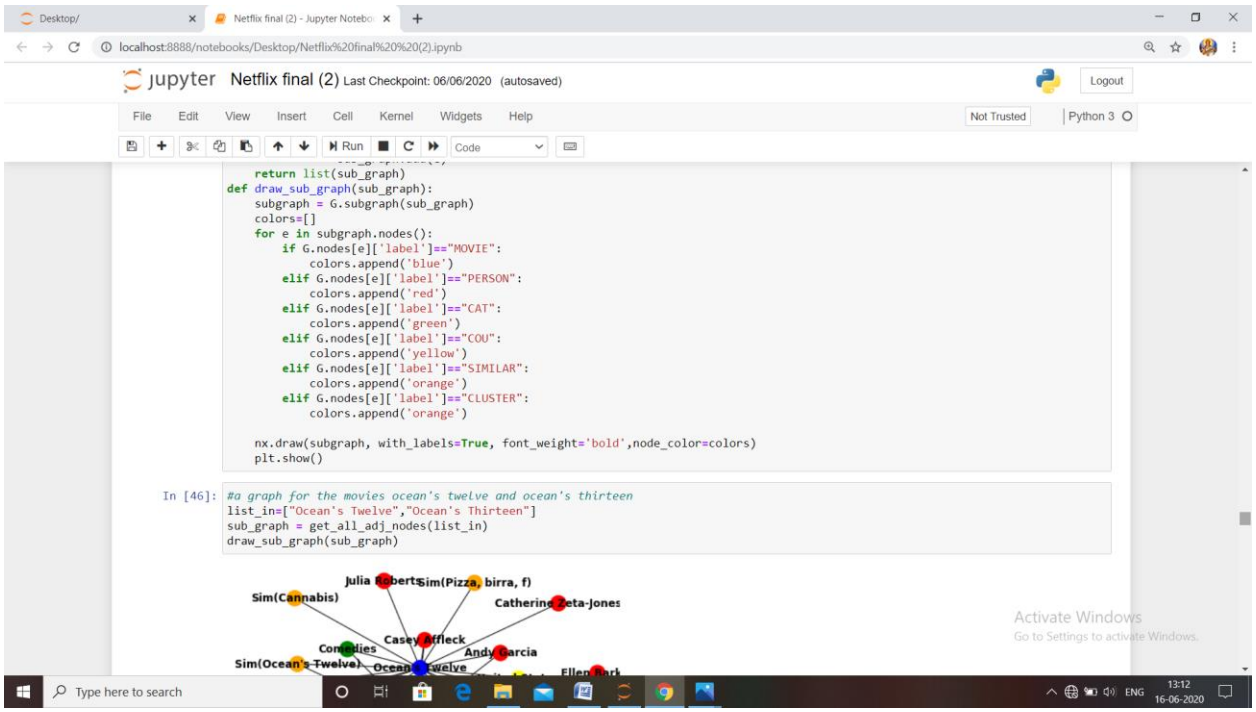
ACTED_IN : relation between an actor and a movie CAT_IN : relation between a categrie and a movie DIRECTED : relation between a director and a movie COU_IN : relation between a country and a movie DESCRIPTION : relation between a cluster and a movie SIMILARITY in the sense of the description «so, two movies are not directly connected, but they share persons, categories,clusters and countries»

In [44]:
```
#creating a graph and adding nodes and edges to it
G = nx.Graph(label="MOVIE")
start_time = time.time()
for i, rowi in df.iterrows():
    if (i%1000==0):
        print(" iter {} -- {} seconds --".format(i,time.time() - start_time))
    G.add_node(rowi['title'],key=rowi['show_id'],label="MOVIE",mtype=rowi['type'],rating=rowi['rating'])
    for element in rowi['actors']:
        G.add_node(element,label="PERSON")
        G.add_edge(rowi['title'], element, label="ACTED_IN")
    for element in rowi['categories']:
        G.add_node(element,label="CAT")
        G.add_edge(rowi['title'], element, label="CAT_IN")
    for element in rowi['directors']:
        G.add_node(element,label="PERSON")
        G.add_edge(rowi['title'], element, label="DIRECTED")
    for element in rowi['countries']:
        G.add_node(element,label="COU")
        G.add_edge(rowi['title'], element, label="COU_IN")

    indices = find_similar(tfidf, i, top_n = 5)
    snode="Sim("+rowi['title'][:15].strip()+")"
    G.add_node(snode,label="SIMILAR")
    G.add_edge(rowi['title'], snode, label="SIMILARITY")
    for element in indices:
        G.add_edge(snode, df['title'].loc[element], label="SIMILARITY")
print(" finish -- {} seconds --".format(time.time() - start_time))
```

```
 iter 0 -- 0.026005983352661133 seconds --
 iter 1000 -- 6.889319419860840 seconds --
```


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
G.add_node(snode,label="SIMILAR")
G.add_edge(rowi['title'], snode, label="SIMILARITY")
for element in indices:
G.add_edge(snode, df['title'].loc[element], label="SIMILARITY")
print(" finish -- {} seconds --".format(time.time() - start_time))
iter 0 -- 0.026005983352661133 seconds --
iter 1000 -- 5.889319419860840 seconds --
iter 2000 -- 11.724404335021973 seconds --
iter 3000 -- 17.33984112739563 seconds --
iter 4000 -- 22.709728717803955 seconds --
iter 5000 -- 28.697287797927856 seconds --
iter 6000 -- 36.05768609046936 seconds --
finish -- 37.61757493019104 seconds --
A sub-graph with two movies
In [45]:
#defining functions to get all adjacent nodes and draw subgraphs. Assigning colours to the nodes
def get_all_adj_nodes(list_in):
sub_graph=set()
for m in list_in:
sub_graph.add(m)
for e in G.neighbors(m):
sub_graph.add(e)
return list(sub_graph)
def draw_sub_graph(sub_graph):
subgraph = G.subgraph(sub_graph)
colors=[]
for e in subgraph.nodes():
if G.nodes[e]['label']=="MOVIE":
colors.append('blue')
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:12
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
return list(sub_graph)
def draw_sub_graph(sub_graph):
subgraph = G.subgraph(sub_graph)
colors=[]
for e in subgraph.nodes():
if G.nodes[e]['label']=="MOVIE":
colors.append('blue')
elif G.nodes[e]['label']=="PERSON":
colors.append('red')
elif G.nodes[e]['label']=="CAT":
colors.append('green')
elif G.nodes[e]['label']=="COU":
colors.append('yellow')
elif G.nodes[e]['label']=="SIMILAR":
colors.append('orange')
elif G.nodes[e]['label']=="CLUSTER":
colors.append('orange')
nx.draw(subgraph, with_labels=True, font_weight='bold',node_color=colors)
plt.show()
In [46]:
#a graph for the movies ocean's twelve and ocean's thirteen
list_in=["Ocean's Twelve","Ocean's Thirteen"]
sub_graph = get_all_adj_nodes(list_in)
draw_sub_graph(sub_graph)
Julia Roberts
Sim(Pizza, birra, f)
Sim(Cannabis)
Catherine Zeta-Jones
Casey Affleck
Andy Garcia
Sim(Ocean's Twelve)
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:12
16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebo | localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

