MOVIE RECOMMENDATION SYSTEM

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the guidance of

MR. MAHENDRA DATTA

BY

OISHI BANERJEE

DEBONITA CHATTERJEE

ISHITA MUKHERJEE

NARULA INSTITUTE OF TECHNOLOGY

KOLKATA, WEST-BENGAL, INDIA

In association with

ARDENT COMPUTECH PVT. LTD.

High-End Technology Training and Project

1. TITLE OF THE PROJECT:
MOVIE RECOMMENDATION SYSTEM

2. PROJECT MEMBERS: -
OISHI BANERJEE
DEBONITA CHATTERJEE
ISHITA MUKHERJEE

3. NAME OF THE GUIDE: - MAHENDRA DATTA.

4. PROJECT VERSION CONTROL: -

Version	Primary Author	escription Of Version	Date Completed
Final	OISHI BANERJEE DEBONITA CHATTERJEE ISHITA MUKHERJEE	Project Report	**03-09-2021**

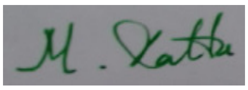
Signature of the Student

Date: 03-09-2021

Signature of the Supervisor

Date:

For Office Use Only



MAHENDRA DATTA

Project proposal / Evaluator

APPROVED

NOT APPROVED

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"MOVIE RECOMMENDATION SYSTEM"** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** at **ARDENT COMPUTECH PVT LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. MAHENDRA DATTA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 03-09-2021

1. **Name of the Student**: OISHI BANERJEE, DEBONITA CHATTERJEE, ISHITA MUKHERJEE

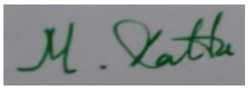
Signature of the student:

ARDENT
COMPUTECH PVT. LTD.
High-End Technology Training and Project

CERTIFICATE

This is to certify that this proposal of minor project entitled **“MOVIE RECOMMENDATION SYSTEM”** is a record of bonafide work, carried out **by OISHI BANERJEE, DEBONITA CHATTERJEE and ISHITA MUKHERJEE** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** and as per regulations of the **ARDENT COMPUTECH PRIVATE LIMITED.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor



MR. MAHENDRA DATTA

Project Engineer

ARDENT COMPUTECH PVT. LTD.

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. Mahendra Datta***, Project Engineer at ARDENT COMPUTECH PRIVATE LIMITED, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent computech pvt.ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

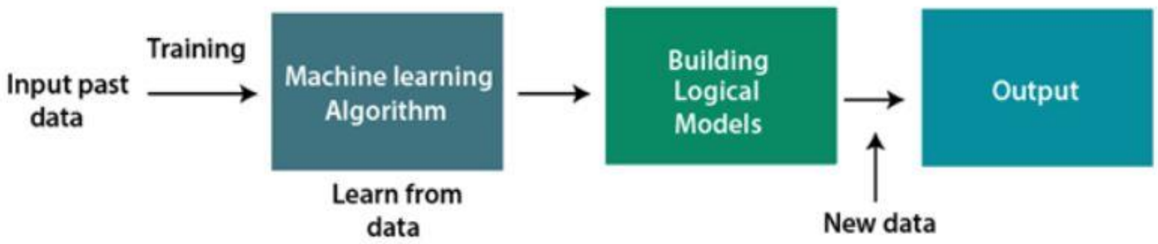
CONTENTS

ABSTRACT

Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. **The primary aim is to allow the computers learn automatically** without human intervention or assistance and adjust actions accordingly.



Classification of Machine Learning

1. Supervised Learning.
2. Unsupervised Learning.
3. Reinforcement Learning.

SUPERVISED LEARNING:

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

1. **Regression**
2. **Classification**

UNSUPERVISED LEARNING:

As the name suggests, unsupervised learning is a machine learning technique in which models are not supervised using training dataset. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things.

1. **Clustering**

2. **Association**

REINFORCEMENT LEARNING

Reinforcement learning (RL) is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.

LIST OF COMMON MACHINE LEARNING ALGORITHMS:

Here is the list of commonly used machine learning algorithms. These algorithms can be applied to almost any data problem:

1. Linear Regression
2. Logistic Regression
3. Decision Tree
4. SVM
5. Naive Bayes
6. KNN (K-Nearest Neighbors)
7. K-Means
8. Random Forest
9. Dimensionality Reduction Algorithms.

1. Linear Regression

It is used to estimate real values (cost of houses, number of calls, total sales etc.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation Y= a *X + b.

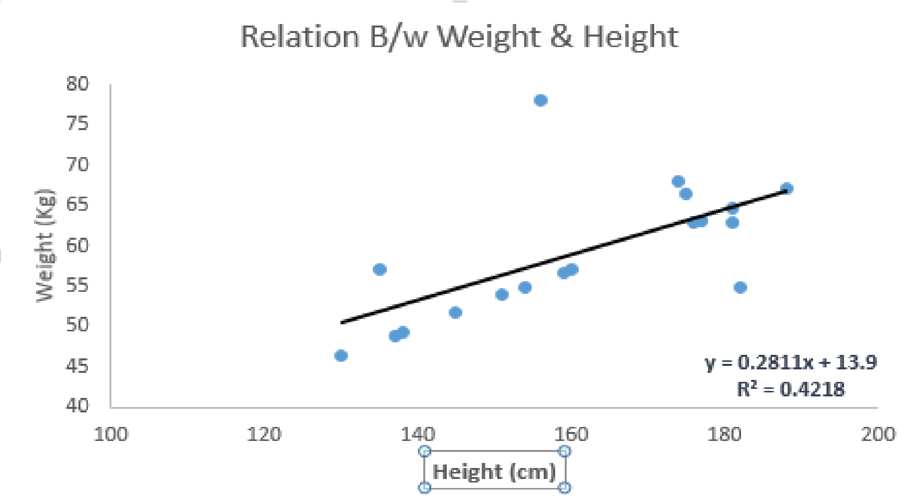
The best way to understand linear regression is to relive this experience of childhood. Let us say, you ask a child in fifth grade to arrange people in his class by increasing order of weight, without asking them their weights! What do you think the child will do? He / she would likely look (visually analyze) at the height and build of people and arrange them using a combination of these visible parameters. This is linear regression in real life! The child has actually figured out that height and build would be correlated to the weight by a relationship, which looks like the equation above.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

These coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line.

Look at the below example. Here we have identified the best fit line having linear equation **y=0.2811x+13.9**. Now using this equation, we can find the weight, knowing the height of a person.



Linear Regression is mainly of two types: Simple Linear Regression and Multiple Linear Regression. Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression (as the name suggests) is characterized by multiple (more than 1) independent variables. While finding the best fit line, you can fit a polynomial or curvilinear regression. And these are known as polynomial or curvilinear regression.

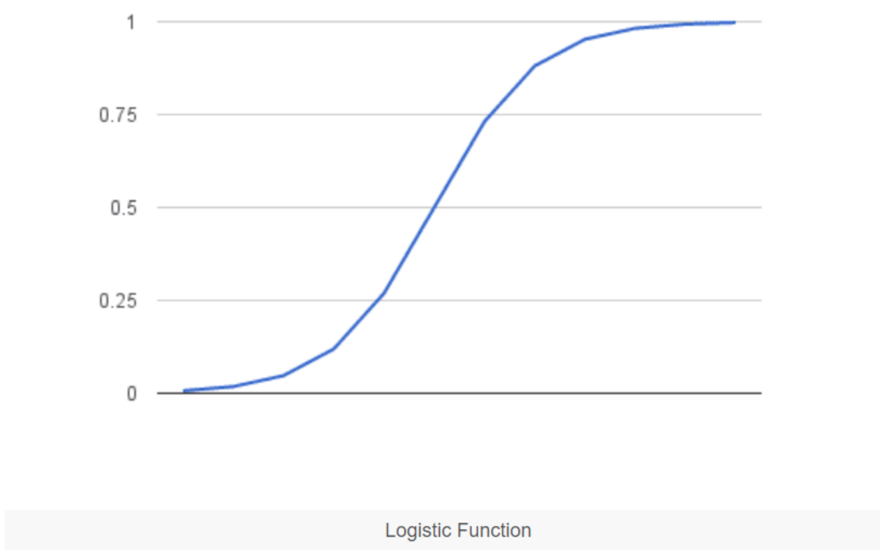
2. Logistic Regression

It is a not a regression algorithm. It is used to estimate discrete values (Binary values like 0/1, yes/no, true/false) based on given set of independent variables. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function. Hence, it is also known as **logit regression**. Since, it predicts the probability, its output values lie between 0 and 1 (as expected).

