Importation of modules

In [1]:

```
#Install packages/modules

#!pip install plotly
```

In [2]:

```
import pandas as pd

#Import visualization module
import matplotlib.pyplot as plt
from matplotlib import style
style.use("ggplot")

import seaborn as sns
import plotly.graph_objs as go
import numpy as np

#Logistic regression
from sklearn.linear_model import LogisticRegression

#Sklearn
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split

#Preprocessing
from sklearn.preprocessing import LabelEncoder

#Accuracy test
from sklearn.metrics import accuracy_score

#import warning module to stop showing modules
import warnings
warnings.filterwarnings("ignore")

%matplotlib inline
```

In [3]:

```
#features_train, features_test, labels_train, labels_test = train_test_split(word_da
```

Import data

In [4]:

```
data = pd.read_excel('netflix_titles.xls')
```

In [5]:

```
#DDisplay read csv dataset
data.head()
```

Out[5]:

	show_id	type	title	director	cast	country	date_added	release_year	rating
0	s1	TV Show	0.03	NaN	Jo√£o Miguel, Bianca Comparato, Michel Gomes, ...	Brazil	August 14, 2020	2020	TV-MA
1	s2	Movie	07:19:00	Jorge Michel Grau	Demi√°n Bichir, H√©ctor Bonilla, Oscar Serrano...	Mexico	December 23, 2016	2016	TV-MA
2	s3	Movie	23:59:00	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	R
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PG-13
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	January 1, 2020	2008	PG-13

In [6]:

```
# Data types of each columns
data.dtypes
```

Out[6]:

```
show_id         object
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 [7]:

```
#get null and type

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7787 entries, 0 to 7786
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       7787 non-null   object
 1   type          7787 non-null   object
 2   title         7787 non-null   object
 3   director      5398 non-null   object
 4   cast          7069 non-null   object
 5   country       7280 non-null   object
 6   date_added    7777 non-null   object
 7   release_year  7787 non-null   int64 
 8   rating        7780 non-null   object
 9   duration      7787 non-null   object
 10  listed_in     7787 non-null   object
 11  description   7787 non-null   object
dtypes: int64(1), object(11)
memory usage: 730.2+ KB
```

In [8]:

```
# columns that have missing data
data.isna()
```

Out[8]:

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration
0	False	False	False	True	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
7782	False	False	False	False	False	False	False	False	False	False
7783	False	False	False	False	False	False	False	False	False	False
7784	False	False	False	True	False	True	False	False	False	False
7785	False	False	False	True	False	False	False	False	False	False
7786	False	False	False	False	True	False	False	False	False	False

7787 rows × 12 columns

In [9]:

```
data.isna().sum()
```

Out[9]:

```
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
```

In [10]:

```
#Get value count of each column
data.groupby(["director", "type"])["type"].nunique()
```

Out[10]:

```
director               type
A. L. Vijay            Movie    1
A. Raajdheep           Movie    1
A. Salaam              Movie    1
A.R. Murugadoss        Movie    1
Aadish Keluskar        Movie    1
                                ..
√áagan Irmak            Movie    1
√Çsold Uggad√≥ttir     Movie    1
√Ìskar Th√≥r Axelsson  Movie    1
√ñmer Faruk Sorak      Movie    1
≈ûenol S√∂nmez         Movie    1
Name: type, Length: 4086, dtype: int64
```

In [11]:

```
#Get columns
data.columns
```

Out[11]:

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'dat
e_added',
       'release_year', 'rating', 'duration', 'listed_in', 'descriptio
n'],
      dtype='object')
```

In [12]:

```
#Get shape of data
data.shape
```

Out[12]:

```
(7787, 12)
```

In [13]:

```
#Get dataset summary
data.describe()

"""This is unecessary since the dataset is composed of non-numerical figures"""
```

Out[13]:

```
'This is unecessary since the dataset is composed of non-numerical fig
ures'
```

Data Preprocessing

How to handle the missing data - Remove the null values, the dataset is mostly composed of non-numerical values.