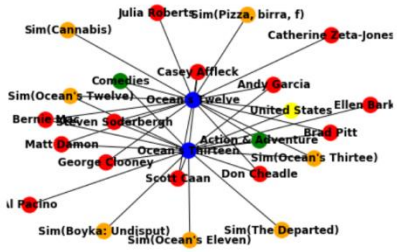
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

In [46]:
```
#a graph for the movies ocean's twelve and ocean's thirteen
list_in=["Ocean's Twelve","Ocean's Thirteen"]
sub_graph = get_all_adj_nodes(list_in)
draw_sub_graph(sub_graph)
```



Recommendation function

In [47]:
```
#defining a function to get recommendations
#Explore the neighborhood of the target film → this is a list of actor, director, country, categorie
#Explore the neighborhood of each neighbor → discover the movies that share a node with the target field
#Calculate Adamic Adar measure → final results
```

Activate Windows
Go to Settings to activate Windows.

Type here to search | ENG 13:13 16-06-2020

Desktop/ | Netflix final (2) - Jupyter Notebo | localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb

jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

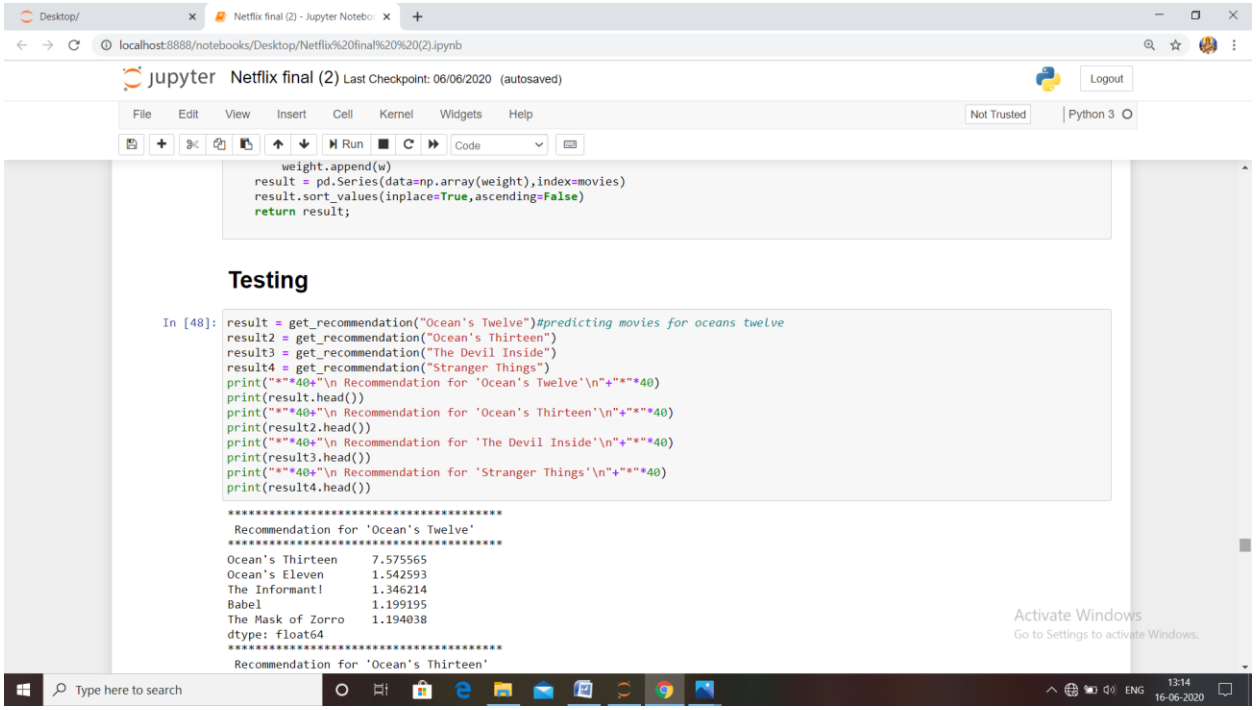
Run | Code

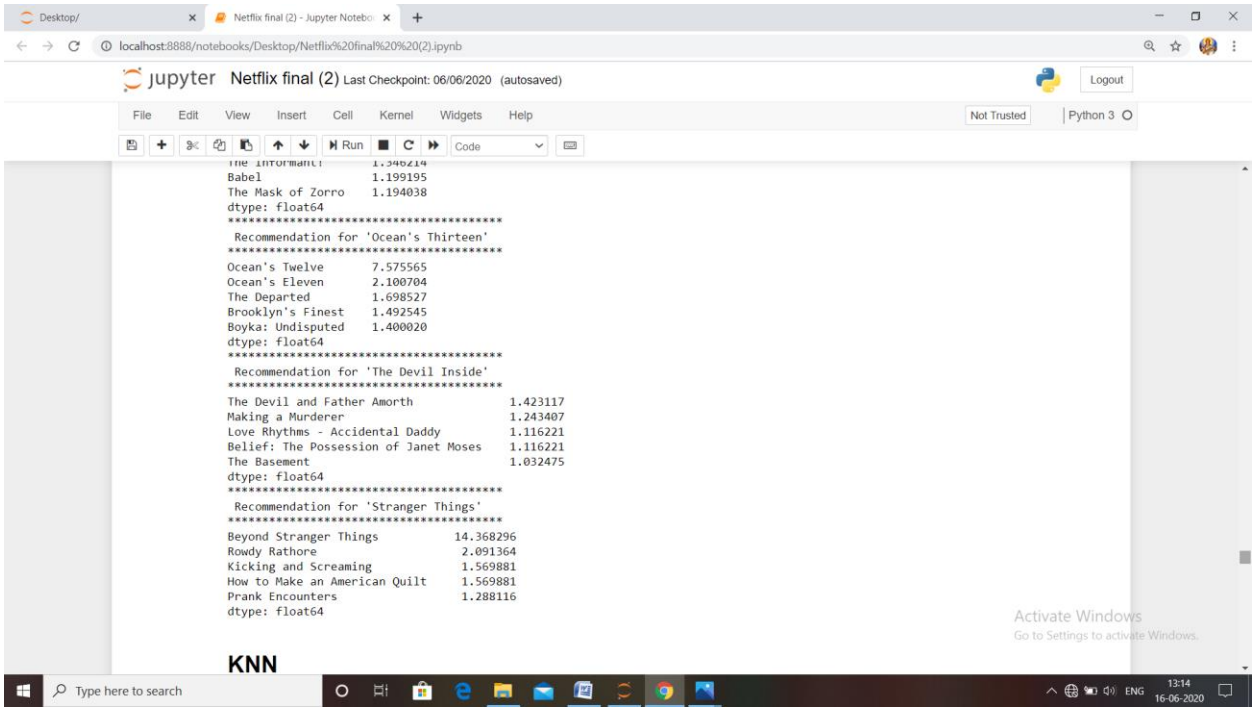
Recommendation function

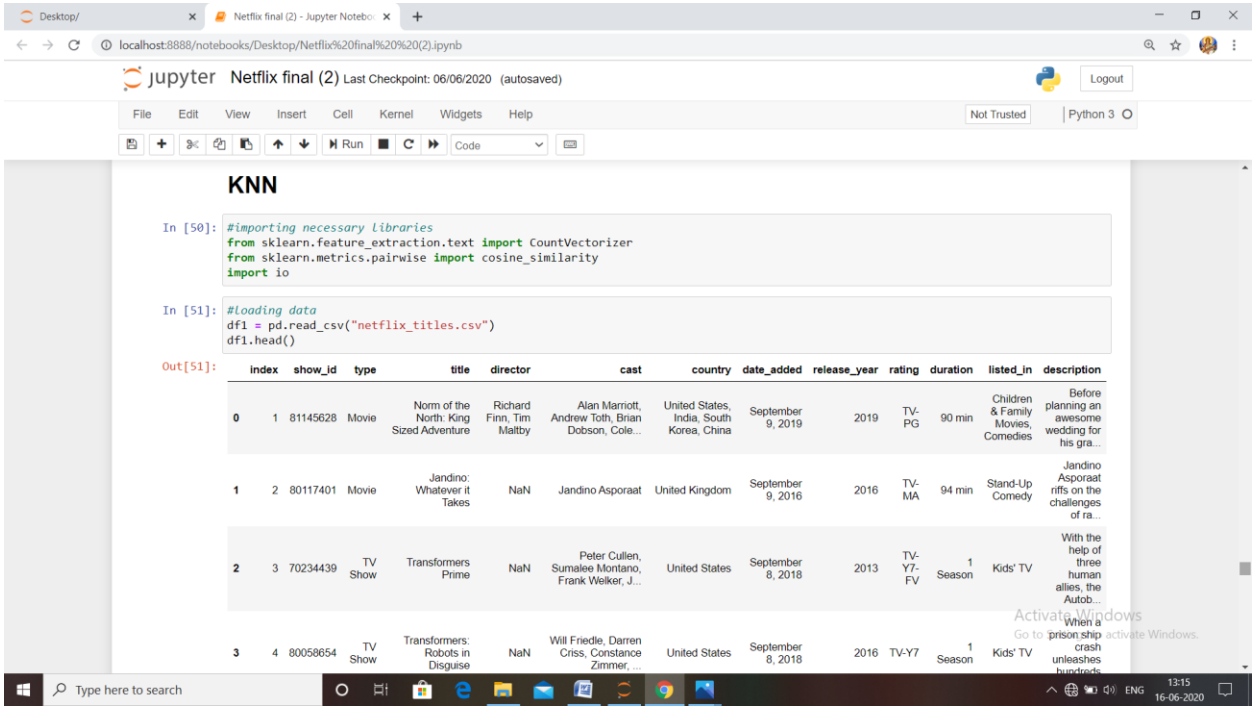
In [47]:
```
#defining a function to get recommendations