Logistic regression is named for the function used at the core of the method, the logistic function.

1 / (1 + e^-value)

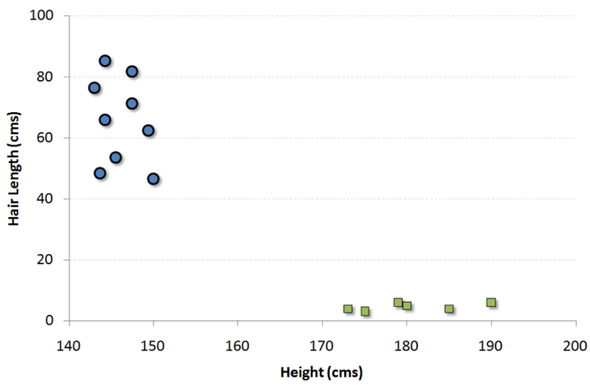
Where e is the base of the natural logarithms (Euler's number) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.



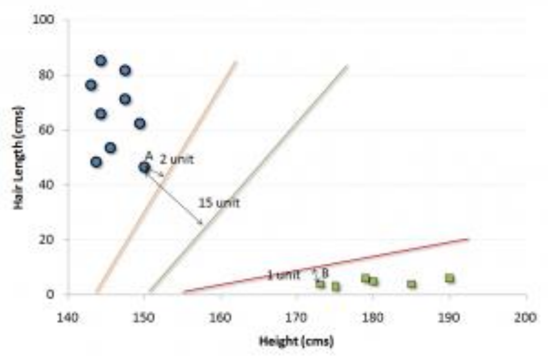
3. SVM (Support Vector Machine)

It is a classification method. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

For example, if we only had two features like Height and Hair length of an individual, we'd first plot these two variables in two-dimensional space where each point has two co-ordinates (these co-ordinates are known as **Support Vectors**)



Now, we will find some *line* that splits the data between the two differently classified groups of data. This will be the line such that the distances from the closest point in each of the two groups will be farthest away.

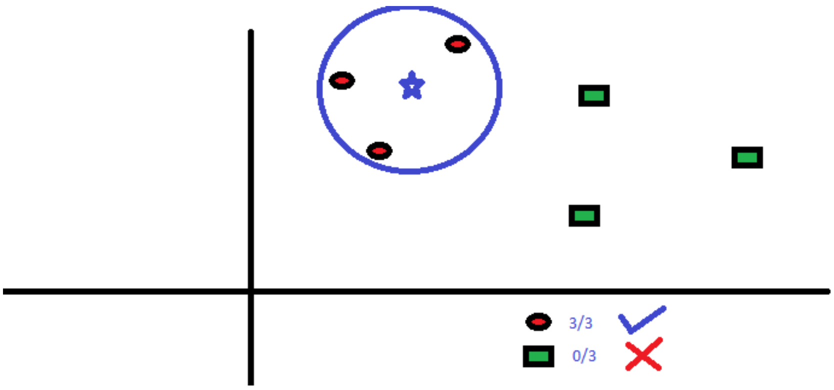


In the example shown above, the line which splits the data into two differently classified groups is the *black* line, since the two closest points are the farthest apart from the line. This line is our classifier. Then, depending on where the testing data lands on either side of the line, that's what class we can classify the new data as.

4. kNN (k- Nearest Neighbors)

It can be used for both classification and regression problems. However, it is more widely used in classification problems in the industry. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases by a majority vote of its k neighbors. The case being assigned to the class is most common amongst its K nearest neighbors measured by a distance function.

These distance functions can be Euclidean, Manhattan, Minkowski and Hamming distance. First three functions are used for continuous function and fourth one (Hamming) for categorical variables. If K = 1, then the case is simply assigned to the class of its nearest neighbor. At times, choosing K turns out to be a challenge while performing kNN modelling.



KNN can easily be mapped to our real lives. If you want to learn about a person, of whom you have no information, you might like to find out about his close friends and the circles he moves in and gain access to his/her information!

Things to consider before selecting kNN:

- KNN is computationally expensive
- Variables should be normalized else higher range variables can bias it
- Works on pre-processing stage more before going for kNN like an outlier, noise removal

5. Random Forest

Random Forest is a trademark term for an ensemble of decision trees. In Random Forest, we've collection of decision trees (so known as "Forest"). To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

Each tree is planted & grown as follows:

1. If the number of cases in the training set is N, then sample of N cases is taken at random but *with replacement*. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

PROBLEM STATEMENT

Create a Movie Recommendation System using ML algorithms and Python.

INTRODUCTION

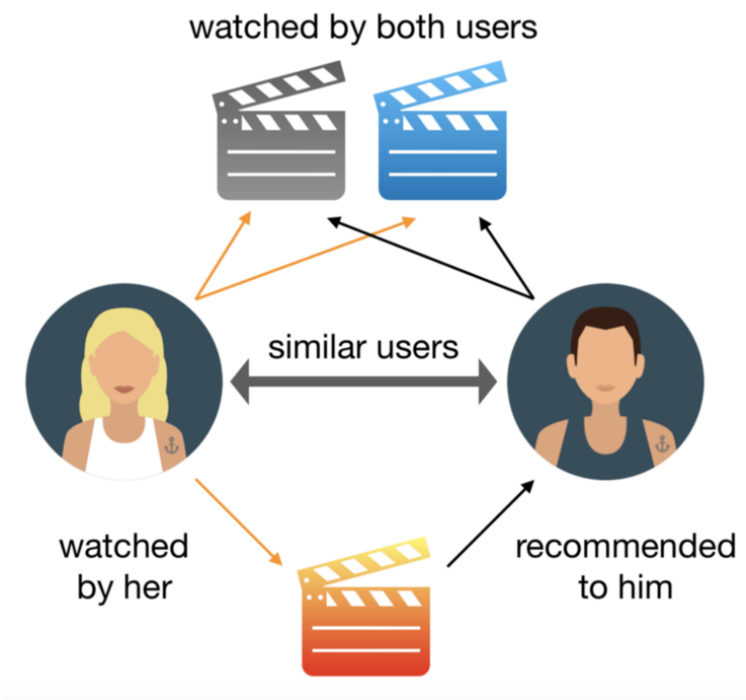
"Every time I go to a movie, it's magic, no matter what the movie's about."
– **Steven Spielberg**

Everyone loves movies irrespective of age, gender, race, color, or geographical location. We all in a way are connected to each other via this amazing medium. Yet what most interesting is the fact that how **unique** our choices and combinations are in terms of movie preferences. Some people like genre-specific movies be it a thriller, romance, or sci-fi, while others focus on lead actors and directors. When we take all that into account, it's astoundingly difficult to generalize a movie and say that everyone would like it. But with all that said, it is still seen that similar movies are liked by a specific part of the society.
So here's where we as data scientists come into play and extract the juice out of all the **behavioral patterns** of not only the audience but also from the movies themselves.

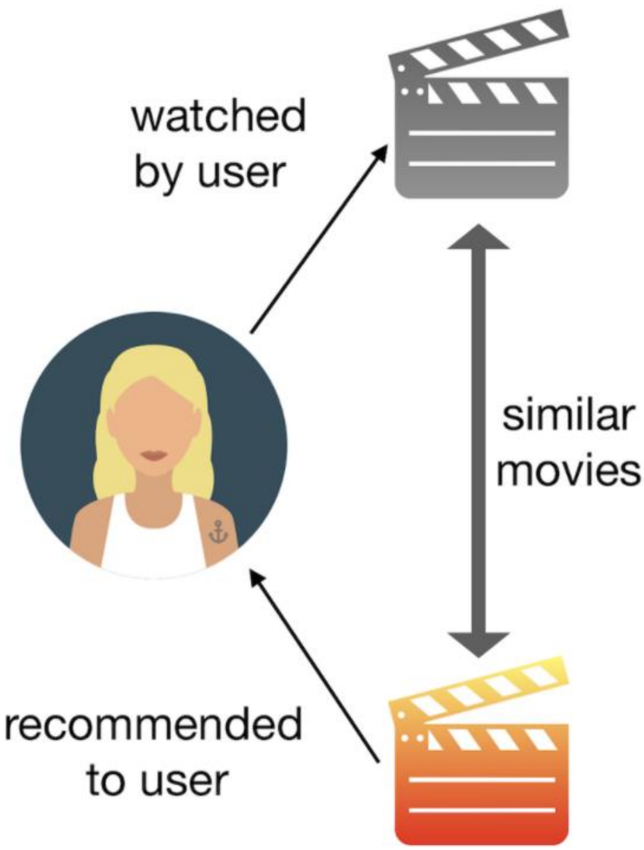
What is Movie Recommendation System?