In [14]:

```
#Get total of null values
data.isna().sum()
```

Out[14]:

```
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
```

In [15]:

```
#Creating new data with no null values
data_two = data.dropna()
```

In [16]:

```
#Test for null values existence
data_two.isna().sum()
```

Out[16]:

```
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
```

In [17]:

```
# Check for duplicate
data_two.duplicated().sum()
```

Out[17]:

```
0
```

In [18]:

```
#function that clean up data

def cleanData(data):
    #remove null values
    data_two = data.dropna()

    #return new dataframe
    return data_two

print(cleanData(data))

#Confirm if null values have been removed
data_two.isna().sum()
```

```
     show_id     type       title           director  \
1         s2    Movie    07:19:00  Jorge Michel Grau
2         s3    Movie    23:59:00       Gilbert Chan
3         s4    Movie           9        Shane Acker
4         s5    Movie          21     Robert Luketic
5         s6  TV Show          46        Serdar Akar
...      ...      ...         ...                ...
7778   s7779    Movie  Zombieland    Ruben Fleischer
7780   s7781    Movie         Zoo       Shlok Sharma
7781   s7782    Movie        Zoom       Peter Hewitt
7782   s7783    Movie        Zozo        Josef Fares
7783   s7784    Movie      Zubaan        Mozez Singh

                                                   cast  \
1     Demi√°n Bichir, H√©ctor Bonilla, Oscar Serrano...
2     Tedd Chan, Stella Chung, Henley Hii, Lawrence ...
3     Elijah Wood, John C. Reilly, Jennifer Connelly...
4     Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...
5     Erdal Be≈üik√ßioƒülu, Yasemin Allen, Melis Bir...
...                                                 ...
7778  Jesse Eisenberg, Woody Harrelson, Emma Stone, ...
7780  Shashank Arora, Shweta Tripathi, Rahul Kumar, ...
7781  Tim Allen, Courteney Cox, Chevy Chase, Kate Ma...
7782  Imad Creidi, Antoinette Turk, Elias Gergi, Car...
7783  Vicky Kaushal, Sarah-Jane Dias, Raaghav Chanan...

                                              country         date_a
dded  \
1                                              Mexico  December 23,
2016
2                                           Singapore  December 20,
2018
3                                       United States  November 16,
2017
4                                       United States    January 1,
2020
5                                              Turkey       July 1,
2017
...                                               ...
...
7778                                    United States   November 1,
2019
7780                                            India       July 1,
2018
7781                                    United States   January 11,
```

```
2020
7782  Sweden, Czech Republic, United Kingdom, Denmar...   October 19,
2020
7783                                                 India      March 2,
2019

      release_year rating  duration  \
1             2016  TV-MA    93 min
2             2011      R    78 min
3             2009  PG-13    80 min
4             2008  PG-13   123 min
5             2016  TV-MA  1 Season
...            ...    ...       ...
7778          2009      R    88 min
7780          2018  TV-MA    94 min
7781          2006     PG    88 min
7782          2005  TV-MA    99 min
7783          2015  TV-14   111 min

                                          listed_in  \
1                        Dramas, International Movies
2                     Horror Movies, International Movies
3     Action & Adventure, Independent Movies, Sci-Fi...
4                                                Dramas
5       International TV Shows, TV Dramas, TV Mysteries
...                                                 ...
7778                          Comedies, Horror Movies
7780   Dramas, Independent Movies, International Movies
7781                Children & Family Movies, Comedies
7782                     Dramas, International Movies
7783     Dramas, International Movies, Music & Musicals

                                           description
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...
...                                                 ...
7778  Looking to survive in a world taken over by zo...
7780  A drug dealer starts having doubts about his t...
7781  Dragged from civilian life, a former superhero...
7782  When Lebanon's Civil War deprives Zozo of his ...
7783  A scrappy but poor boy worms his way into a ty...

[4808 rows x 12 columns]
```