#Explore the neighborhood of the target film → this is a list of actor, director, country, categorie
#Explore the neighborhood of each neighbor → discover the movies that share a node with the target field
#Calculate Adamic Adar measure → final results
def get_recommendation(root):
    commons_dict = {}
    for e in G.neighbors(root):
        for e2 in G.neighbors(e):
            if e2==root:
                continue
            if G.nodes[e2]['label']=="MOVIE":
                commons = commons_dict.get(e2)
                if commons==None:
                    commons_dict.update({e2 : [e]})
                else:
                    commons.append(e)
                    commons_dict.update({e2 : commons})
    movies=[]
    weight=[]
    for key, values in commons_dict.items():
        w=0.0
        for e in values:
            w=w+1/math.log(G.degree(e))
        movies.append(key)
        weight.append(w)
    result = pd.Series(data=np.array(weight),index=movies)
    result.sort_values(inplace=True,ascending=False)
    return result;
```

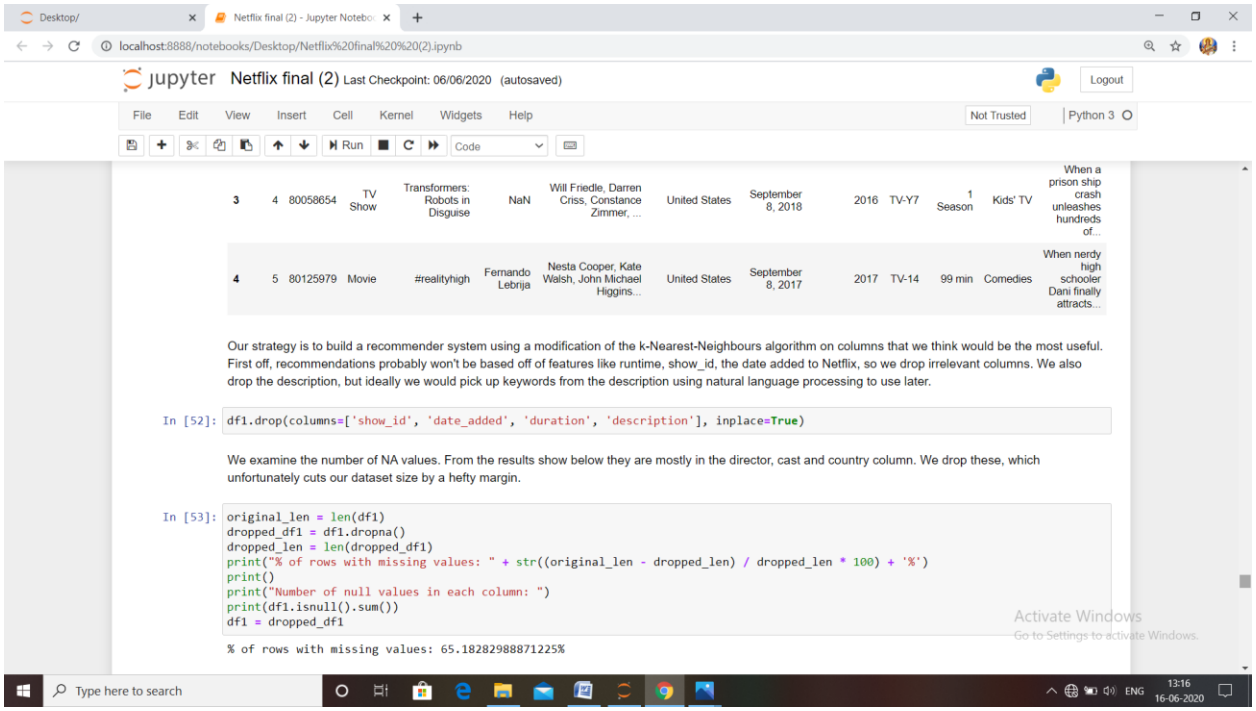
Activate Windows
Go to Settings to activate Windows.