A recommendation system is a system that provides suggestions to users for certain resources like books, movies, songs, etc., based on some data set. Movie recommendation systems usually predict what movies a user will like based on the attributes present in previously liked movies. Such recommendation systems are beneficial for organizations that collect data from large amounts of customers, and wish to effectively provide the best suggestions possible. A lot of factors can be considered while designing a movie recommendation system like the genre of the movie, actors present in it or even the director of the movie. The systems can recommend movies based on one or a combination of two or more attributes.

The two main types of recommender systems are either collaborative or content-based filters



Collaborative-based filter.



Content-based filter.

DETAILS OF THE PROJECT

1. Data Collection.

- The dataset required for this project was taken from *“kaggle.com”*.
- The dataset name is “netflix_titles.csv”
- This dataset consists of tv shows and movies available on Netflix as of 2019.

2. About the Data.

The dataset consists of 7787 Rows and 12 Columns.
The description of each Columns is as follows: -

1. Show_id: The unique ID of the show
2. Type: whether it is TV show or a Movie
3. Title
4. Director: Name of the director
5. Cast: The name of the actors
6. Country: The name of the country the movie is from.
7. Date_added: The date on which it was added to Netflix.
8. Release_year: The year the movie was released
9. Rating: The rating it has got
10. Duration: Either the number of seasons or time in hours.
11. Listed_in: In which genre it is listed.
12. Description: A brief description about the movie.

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

- **OS**: windows or linux
- **Python IDE**: python 3.1.x and above
- Jupyter notebook
- Setup tools and pin to be installed for 3.6 and above
- Language python

HARDWARE REQUIREMENTS

- **RAM**: 4GB and higher
- **Processor:** intel i3 and above
- **Hard disk**: 500GB minimum

MODULES USED IN THE PROJECT

- **NumPy: -** It is a general-purpose array processing package.it provides a high-performance multidimensional array object, and tools for working with these arrays. It's also an efficient multidimensional container of generic data.
- **Pandas:** - It is the most popular python library that is used for data analysis.it provides highly optimized performance with back-end source code is purely written in **Python.**
- **Matplotlib: -** It is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environment across platforms .it tries to make easy things easy and hard things possible. you can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code.
- **Seaborn: -** It is a python data visualization library based on matplotlib.it provides a high-level interface for drawing attractive and informative statistical graphics.it is closely integrated with Pandas data structures.

  Scikit-learn: - It is a free machine-learning library for python programming language.it features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means, DBSCAN and is designed to incorporate with the python numerical and scientific libraries NumPy and SciPy.

ACTUAL CODES WITH OUTPUT

Importing necessary libraries

1. Importing libraries

```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sb
        import matplotlib.pyplot as plt
        %matplotlib inline

        import warnings
        warnings.filterwarnings("ignore")
```

Explanation: -

We import the NumPy, pandas, matplotlib and seaborn (provides high-level interface for attractive and informative statistical graphics).

Importing dataset

Reading Data

```
In [2]: df = pd.read_csv("netflix_titles.csv")
```

Explanation: -

The csv file(dataset) is read using the "pd.read_csv('filename)" command.

Understanding the Data

```
df.head()
```

Out[2]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	TV Show	3%	NaN	João Miguel, Bianca Comparato, Michel Gomes, R...	Brazil	August 14, 2020	2020	TV-MA	4 Seasons	International TV Shows, TV Dramas, TV Sci-Fi &...	In a future where the elite inhabit an island ...
1	s2	Movie	7:19	Jorge Michel Grau	Demián Bichir, Héctor Bonilla, Oscar Serrano, ...	Mexico	December 23, 2016	2016	TV-MA	93 min	Dramas, International Movies	After a devastating earthquake hits Mexico Cit...
2	s3	Movie	23:59	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	R	78 min	Horror Movies, International Movies	When an army recruit is found dead, his fellow...
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PG-13	80 min	Action & Adventure, Independent Movies, Sci-Fi...	In a postapocalyptic world, rag-doll robots hi...

Explanation: -

head (), gives us a quick look at our dataset.

Data Exploration:

2. Data Exploration

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 12 columns):
show_id         7787 non-null object
type            7787 non-null object
title           7787 non-null object
director        5398 non-null object
cast            7069 non-null object
country         7280 non-null object
date_added      7777 non-null object
release_year    7787 non-null int64
rating          7780 non-null object
duration        7787 non-null object
listed_in       7787 non-null object
description     7787 non-null object
dtypes: int64(1), object(11)
memory usage: 730.2+ KB
```

Explanation: -

We will begin exploring the dataset to gain an understanding of the type data in our dataset. For this purpose, we will use pandas' built-in describe feature.

Finding Null Values

--> Checking for missing values

```
In [4]: df.isna().sum()

Out[4]: show_id            0
        type               0
        title              0
        director        2389
        cast             718
        country          507
        date_added        10
        release_year       0
        rating             7
        duration           0
        listed_in          0
        description        0
        dtype: int64
```

Explanation: -

- For finding the number of missing values in a dataset we use data.isnull().We call data.isnull().sum() to give the output of series containing data about count of NaN in each column.
- We get no null values…thus the data set has no missing values.

MISSING VALUES HANDLING:

--> Handling missing values

```
In [6]: df['country'] = df['country'].fillna(df['country'].mode()[0])
        df['date_added'] = df['date_added'].fillna(df['date_added'].mode()[0])
        df['rating'] = df['rating'].fillna(df['country'].mode()[0])

In [7]: df = df.dropna( how='any',subset=['cast', 'director'])

In [8]: df.isna().sum()

Out[8]: show_id         0
        type            0
        title           0
        director        0
        cast            0
        country         0
        date_added      0
        release_year    0
        rating          0
        duration        0
        listed_in       0
        description     0
        dtype: int64
```

- All the missing values in the dataset have either been removed or filled. There are no missing values left.

Explanation:

- There are missing values in column director,cast,country and date_added.

- We can't randomly fill the missing values in columns of director and cast, so we can drop them.
- For minimal number of missing values in country and date_added,rating, we can fill them using mode(most common value) and mean.

Cleaning the data:

```
In [10]: #Rename the 'listed_in' column as 'Genre' for easy understanding
         df = df.rename(columns={"listed_in":"Genre"})
         df['Genre'] = df['Genre'].apply(lambda x: x.split(",")[0])
         df['Genre'].head()

Out[10]: 1                  Dramas
         2           Horror Movies
         3      Action & Adventure
         4                  Dramas
         5    International TV Shows
         Name: Genre, dtype: object
```

```
In [11]: df['year_add'] = df['date_added'].apply(lambda x: x.split(" ")[-1])
         df['year_add'].head()

Out[11]: 1    2016
         2    2018
         3    2017
         4    2020
         5    2017
         Name: year_add, dtype: object
```

```
In [12]: df['month_add'] = df['date_added'].apply(lambda x: x.split(" ")[0])
         df['month_add'].head()

Out[12]: 1    December
         2    December
         3    November
         4     January
         5        July
         Name: month_add, dtype: object
```

```
In [13]: df['country_main'] = df['country'].apply(lambda x: x.split(",")[0])
         df['country_main'].head()

Out[13]: 1           Mexico
         2        Singapore
         3    United States
         4    United States
         5           Turkey
         Name: country_main, dtype: object
```

Explanation:

For our better understanding we simplified the data by adding some new columns like-

- listed_in - Genre
- Year Added - year_add
- Month Added - month_add
- Princial Country - country_main

SEPARATING THE DATAFRAME:

In [15]:
```
df['rating'].value_counts()
```

Out[15]:
```
TV-MA            1724
TV-14            1183
R                 656
TV-PG             426
PG-13             378
PG                241
TV-Y               90
TV-G               85
TV-Y7              82
NR                 62
G                  38
UR                  5
United States       4
TV-Y7-FV            3
NC-17               2
Name: rating, dtype: int64
```

-- Making two new dataframes, one with movies collection and other with TV shows collection:

- movie_df
- tv_df

In [16]:
```
movie_df = df[df['type'] == 'Movie']
tv_df = df[df['type'] == 'TV Show']
```

Explanation:

Making two new dataframes, one with movies collection and other with TV shows collection:

- movie_df
- tv_df

As these two categories has most values.

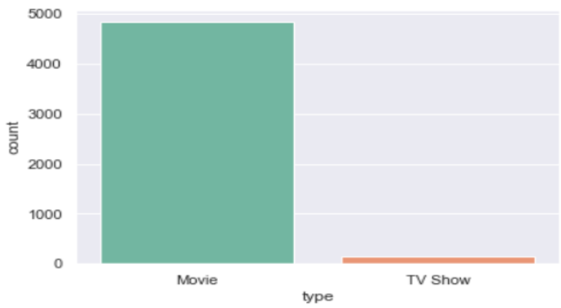
Exploratory Data Analysis

1. Number of movies vs Tv shows

--> Number of Movies vs TV Shows

In [18]:
```
sb.set(style="darkgrid")
sb.countplot(x="type", data= df, palette="Set2")
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1fdef954a48>



- There are more Movies on Netflix than TV shows.