Out[18]:

```
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
```

In [19]:

```
data_two.head()
```

Out[19]:

	show_id	type	title	director	cast	country	date_added	release_year	ratin
1	s2	Movie	07:19:00	Jorge Michel Grau	Demi√°n Bichir, H√©ctor Bonilla, Oscar Serrano...	Mexico	December 23, 2016	2016	T\ M
2	s3	Movie	23:59:00	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PC 1
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	January 1, 2020	2008	PC 1
5	s6	TV Show	46	Serdar Akar	Erdal Be≈üik√ßioƒülu, Yasemin Allen, Melis Bir...	Turkey	July 1, 2017	2016	T\ M

Create new columns by extracting year

In [20]:

```
data_two['year_added'] = data_two.loc[:, 'date_added'].apply(lambda x: x.split(" ")[

data_two['year_added'].head(10)
```

Out[20]:

```
1     2016
2     2018
3     2017
4     2020
5     2017
6     2020
7     2019
8     2019
9     2017
10    2017
Name: year_added, dtype: object
```

In [21]:

```
data_two.date_added.head()
```

Out[21]:

```
1    December 23, 2016
2    December 20, 2018
3    November 16, 2017
4      January 1, 2020
5         July 1, 2017
Name: date_added, dtype: object
```

In [22]:

```
#Create new columns by extracting year

data_two['month_added'] = data_two.loc[:, 'date_added'].apply(lambda x: x.split(" ")

data_two.month_added.head(10)
```

Out[22]:

```
1     December
2     December
3     November
4      January
5         July
6         June
7     November
8        April
9     December
10     October
Name: month_added, dtype: object
```

In [23]:

```
#Confirming if the new colmns were added
data_two.columns
```

Out[23]:

```
Index(['show_id', 'type', 'title', 'director', 'cast', 'country', 'dat
e_added',
       'release_year', 'rating', 'duration', 'listed_in', 'descriptio
n',
       'year_added', 'month_added'],
      dtype='object')
```

Replace movies and TV Shows and Movies with 0 and 1

In [24]:

```
data_two.replace(['Tv Show', 'Movies'], [0, 1]).head()
```

Out[24]:

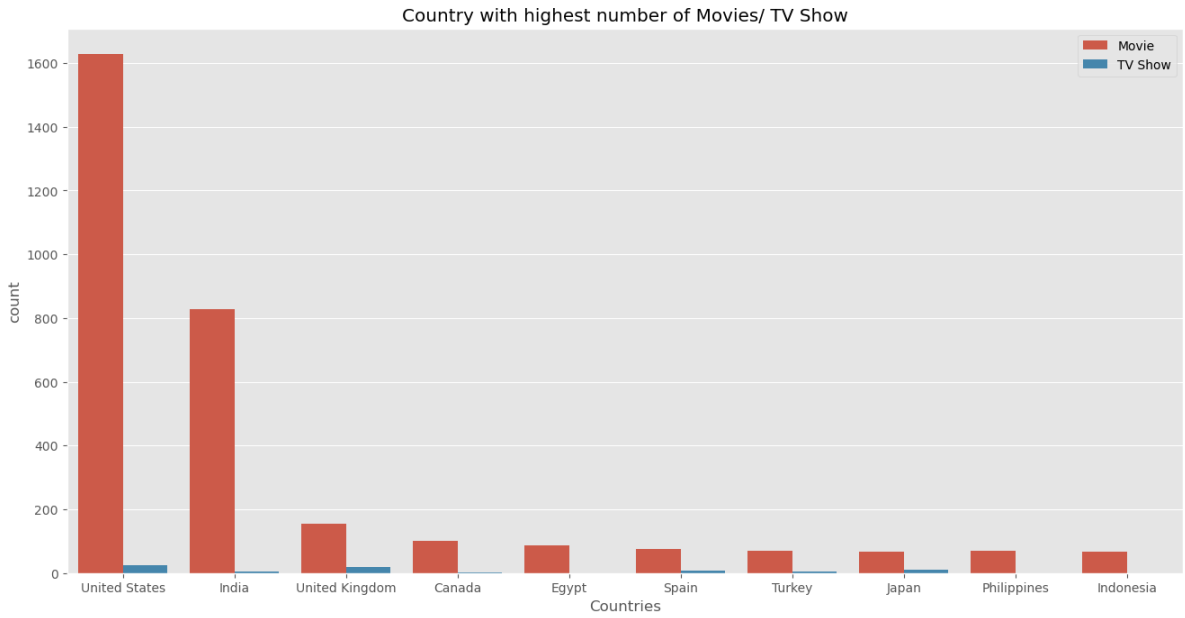
	show_id	type	title	director	cast	country	date_added	release_year	ratin
1	s2	Movie	07:19:00	Jorge Michel Grau	Demi√°n Bichir, H√©ctor Bonilla, Oscar Serrano...	Mexico	December 23, 2016	2016	TV M
2	s3	Movie	23:59:00	Gilbert Chan	Tedd Chan, Stella Chung, Henley Hii, Lawrence ...	Singapore	December 20, 2018	2011	
3	s4	Movie	9	Shane Acker	Elijah Wood, John C. Reilly, Jennifer Connelly...	United States	November 16, 2017	2009	PG 1
4	s5	Movie	21	Robert Luketic	Jim Sturgess, Kevin Spacey, Kate Bosworth, Aar...	United States	January 1, 2020	2008	PG 1
5	s6	TV Show	46	Serdar Akar	Erdal Be≈üik√ßio∆ülu, Yasemin Allen, Melis Bir...	Turkey	July 1, 2017	2016	TV M

Data Visualization

Noteable difference by country

In [25]:

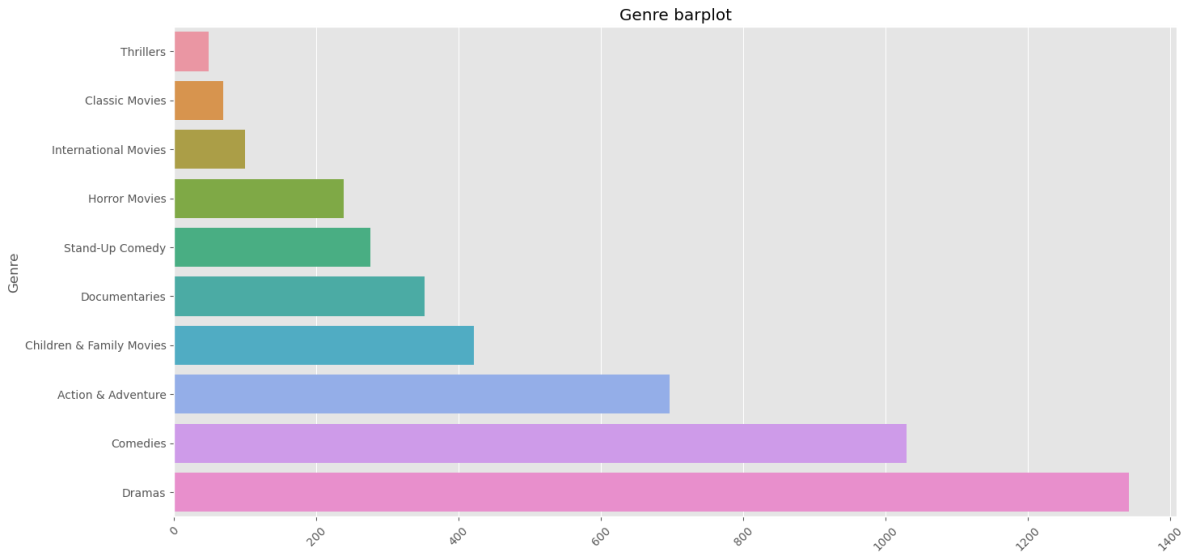
```
#determine size of visualization output
plt.figure(figsize=(16, 8), dpi = 100)

# Plot
sns.countplot(x = "country", data=data_two, order=data_two.country.value_counts()[:1

# Create title
plt.title('Country with highest number of Movies/ TV Show')
plt.xlabel('Countries')

# Visualize the plot
plt.legend(loc = 1)
plt.show()
```