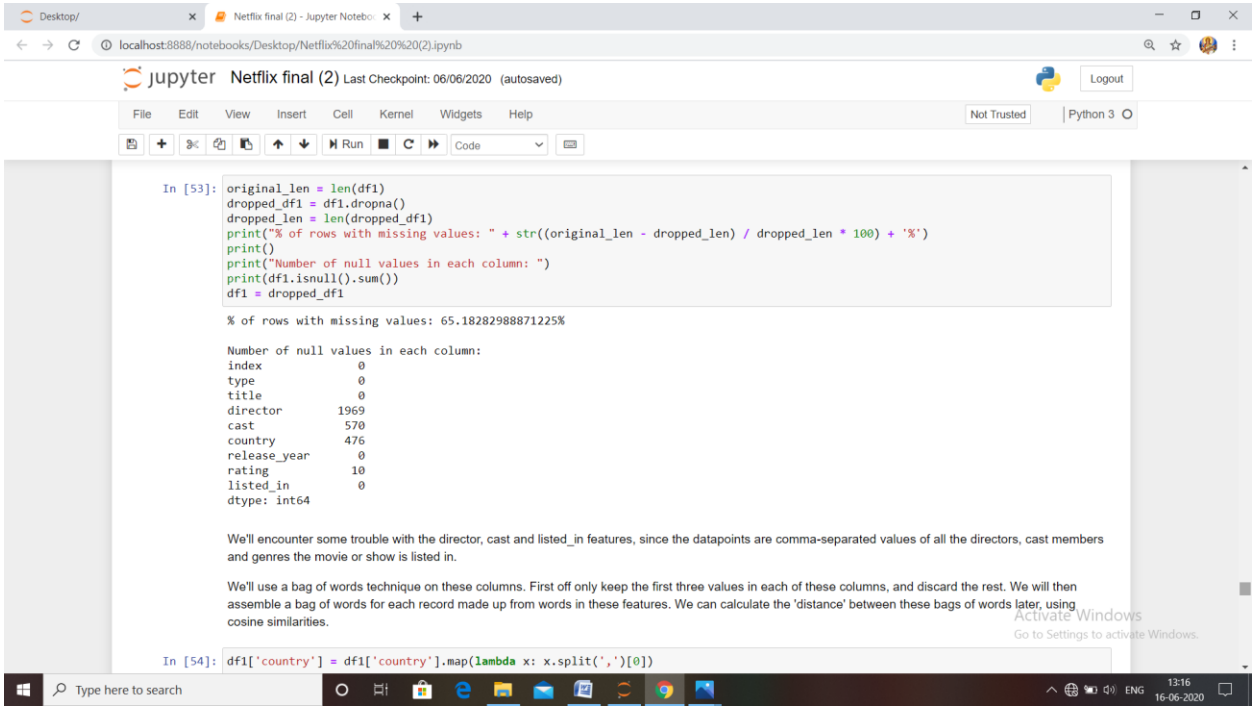
Type here to search | ENG 13:13 16-06-2020

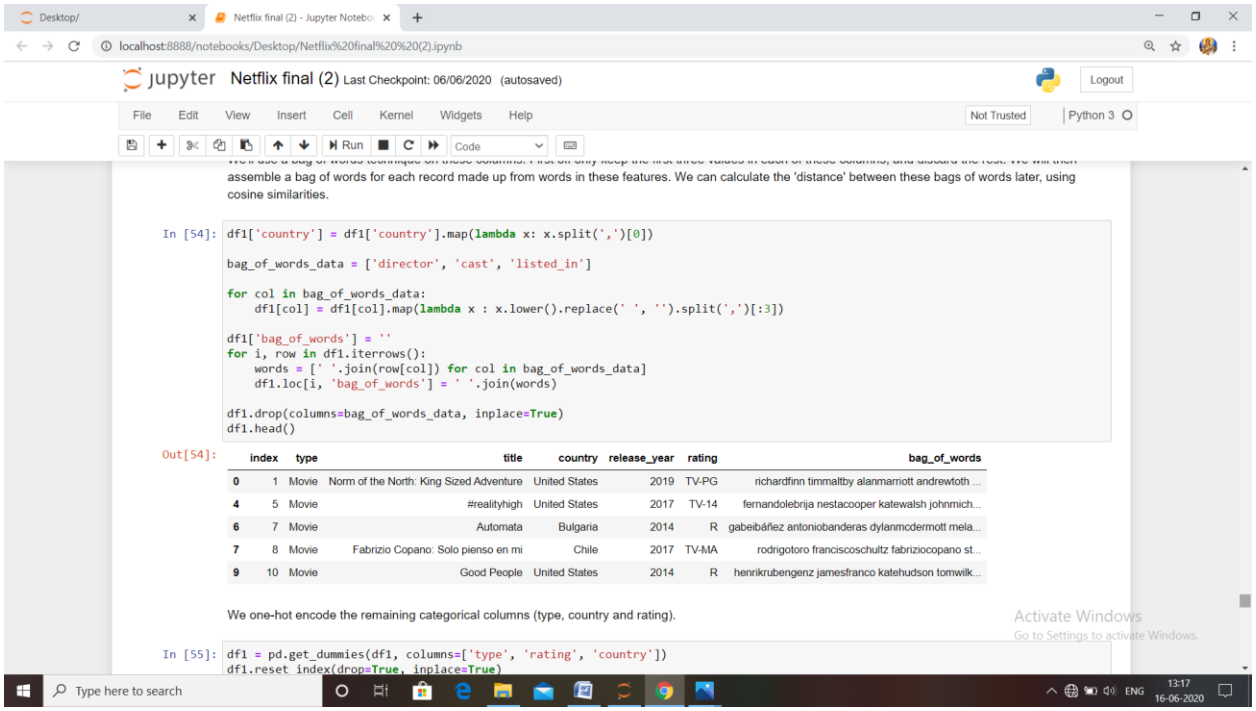

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
weight.append(w)
result = pd.Series(data=np.array(weight),index=movies)
result.sort_values(inplace=True,ascending=False)
return result;
Testing
In [48]:
result = get_recommendation("Ocean's Twelve")#predicting movies for oceans twelve
result2 = get_recommendation("Ocean's Thirteen")
result3 = get_recommendation("The Devil Inside")
result4 = get_recommendation("Stranger Things")
print("*"*40+"\n Recommendation for 'Ocean's Twelve'\n"+"*"*40)
print(result.head())
print("*"*40+"\n Recommendation for 'Ocean's Thirteen'\n"+"*"*40)
print(result2.head())
print("*"*40+"\n Recommendation for 'The Devil Inside'\n"+"*"*40)
print(result3.head())
print("*"*40+"\n Recommendation for 'Stranger Things'\n"+"*"*40)
print(result4.head())
****
Recommendation for 'Ocean's Twelve'
****
Ocean's Thirteen 7.575565
Ocean's Eleven 1.542593
The Informant! 1.346214
Babel 1.199195
The Mask of Zorro 1.194038
dtype: float64
****
Recommendation for 'Ocean's Thirteen'
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:14
16-06-2020