Explanation:

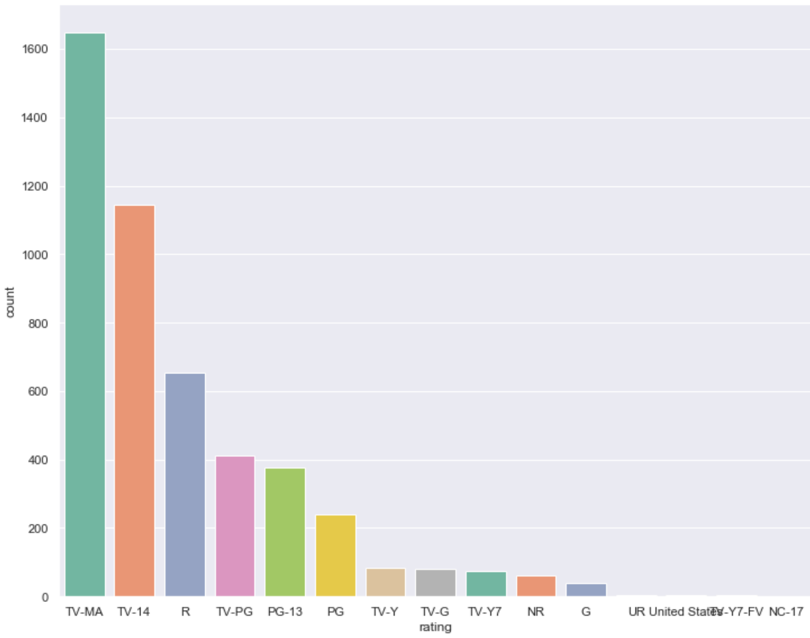
To do the comparison between tv shows and movies we use this countplot function and number of movies are much higher than tv shows.

2. Movies & TV Shows Ratings analysis

--> Movies & TV Shows Ratings analysis

In [19]:
```
#MOVIES RATINGS
plt.figure(figsize=(12,10))
sb.set(style="darkgrid")
sb.countplot(x="rating", data= movie_df, palette="Set2", order=movie_df['rating'].value_counts().index[0:15])
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1fdef950948>



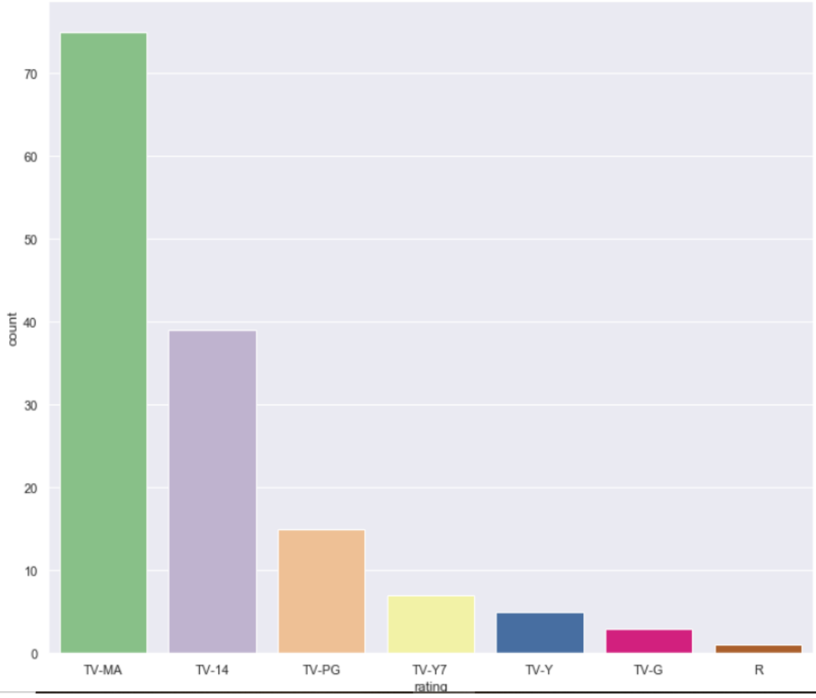
Explanation:

- The largest count of movies is made with the 'TV-MA' rating. “TV-MA" is a rating assigned by the TV Parental Guidelines to a television program that was designed for mature audiences only.
- Second largest is the 'TV-14' stands for content that may be inappropriate for children younger than 14 years of age.
- Third largest is the very popular 'R' rating. An R-rated film is a film that has been assessed as having material which may be unsuitable for children under the age of 17.

3. Tv shows rating:

```
In [20]: # TV SHOWS RATINGS
         plt.figure(figsize=(12,10))
         sb.set(style="darkgrid")
         sb.countplot(x="rating", data=tv_df, palette="Accent", order=tv_df['rating'].value_counts().index[0:15])

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x1fdef9b0b48>
```



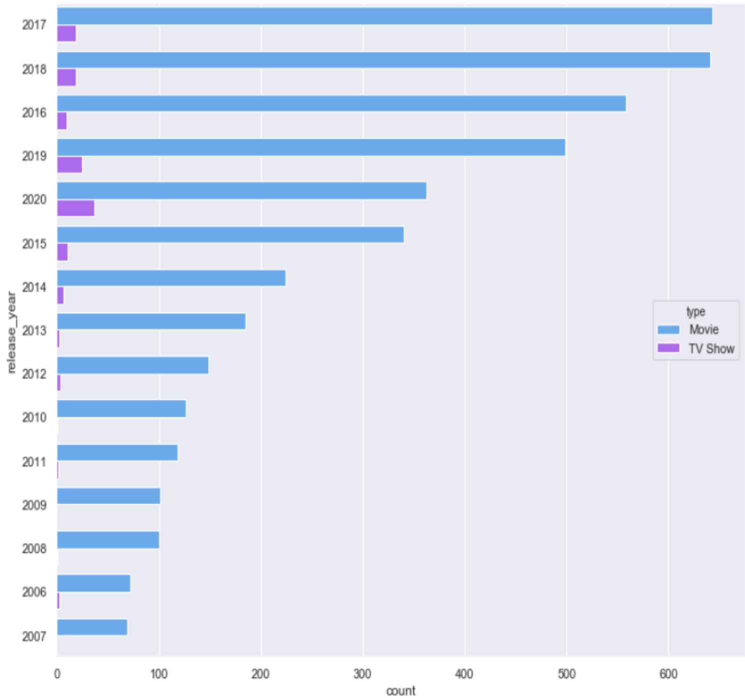
Explanation:

- Most of the TV Shows has 'TV-14' ratings which stands for the content can be inappropriate for children under 14 years of age.
- Second highest count of ratings in TV Shows is 'TV-MA', for which the content is for matured audience only.
- TV Shows has least number of counts with 'R' ratings.

4. Yearly Analysis of content

```
In [21]: plt.figure(figsize=(12,10))
         sb.set(style="darkgrid")
         sb.countplot(y="release_year", data= df, palette="cool", order= df['release_year'].value_counts().index[0:15],hue=df['type'])