Conclusion: United Stated has the largest viewership followed by India and United Kingdom

In [26]:

```
#Determine size of the visual
plt.figure(figsize=(16, 8), dpi = 100)

#Rename column kisted_in to genre
data_two = data_two.rename(columns={"listed_in": "genre"})

#Slpit the renamed column to get genre
data_two.genre = data_two.genre.apply(lambda x: x.split(", ")[0])

#Plot
sns.barplot(y=data_two.genre.value_counts()[:10].sort_values().index, x=data_two.ger

#Visualize data
plt.title('Genre barplot')
plt.ylabel('Genre')
plt.xticks(rotation=45)
plt.show()
```

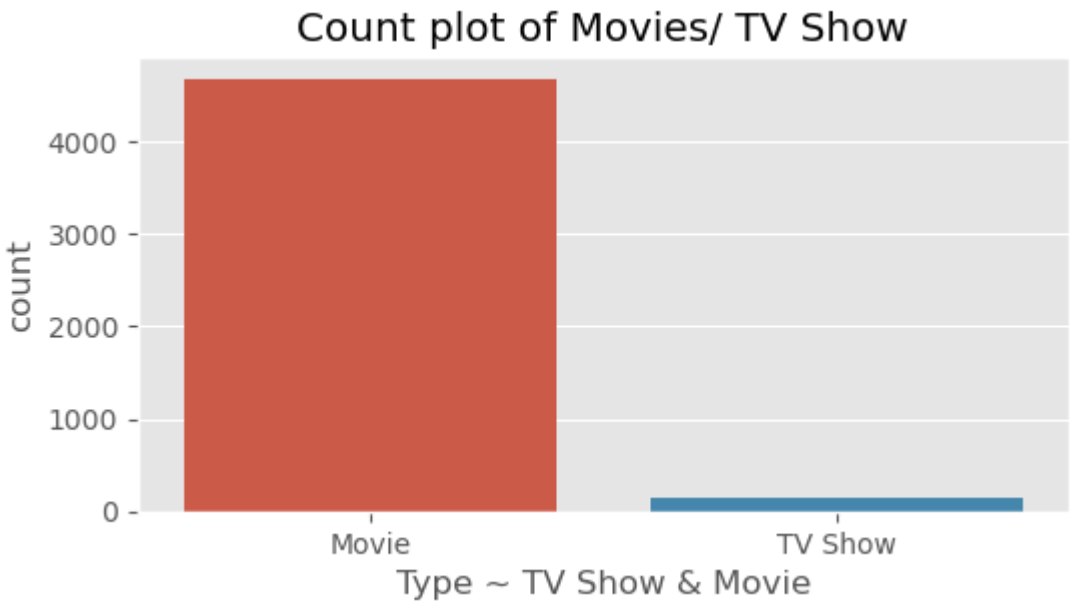


Conclusion: United Stated has the largest viewership followed by India and United Kingdom

Genre that have gotten less / more popular

In [27]:

```
#determine size of visualization output
plt.figure(figsize=(6, 3), dpi = 100)

#categorical plot
sns.countplot(x ='type', data = data_two)

# Create title
plt.title('Count plot of Movies/ TV Show')
plt.xlabel('Type ~ TV Show & Movie')
# Visualize the plot
plt.show()
```