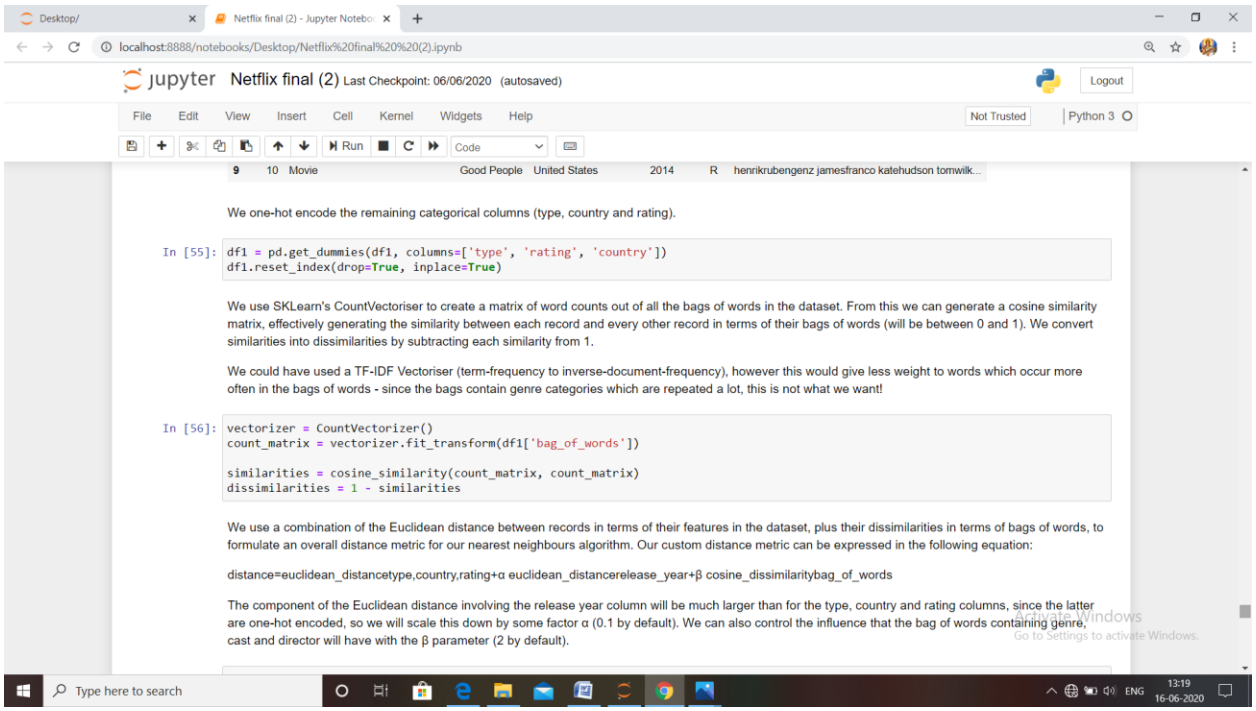

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
The Informant! 1.346214
Babel 1.199195
The Mask of Zorro 1.194038
dtype: float64
****
Recommendation for 'Ocean's Thirteen'
****
Ocean's Twelve 7.575565
Ocean's Eleven 2.100704
The Departed 1.698527
Brooklyn's Finest 1.492545
Boyka: Undisputed 1.400020
dtype: float64
****
Recommendation for 'The Devil Inside'
****
The Devil and Father Amorth 1.423117
Making a Murderer 1.243407
Love Rhythms - Accidental Daddy 1.116221
Belief: The Possession of Janet Moses 1.116221
The Basement 1.032475
dtype: float64
****
Recommendation for 'Stranger Things'
****
Beyond Stranger Things 14.368296
Rowdy Rathore 2.091364
Kicking and Screaming 1.569881
How to Make an American Quilt 1.569881
Prank Encounters 1.288116
dtype: float64
Activate Windows
Go to Settings to activate Windows.
KNN
Type here to search
ENG
13:14
16-06-2020