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x1fdef950d08>
```



Explanation:

- We can see that Netflix released most number of content in year 2017.
- Noticeable growth in releasing content can be seen from the year 2015.

5. Analysis of movies duration

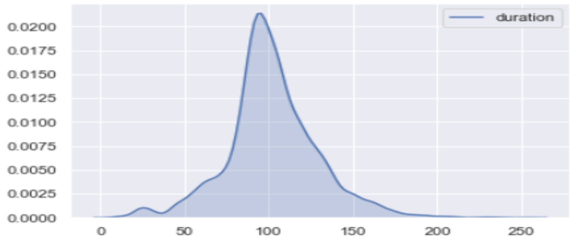
--> Analysis of movies duration

In [22]:
```
movie_df['duration']=movie_df['duration'].str.replace(' min','')
movie_df['duration']=movie_df['duration'].astype(str).astype(int)
movie_df['duration']
```

Out[22]:
```
1        93
2        78
3        80
4       123
6        95
       ...
7778     88
7780     94
7781     88
7782     99
7783    111
Name: duration, Length: 4834, dtype: int32
```

In [23]:
```
sb.set(style="darkgrid")
sb.kdeplot(data=movie_df['duration'], shade=True)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fdefeeef08>
```



Explanation:

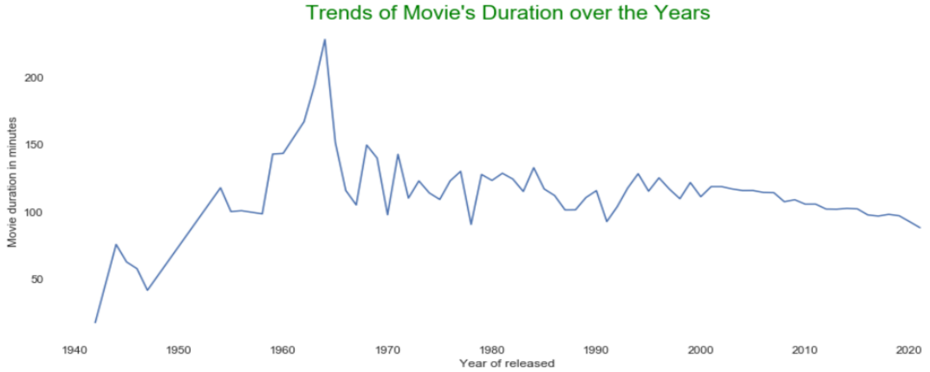
- So, a good number of movies on Netflix are among the **duration of 75-120 mins.**

6. Trends of movies duration

--> **Trends of movies duration**

```
In [24]: duration_year = movie_df.groupby(['release_year']).mean()
         duration_year = duration_year.sort_index()

         plt.figure(figsize=(15,6))
         sb.lineplot(x=duration_year.index, y=duration_year.duration.values)
         plt.box(on=None)
         plt.ylabel('Movie duration in minutes');
         plt.xlabel('Year of released');
         plt.title("Trends of Movie's Duration over the Years", fontsize=20, color='Green');
```



Explanation:

- In the years of **1960 to 1965**, Movie's durations were over **200 minutes, after 1965 the durations became comparatively shorter.**
- From the year **1980**, we can see consistent trend of movie durations, of which duration time is around in **between 100-150 minutes.**

7. Analysis of TV Shows with most number of seasons

--> Analysis of TV Shows with most number of seasons

In [25]:
```
tv_df['duration']=tv_df['duration'].str.replace(' Season','')
tv_df['duration']=tv_df['duration'].str.replace('s','')
tv_df['duration']=tv_df['duration'].astype(str).astype(int)
tv_df['duration']
```

Out[25]:
```
5       1
132     1
218     1
260     1
276     1
       ..
7385    1
7400    4
7492    1
7666    2
7721    2
Name: duration, Length: 145, dtype: int32
```

--> Extract TV Shows titles and its number of seasons:

In [26]:
```
#Extract the columns from tv_df
columns=['title','duration']
tv_shows = tv_df[columns]
```

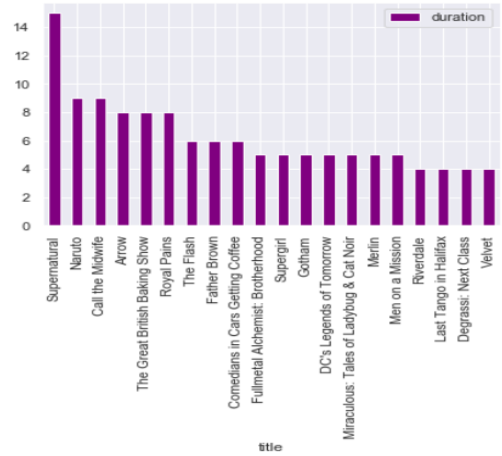
In [27]:
```
#sort the dataframe by number of seasons
tv_shows = tv_shows.sort_values(by='duration',ascending=False)
tv_shows
top20 = tv_shows[0:20]
top20
```

Out[27]:

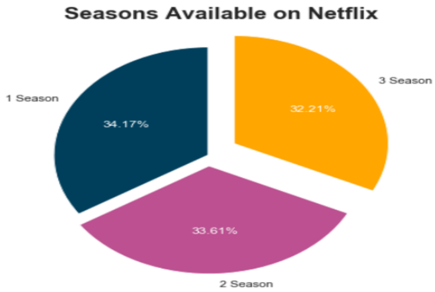
	title	duration
5912	Supernatural	15
4404	Naruto	9
1181	Call the Midwife	9
584	Arrow	8
6415	The Great British Baking Show	8
5291	Royal Pains	8
6359	The Flash	6
2130	Father Brown	6
1470	Comedians in Cars Getting Coffee	6
2313	Fullmetal Alchemist: Brotherhood	5
5908	Supergirl	5
2504	Gotham	5
1647	DC's Legends of Tomorrow	5
4121	Miraculous: Tales of Ladybug & Cat Noir	5
4047	Merlin	5
4033	Men on a Mission	5
5226	Riverdale	4
3544	Last Tango in Halifax	4
1687	Degrassi: Next Class	4
7400	Velvet	4

In [28]:
```
plt.figure(figsize=(10,6))
top20.plot(kind='bar',x='title',y='duration', color='purple')
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fdf02771c8>
<Figure size 720x432 with 0 Axes>
```



In [29]:
```
# TV SHOWS AND THEIR SEASONS
plt.figure(figsize=(8, 6))
labels=['1 Season', '2 Season', '3 Season']
_, _, texts = plt.pie(df.duration.value_counts()[:3], labels=labels, autopct='%1.2f%%', startangle=90,
                      explode=(0.0, 0.1, 0.2), colors=['#003f5c', '#bc5090', '#ffa600'])
plt.axis('equal')
plt.title('Seasons Available on Netflix', fontsize=20, fontweight='bold');
for text in texts:
    text.set_color('white')
```



Explanation:

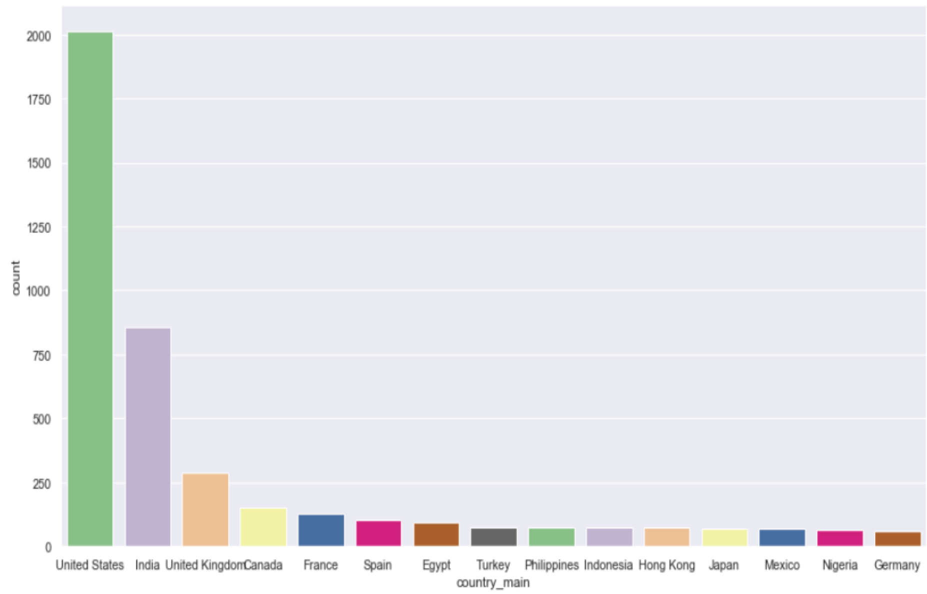
At first, we extract TV Shows titles and its number of seasons. Then sort the data frame by number of seasons. And after plotting it we get Supernatural and Naruto has highest number of seasons. At last, after using pie plot, we analyze-

35.04% TV Shows has only 1 Season,32.48% TV Shows has 2 seasons and 32.48% Tv Shows has 3 seasons available

8. Countries on top for movies content creation

In [30]:
```
plt.figure(figsize=(15,8))
sb.set(style="darkgrid")
sb.countplot(x="country_main", data=movie_df, palette="Accent", order=movie_df['country_main'].value_counts().index[0:15])
```

Out[30]: `<matplotlib.axes._subplots.AxesSubplot at 0x1fdf021d048>`



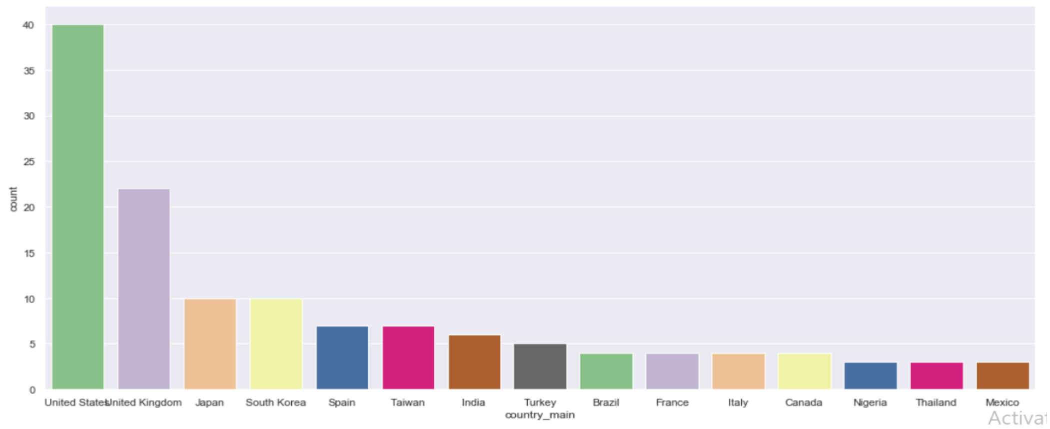
Explanation:

- **United States** creates highest number of movies followed by ***India and UK.***

9. Countries on top for TV Show content creation

In [31]:
```
plt.figure(figsize=(18,8))
sb.set(style="darkgrid")
sb.countplot(x="country_main", data=tv_df, palette="Accent", order=tv_df['country_main'].value_counts().index[0:15])
```

Out[31]: `<matplotlib.axes._subplots.AxesSubplot at 0x1fdf023c948>`

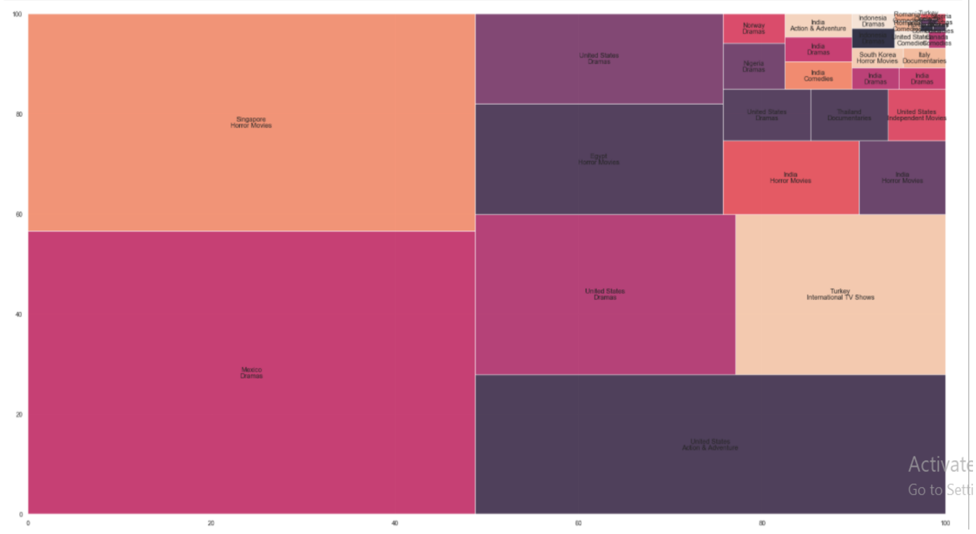


Explanation:

- **United States, United Kingdom, South Korea, Japan** creates most of the amount of TV Shows on Netflix.

10.Understanding what content is available in different countries

```
In [37]: # GENRES BY COUNTRIES
         import squarify
         plt.figure(figsize=(30,15))
         squarify.plot(value= gen_country['Genre'], label= gen_country['country_main'], sizes= size, alpha=0.8)
         plt.rc('font', size=14)
         plt.show()
```



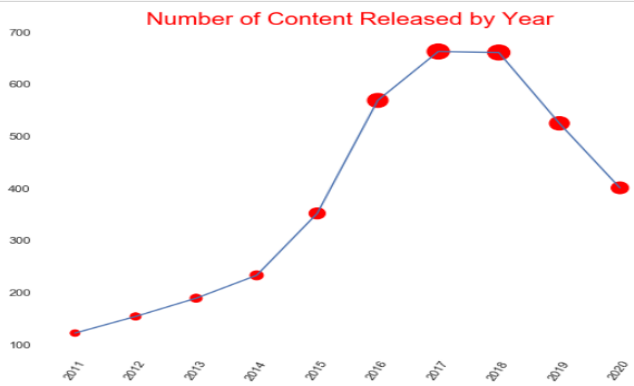
Explanation:

Here we plot genres by countries and analyse-United States produces most amount of content in 'Comedies' and 'Children & Family movies' Genres.

11.Contents released by years

```
In [38]: release = df['release_year'].value_counts()
         release = release.sort_index(ascending=True)

         plt.figure(figsize=(9,7))
         plt.plot(release[-11:-1])
         plt.scatter(release[-11:-1].index, release[-11:-1].values, s=0.5*release[-11:-1].values, c='Red');
         plt.box(on=None);
         plt.xticks(rotation = 60)
         plt.xticks(release[-11:-1].index);
         plt.title('Number of Content Released by Year', color='red', fontsize=20);
```

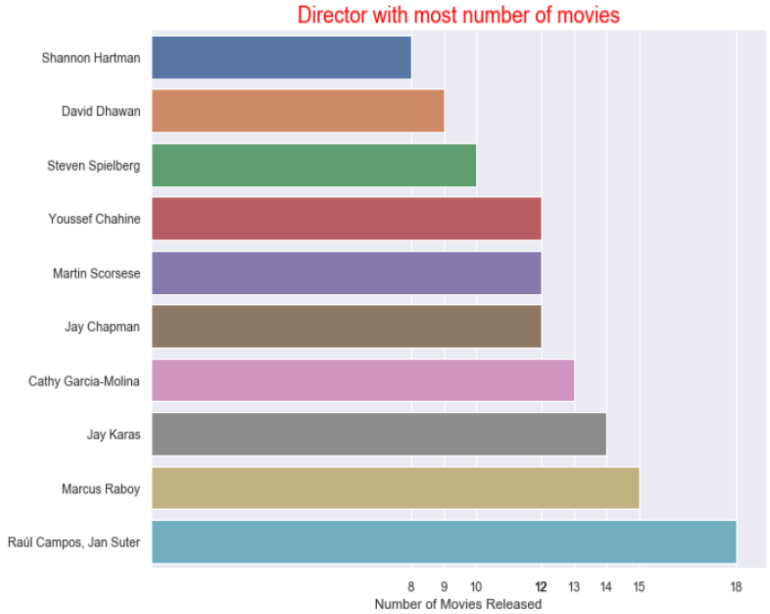


Explanation:

In the year of 2017-2018 the greatest number of contents released.

12.Directors with most number of Movies produced

In [39]:
```
plt.figure(figsize=(10,8))
sb.barplot(y= movie_df.director.value_counts()[:10].sort_values().index, x=movie_df.director.value_counts()[:10].sort_values().v
plt.title('Director with most number of movies', color='red', fontsize=18)
plt.xticks(movie_df.director.value_counts()[:10].sort_values().values);
plt.xlabel('Number of Movies Released');
```



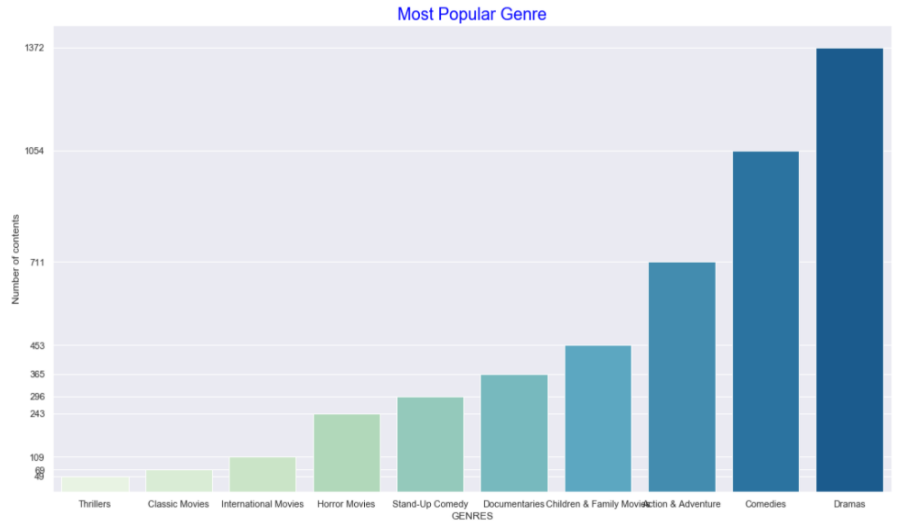
Ac
Go

Explanation:

- Director **Raul Campos, Jan Suter** Produced **highest number of movies: 18** on Netflix till now.

13.Most Popular Genre on Netflix