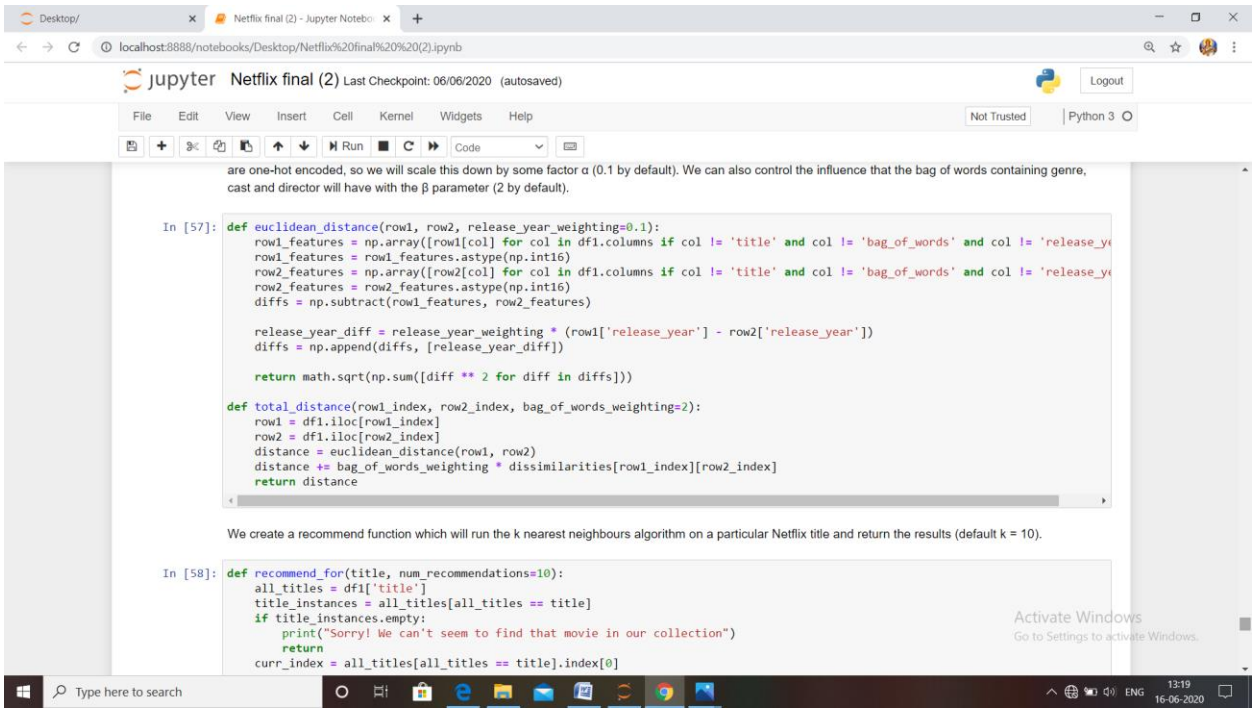

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
KNN
In [50]:
#importing necessary libraries
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import io
In [51]:
#loading data
df1 = pd.read_csv("netflix_titles.csv")
df1.head()
Out[51]:
index show_id type title director cast country date_added release_year rating duration listed_in description
0 1 81145628 Movie Norm of the North: King Sized Adventure Richard Finn, Tim Maltby Alan Marriott, Andrew Toth, Brian Dobson, Cole... United States, India, South Korea, China September 9, 2019 2019 TV-PG 90 min Children & Family Movies, Comedies Before planning an awesome wedding for his gra...
1 2 80117401 Movie Jandino: Whatever it Takes NaN Jandino Asporaat United Kingdom September 9, 2016 2016 TV-MA 94 min Stand-Up Comedy Jandino Asporaat riffs on the challenges of ra...
2 3 70234439 TV Show Transformers Prime NaN Peter Cullen, Sumalee Montano, Frank Welker, J... United States September 8, 2018 2013 TV-Y7-FV 1 Season Kids' TV With the help of three human allies, the Autob...
3 4 80058654 TV Show Transformers: Robots in Disguise NaN Will Friedle, Darren Criss, Constance Zimmer, ... United States September 8, 2018 2016 TV-Y7 1 Season Kids' TV When a prison ship crash unleashes hundreds
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:15
16-06-2020

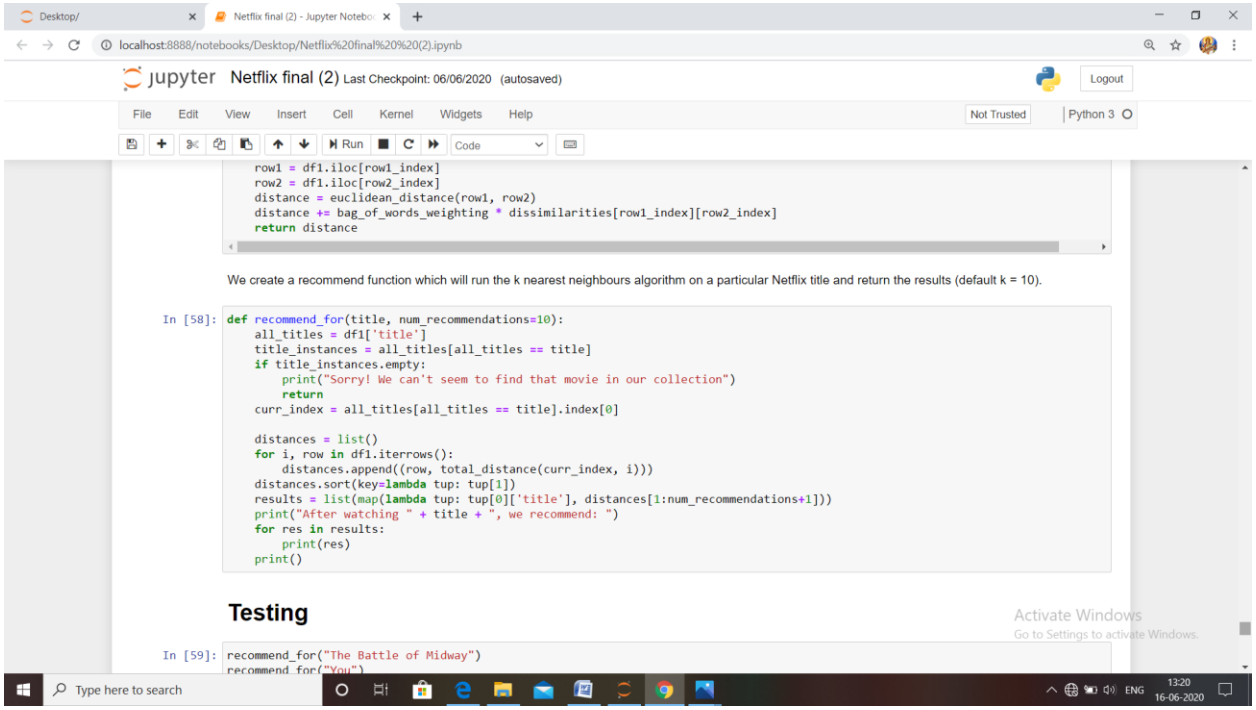

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
3 4 80058654 TV Show Transformers: Robots in Disguise NaN Will Friedle, Darren Criss, Constance Zimmer, ... United States September 8, 2018 2016 TV-Y7 1 Season Kids' TV When a prison ship crash unleashes hundreds of...
4 5 80125979 Movie #realityhigh Fernando Lebrija Nesta Cooper, Kate Walsh, John Michael Higgins... United States September 8, 2017 2017 TV-14 99 min Comedies When nerdy high schooler Dani finally attracts...
Our strategy is to build a recommender system using a modification of the k-Nearest-Neighbours algorithm on columns that we think would be the most useful. First off, recommendations probably won't be based off of features like runtime, show_id, the date added to Netflix, so we drop irrelevant columns. We also drop the description, but ideally we would pick up keywords from the description using natural language processing to use later.
In [52]:
df1.drop(columns=['show_id', 'date_added', 'duration', 'description'], inplace=True)
We examine the number of NA values. From the results show below they are mostly in the director, cast and country column. We drop these, which unfortunately cuts our dataset size by a hefty margin.
In [53]:
original_len = len(df1)
dropped_df1 = df1.dropna()
dropped_len = len(dropped_df1)
print("% of rows with missing values: " + str((original_len - dropped_len) / dropped_len * 100) + '%')
print()
print("Number of null values in each column: ")
print(df1.isnull().sum())
df1 = dropped_df1
% of rows with missing values: 65.18282988871225%
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:16
16-06-2020