```
In [41]: plt.figure(figsize=(18,10))
         sb.barplot(x= df.Genre.value_counts()[:10].sort_values().index, y=df.Genre.value_counts()[:10].sort_values().values,palette='GnB
         plt.title('Most Popular Genre', color='Blue', fontsize=20)
         plt.yticks(df.Genre.value_counts()[:10].sort_values().values);
         plt.xlabel('GENRES');
         plt.ylabel('Number of contents');
```

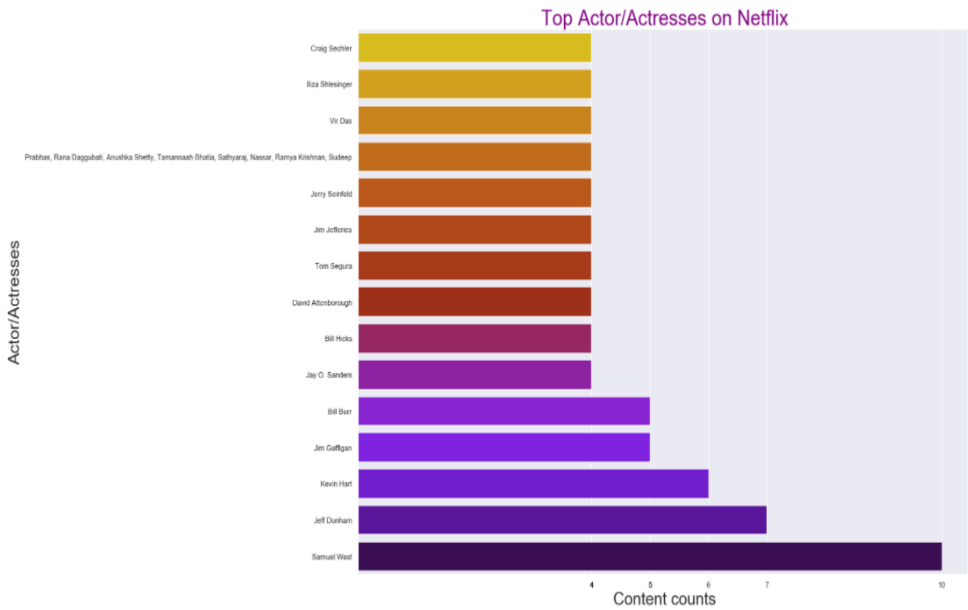


Explanation:

Drama is the most popular genre.

14.Top Actor/Actresses on Netflix

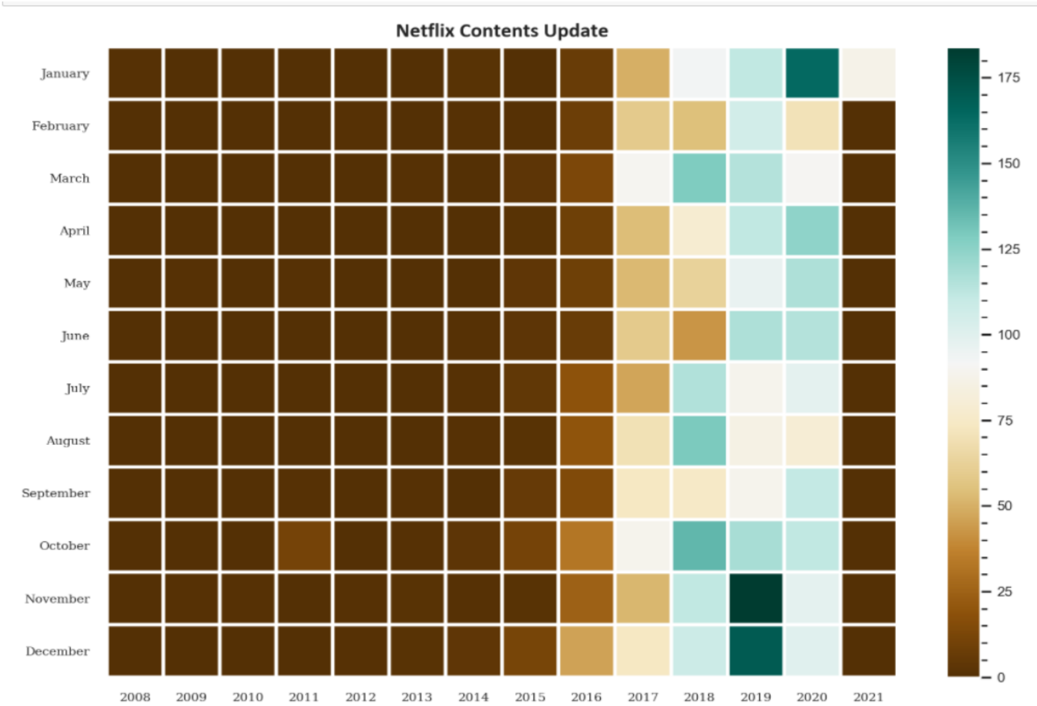
```
In [42]: plt.figure(figsize=(18,14))
         sb.barplot(y= df.cast.value_counts()[:15].sort_values().index, x=df.cast.value_counts()[:15].sort_values().values,palette='gnup
         plt.title('Top Actor/Actresses on Netflix', color='purple', fontsize=30)
         plt.xticks(df.cast.value_counts()[:10].sort_values().values);
         plt.ylabel('Actor/Actresses', fontsize=25);
         plt.xlabel('Content counts', fontsize=25);
```



Explanation:

- Actor **'Samuel West'** has highest number of movies/Tv shows on Netflix.

15.Best Month for directors to release content



Explanation:
We can **analyze** the months in which least **number** of contents are added, that months can be best for directors to release their content for better audience attention.

Plot description based Recommender

In [45]:
```
df['description'].head()
```

Out[45]:
```
1    After a devastating earthquake hits Mexico Cit...
2    When an army recruit is found dead, his fellow...
3    In a postapocalyptic world, rag-doll robots hi...
4    A brilliant group of students become card-coun...
5    A genetics professor experiments with a treatm...
Name: description, dtype: object
```

We need to convert the word vector of each overview.We'll compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each description.The overall importance of each word to the documents in which they appear is equal to TF * IDF.This is done to reduce the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score.

In [46]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

#Define a TF-IDF Vectorizer Object. Remove all english stop words such as 'the', 'a'
tfidf = TfidfVectorizer(stop_words='english')
```

In [47]:
```
#Replace NaN with an empty string
df['description'] = df['description'].fillna('')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df['description'])

#Output the shape of tfidf_matrix
tfidf_matrix.shape
```

Out[47]:
```
(4979, 13910)
```

In [48]:
```
# Import linear_kernel
from sklearn.metrics.pairwise import linear_kernel

# Compute the cosine similarity matrix
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
```

-- we need a mechanism to identify the index of a movie in our metadata DataFrame, given its title.

In [49]:
```
#Construct a reverse map of indices and movie titles
indices = pd.Series(df.index, index=df['title']).drop_duplicates()
```

-- Let's define a function that takes in a movie title as an input and outputs a list of the 10 most similar movies.