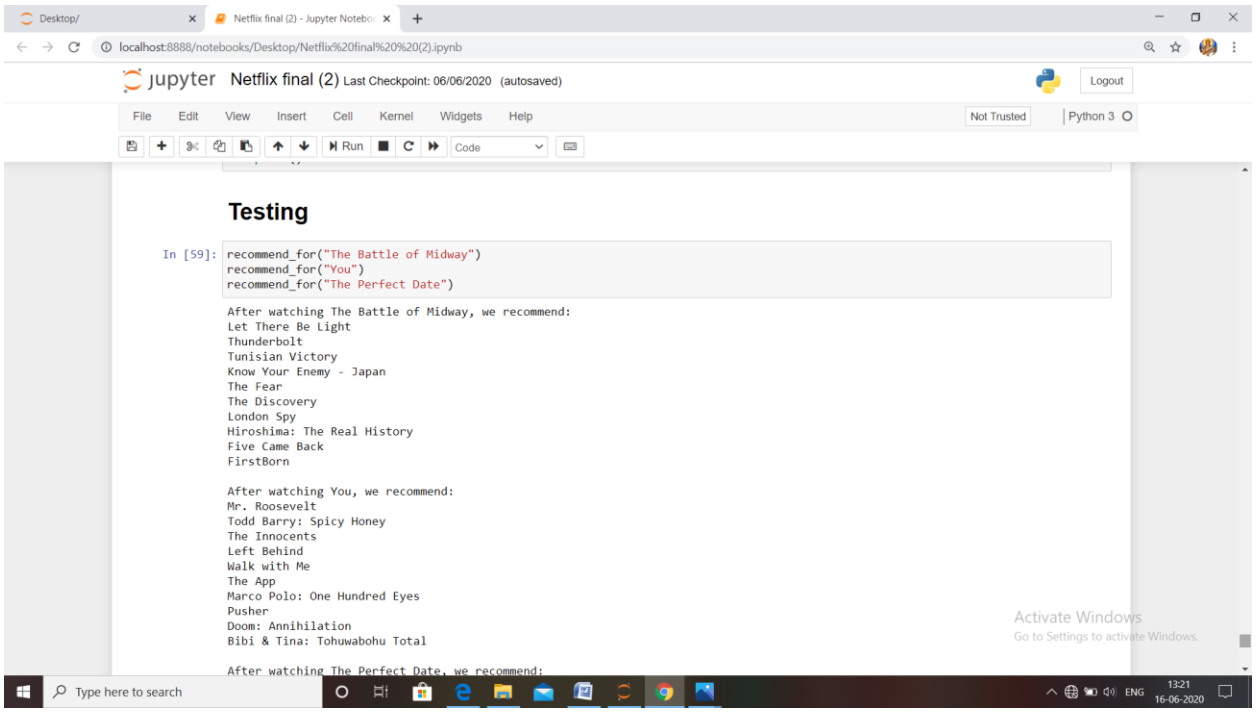

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [53]:
original_len = len(df1)
dropped_df1 = df1.dropna()
dropped_len = len(dropped_df1)
print("% of rows with missing values: " + str((original_len - dropped_len) / dropped_len * 100) + '%')
print()
print("Number of null values in each column: ")
print(df1.isnull().sum())
df1 = dropped_df1
% of rows with missing values: 65.18282988871225%
Number of null values in each column:
index 0
type 0
title 0
director 1969
cast 570
country 476
release_year 0
rating 10
listed_in 0
dtype: int64
We'll encounter some trouble with the director, cast and listed_in features, since the datapoints are comma-separated values of all the directors, cast members and genres the movie or show is listed in.
We'll use a bag of words technique on these columns. First off only keep the first three values in each of these columns, and discard the rest. We will then assemble a bag of words for each record made up from words in these features. We can calculate the 'distance' between these bags of words later, using cosine similarities.
Activate Windows
Go to Settings to activate Windows.
In [54]:
df1['country'] = df1['country'].map(lambda x: x.split(',')[0])
Type here to search
ENG
13:16
16-06-2020