In [50]:
```
def get_recommendations(title, cosine_sim=cosine_sim):
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df['title'].iloc[movie_indices]
```

```
In [51]: get_recommendations('#realityhigh')
Out[51]: 4427    Natural Selection
         4137           Miss India
         6156      The Book of Sun
         4989           Prom Night
         6341     The F**k-It List
         765                Battle
         1145              Burning
         7326             Uncorked
         7662              Work It
         5242             Rock On!!
         Name: title, dtype: object

In [52]: get_recommendations('PK')
Out[52]: 133                            7 años
         4803                           Payday
         3478                 Kyaa Kool Hai Hum
         1243                 Catching Feelings
         3940    Mariah Carey's Merriest Christmas
         2154                            Fifty
         3703                             Loev
         1269                          Chameli
         3576         LEGENDS OF THE HIDDEN TEMPLE
         7754                             Yuva
         Name: title, dtype: object
```

Explanation:

We will calculate similarity scores for all movies based on their plot descriptions and recommend movies based on that similarity score. The plot description is given in the **description** feature of our dataset.

We need to convert the word vector of each overview. We'll compute Term Frequency-Inverse Document Frequency (TF-IDF) vectors for each description. The overall importance of each word to the documents in which they appear is equal to TF * IDF. This is done to reduce the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score.

Since we have used the TF-IDF vectorizer, calculating the dot product will directly give us the cosine similarity score. Therefore, we will use sklearn's **linear_kernel()** instead of cosine_similarities() since it is faster.

we need a mechanism to identify the index of a movie in our metadata Data Frame, given its title.
Then we define a function that takes in a movie title as an input and outputs a list of the 10 most similar movies.

At last, this is completely plot based recommendations. we can see these are not so accurate, so we can try to add more metrics to improve model performance.

Multiple metrics(Genre,cast,director) based Recommender System

In [53]:
```
features=['Genre','director','cast','description','title']
filters = df[features]
```

In [54]:
```
#Cleaning the data by making all the words in lower case.
def clean_data(x):
        return str.lower(x.replace(" ", ""))
```

In [55]:
```
for feature in features:
    filters[feature] = filters[feature].apply(clean_data)

filters.head()
```

Out[55]:

	Genre	director	cast	description	title
1	dramas	jorgemichelgrau	demiánbichir,héctorbonilla,oscarserrano,azalia...	afteradevastatingearthquakehitsmexicocity,trap...	7:19
2	horrormovies	gilbertchan	teddchan,stellachung,henleyhii,lawrencekoh,tom...	whenanarmyrecruitisfounddead,hisfellowsoldiers...	23:59
3	action&adventure	shaneacker	elijahwood,johnc.reilly,jenniferconnelly,chris...	inapostapocalypticworld,rag-dollrobotshideinfe...	9
4	dramas	robertluketic	jimsturgess,kevinspacey,katebosworth,aaronyoo,...	abrilliantgroupofstudentsbecomecard-countingex...	21
5	internationaltvshows	serdarakar	erdalbeşikçioğlu,yaseminallen,melisbirkan,sayg...	ageneticsprofessorexperimentswithatreatmentfor...	46

- We can now create our "metadata soup", which is a string that contains all the metadata that we want to feed to our vectorizer.

In [56]:
```
def create_soup(x):
    return x['director'] + ' ' + x['cast'] + ' ' +x['Genre']+' '+ x['description']
```

In [57]:
```
filters['soup'] = filters.apply(create_soup, axis=1)
```

In [58]:
```
# Import CountVectorizer and create the count matrix
from sklearn.feature_extraction.text import CountVectorizer

count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(filters['soup'])
```

In [59]:
```
# Compute the Cosine Similarity matrix based on the count_matrix
from sklearn.metrics.pairwise import cosine_similarity

cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
```

In [60]:
```
filters
```

Out[60]:

	Genre	director	cast	description	title	
1	dramas	jorgemichelgrau	demiánbichir,héctorbonilla,oscarserrano,azalia...	afteradevastatingearthquakehitsmexicocity,trap...	7:19	demi
2	horrormovies	gilbertchan	teddchan,stellachung,henleyhii,lawrencekoh,tom...	whenanarmyrecruitisfounddead,hisfellowsoldiers...	23:59	teddchan
3	action&adventure	shaneacker	elijahwood,johnc.reilly,jenniferconnelly,chris...	inapostapocalypticworld,rag-dollrobotshideinfe...	9	elijahwc
4	dramas	robertluketic	jimsturgess,kevinspacey,katebosworth,aaronyoo,...	abrilliantgroupofstudentsbecomecard-countingex...	21	jimsturge
5	internationaltvshows	serdarakar	erdalbeşikçioğlu,yaseminallen,melisbirkan,sayg...	ageneticsprofessorexperimentswithatreatmentfor...	46	erdalbeşik
...	...	...	...	...	...	
7778	comedies	rubenfleischer	jesseeisenberg,woodyharrelson,emmastone,abigai...	lookingtosurviveinaworldtakenoverbyzombies,ado...	zombieland	jesseeis
7780	dramas	shloksharma	shashankarora,shwetatripathi,rahulkumar,gopalk...	adrugdealerstartshavingdoubtsabouthistradeashi...	zoo	shashank
7781	children&familymovies	peterhewitt	timallen,courteneycox,chevychase,katemara,ryan...	draggedfromcivilianlife,aformersuperheromusttr...	zoom	timallen,cc

Activate W
Go to Settings

```
In [62]: def get_recommendations_new(title, cosine_sim=cosine_sim):
             title=title.replace(' ','').lower()
             idx = indices[title]

             # Get the pairwsie similarity scores of all movies with that movie
             sim_scores = list(enumerate(cosine_sim[idx]))

             # Sort the movies based on the similarity scores
             sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

             # Get the scores of the 10 most similar movies
             sim_scores = sim_scores[1:11]

             # Get the movie indices
             movie_indices = [i[0] for i in sim_scores]

             # Return the top 10 most similar movies
             return df['title'].iloc[movie_indices]
```

```
In [63]: get_recommendations_new('PK', cosine_sim2)

Out[63]: 100                         3 Idiots
         6585    The Legend of Michael Mishra
         552                Anthony Kaun Hai?
         4278                  Mumbai Matinee
         1004                     BluffMaster!
         2149               Ferrari Ki Sawaari
         1271                  Chance Pe Dance
         1831                          Dostana
         1878                            Drive
         1940               Ek Main Aur Ekk Tu
         Name: title, dtype: object
```

```
In [64]: get_recommendations_new('Black panther', cosine_sim2)

Out[64]: 4247    Mowgli: Legend of the Jungle
         1236                 Casino Tycoon 2
         2837                    How It Ends
         5750            Spenser Confidential
         7392                   Vantage Point
         718                     Bang Rajan 2
         4607              Olympus Has Fallen
         1569                     Da 5 Bloods
         3006                       Inkheart
         4667             Operation Chromite
         Name: title, dtype: object
```

```
In [65]: get_recommendations_new('Naruto', cosine_sim2)

Out[65]: 4405                   Naruto Shippûden the Movie: Bonds
         4410      Naruto the Movie 2: Legend of the Stone of Gelel
         4407                     Naruto Shippuden : Blood Prison
         4406          Naruto Shippûden the Movie: The Will of Fire
         4408                         Naruto Shippuden: The Movie
         4411     Naruto the Movie 3: Guardians of the Crescent ...
         4409           Naruto Shippuden: The Movie: The Lost Tower
         2313                     Fullmetal Alchemist: Brotherhood
         6477                             The Idhun Chronicles
         2431                                  Girls und Panzer
         Name: title, dtype: object
```

Explanation:

At first from the Genre, cast and director features, we need to extract the three most important actors, the director and genres associated with that movie.

Now we can now create our "metadata soup", which is a string that contains all the metadata that we want to feed to our vectorizer.

The next steps are the same as what we did with our plot description-based recommender. One important difference is that we use the **CountVectorizer()** instead of TF-IDF.

CONCLUSION

This Software Requirement Specification document is prepared to give requirement details of the project “Movie Recommendation System”. The detailed functional and nonfunctional requirements, system, hardware interfaces, data are stated in an extended outline. This document will be helpful at constituting a basis for design and development of the system to be developed. The Movie recommendation system is now being used widely in the OTT platforms to give better user experience.

BIBLIOGRAPHY

1. https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109

2. https://www.analyticsvidhya.com/blog/2020/11/create-your-own-movie-movie-recommendation-system/

3. https://www.kaggle.com/rounakbanik/movie-recommender-systems