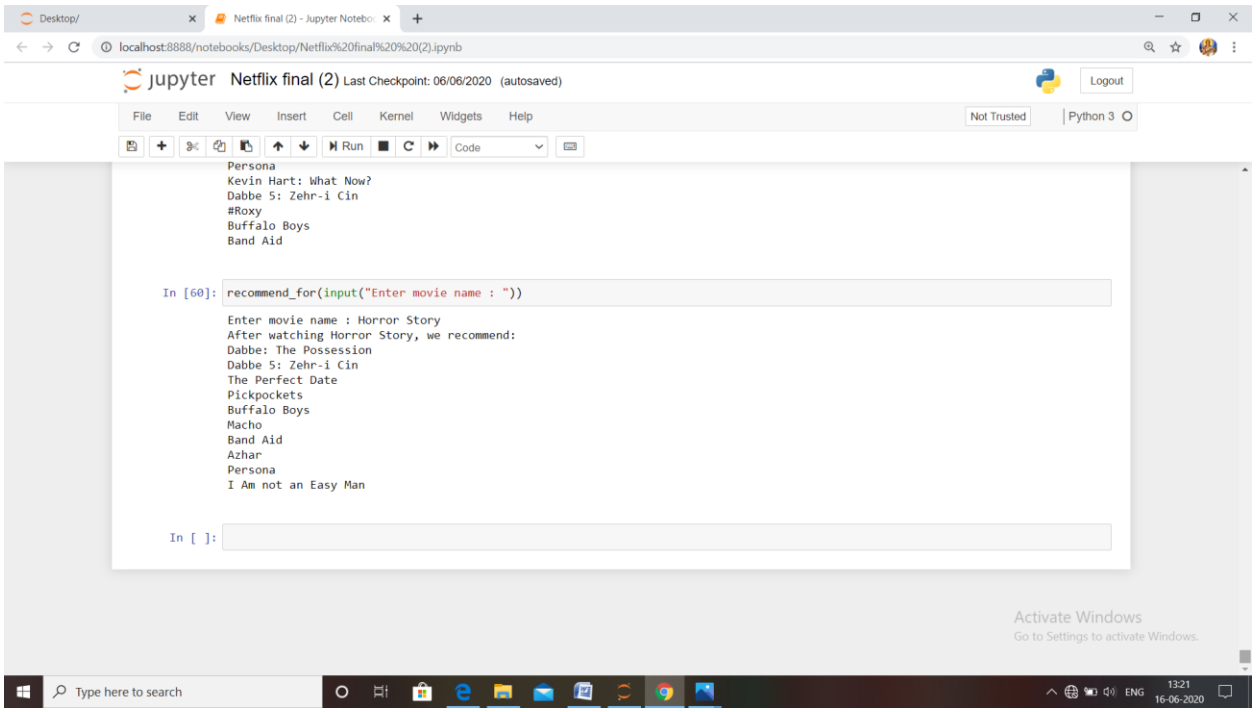

Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
assemble a bag of words for each record made up from words in these features. We can calculate the 'distance' between these bags of words later, using cosine similarities.
In [54]:
df1['country'] = df1['country'].map(lambda x: x.split(',')[0])
bag_of_words_data = ['director', 'cast', 'listed_in']
for col in bag_of_words_data:
df1[col] = df1[col].map(lambda x : x.lower().replace(' ', '').split(',')[:3])
df1['bag_of_words'] = ''
for i, row in df1.iterrows():
words = [' '.join(row[col]) for col in bag_of_words_data]
df1.loc[i, 'bag_of_words'] = ' '.join(words)
df1.drop(columns=bag_of_words_data, inplace=True)
df1.head()
Out[54]:
index type title country release_year rating bag_of_words
0 1 Movie Norm of the North: King Sized Adventure United States 2019 TV-PG richardfinn timmaltby alanmarriott andrewtoth ...
4 5 Movie #realityhigh United States 2017 TV-14 fernandolebrija nestacooper katewalsh johnmich...
6 7 Movie Automata Bulgaria 2014 R gabeibáñez antoniobanderas dylanmcdermott mela...
7 8 Movie Fabrizio Copano: Solo pienso en mi Chile 2017 TV-MA rodrigotoro franciscoschultz fabriziocopano st...
9 10 Movie Good People United States 2014 R henrikrubengenz jamesfranco katehudson tomwilk...
We one-hot encode the remaining categorical columns (type, country and rating).
Activate Windows
Go to Settings to activate Windows.
In [55]:
df1 = pd.get_dummies(df1, columns=['type', 'rating', 'country'])
df1.reset_index(drop=True, inplace=True)
Type here to search
ENG
13:17
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
9 10 Movie Good People United States 2014 R henrikrubengenz jamesfranco katehudson tomwilk...
We one-hot encode the remaining categorical columns (type, country and rating).
In [55]:
df1 = pd.get_dummies(df1, columns=['type', 'rating', 'country'])
df1.reset_index(drop=True, inplace=True)
We use SKLearn's CountVectoriser to create a matrix of word counts out of all the bags of words in the dataset. From this we can generate a cosine similarity matrix, effectively generating the similarity between each record and every other record in terms of their bags of words (will be between 0 and 1). We convert similarities into dissimilarities by subtracting each similarity from 1.
We could have used a TF-IDF Vectoriser (term-frequency to inverse-document-frequency), however this would give less weight to words which occur more often in the bags of words - since the bags contain genre categories which are repeated a lot, this is not what we want!
In [56]:
vectorizer = CountVectorizer()
count_matrix = vectorizer.fit_transform(df1['bag_of_words'])
similarities = cosine_similarity(count_matrix, count_matrix)
dissimilarities = 1 - similarities
We use a combination of the Euclidean distance between records in terms of their features in the dataset, plus their dissimilarities in terms of bags of words, to formulate an overall distance metric for our nearest neighbours algorithm. Our custom distance metric can be expressed in the following equation:
distance=euclidean_distancetype,country,rating+α euclidean_distancerelease_year+β cosine_dissimilaritybag_of_words
The component of the Euclidean distance involving the release year column will be much larger than for the type, country and rating columns, since the latter are one-hot encoded, so we will scale this down by some factor α (0.1 by default). We can also control the influence that the bag of words containing genre, cast and director will have with the β parameter (2 by default).
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:19
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
are one-hot encoded, so we will scale this down by some factor α (0.1 by default). We can also control the influence that the bag of words containing genre, cast and director will have with the β parameter (2 by default).
In [57]:
def euclidean_distance(row1, row2, release_year_weighting=0.1):
row1_features = np.array([row1[col] for col in df1.columns if col != 'title' and col != 'bag_of_words' and col != 'release_y
row1_features = row1_features.astype(np.int16)
row2_features = np.array([row2[col] for col in df1.columns if col != 'title' and col != 'bag_of_words' and col != 'release_y
row2_features = row2_features.astype(np.int16)
diffs = np.subtract(row1_features, row2_features)
release_year_diff = release_year_weighting * (row1['release_year'] - row2['release_year'])
diffs = np.append(diffs, [release_year_diff])
return math.sqrt(np.sum([diff ** 2 for diff in diffs]))
def total_distance(row1_index, row2_index, bag_of_words_weighting=2):
row1 = df1.iloc[row1_index]
row2 = df1.iloc[row2_index]
distance = euclidean_distance(row1, row2)
distance += bag_of_words_weighting * dissimilarities[row1_index][row2_index]
return distance
We create a recommend function which will run the k nearest neighbours algorithm on a particular Netflix title and return the results (default k = 10).
In [58]:
def recommend_for(title, num_recommendations=10):
all_titles = df1['title']
title_instances = all_titles[all_titles == title]
if title_instances.empty:
print("Sorry! We can't seem to find that movie in our collection")
return
curr_index = all_titles[all_titles == title].index[0]
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:19
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
row1 = df1.iloc[row1_index]
row2 = df1.iloc[row2_index]
distance = euclidean_distance(row1, row2)
distance += bag_of_words_weighting * dissimilarities[row1_index][row2_index]
return distance
We create a recommend function which will run the k nearest neighbours algorithm on a particular Netflix title and return the results (default k = 10).
In [58]:
def recommend_for(title, num_recommendations=10):
all_titles = df1['title']
title_instances = all_titles[all_titles == title]
if title_instances.empty:
print("Sorry! We can't seem to find that movie in our collection")
return
curr_index = all_titles[all_titles == title].index[0]
distances = list()
for i, row in df1.iterrows():
distances.append((row, total_distance(curr_index, i)))
distances.sort(key=lambda tup: tup[1])
results = list(map(lambda tup: tup[0]['title'], distances[1:num_recommendations+1]))
print("After watching " + title + ", we recommend: ")
for res in results:
print(res)
print()
Testing
In [59]:
recommend_for("The Battle of Midway")
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:20
16-06-2020


Desktop/
Netflix final (2) - Jupyter Notebo
localhost:8888/notebooks/Desktop/Netflix%20final%20%20(2).ipynb
jupyter Netflix final (2) Last Checkpoint: 06/06/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
Testing
In [59]:
recommend_for("The Battle of Midway")
recommend_for("You")
recommend_for("The Perfect Date")
After watching The Battle of Midway, we recommend:
Let There Be Light
Thunderbolt
Tunisian Victory
Know Your Enemy - Japan
The Fear
The Discovery
London Spy
Hiroshima: The Real History
Five Came Back
FirstBorn
After watching You, we recommend:
Mr. Roosevelt
Todd Barry: Spicy Honey
The Innocents
Left Behind
Walk with Me
The App
Marco Polo: One Hundred Eyes
Pusher
Doom: Annihilation
Bibi & Tina: Tohuwabohu Total
After watching The Perfect Date, we recommend:
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
13:21
16-06-2020



4.2 Data Modelling:

4.2.1 Architecture:

We used Jupyter notebook for the execution of the whole project. 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.

4.2.2 Attribute Study- Data and Features

Data and Features:

- The project is easily usable.
- It is efficient.

- The user can find new different movies from the genres that he has watched before.

5.Results

The result of the experiment after testing is that, it recommends the user with different movies depending on the input the user has given. It recommends the user with five other different movies along with the probability of the user wanting to watch that movie.

6.Future Work

Using machine learning pervasively across Netflix brings many new challenges where we need to push forward the state-of-the-art. This means coming up with new ideas and testing them out, be it new models and algorithms or improvements to existing ones, better metrics or evaluation methodologies, and addressing the challenges of scale. To this experiment, we can add up extra features like giving all the movies that a particular director has directed when given the director name as the input and the same can be done with actors too. Another implementation that can be one is all the movies of a particular genre to be appeared when genre is given as the input.

7.Conclusion

Under the condition of massive information, the requirements of movie recommendation system from film amateur are increasing. This article designs and implements a complete movie recommendation system

prototype based on K - Means algorithm, Cosine Algorithm and tfidf Vectorizer.