Conclusion: Movies have gotten more popular than TV shows

Is Netflix investing in TV shows/ Movies

In [28]:

```
#determine size of visualization output
plt.figure(figsize=(8, 8), dpi = 100)

#Plot & defining
df_movie = data_two[data_two.type =='Movie'].groupby('release_year').count()

df_tv = data_two[data_two.type =='TV Show'].groupby('release_year').count()


df_movie.reset_index(level=0, inplace=True)
df_tv.reset_index(level=0, inplace=True)


# trend of movies and tv shows in recent year (from 1930 to 2020)
visual = go.Figure()
visual.add_trace(go.Scatter(x = df_movie.release_year, y = df_movie.type,
                    mode='lines',
                    name='Movies', marker_color='red'))

visual.add_trace(go.Scatter(x=df_tv.release_year, y=df_tv.type,
                    mode='lines',
                    name='TV Shows', marker_color='green'))

#Create titles
visual.update_layout(title_text='Trend Movies vs TV Shows in recent years',
                     xaxis_title="Year",
                     yaxis_title = 'Number of movies produced',
                     title_x=0.5)

# Visualize the plot
visual.show()
```

```
<Figure size 800x800 with 0 Axes>
```

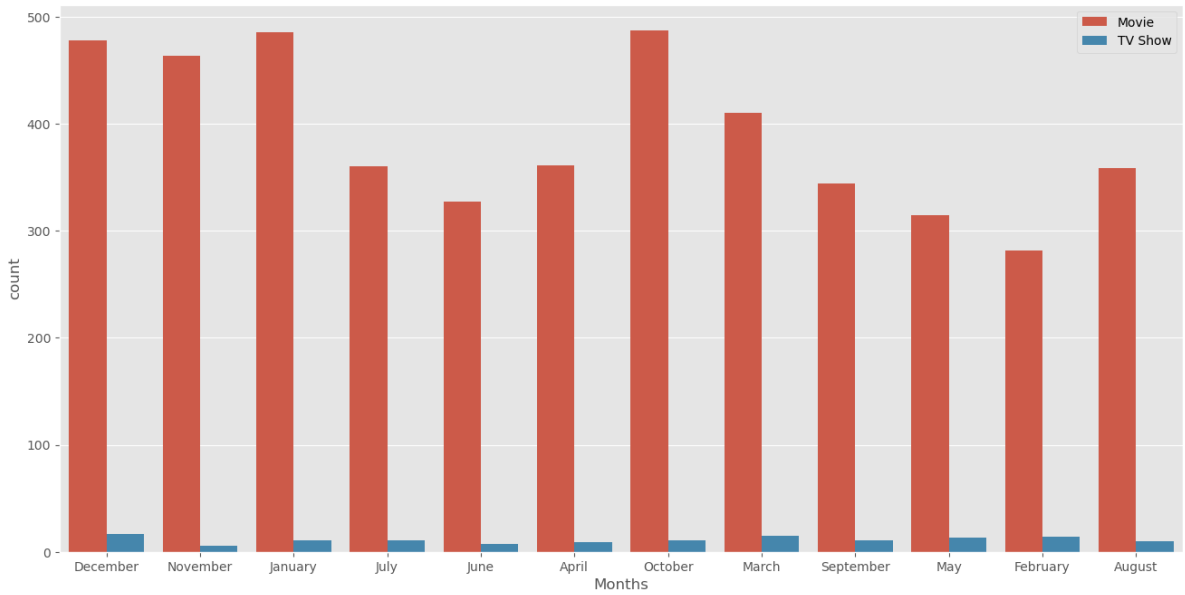
Conclusion:

Netflix is investing in movies this is evident in the trend, each year netflix is releasing movies than TV shows

Number of movies/ Tv Shows produced each month

In [29]:

```
#determine size of visualization output
plt.figure(figsize=(16, 8), dpi = 100)

#Plot & defining
sns.countplot(x = "month_added", data=data_two, hue="type")

#Titles
plt.xlabel('Months')

plt.legend(loc = 1)
plt.show()
```

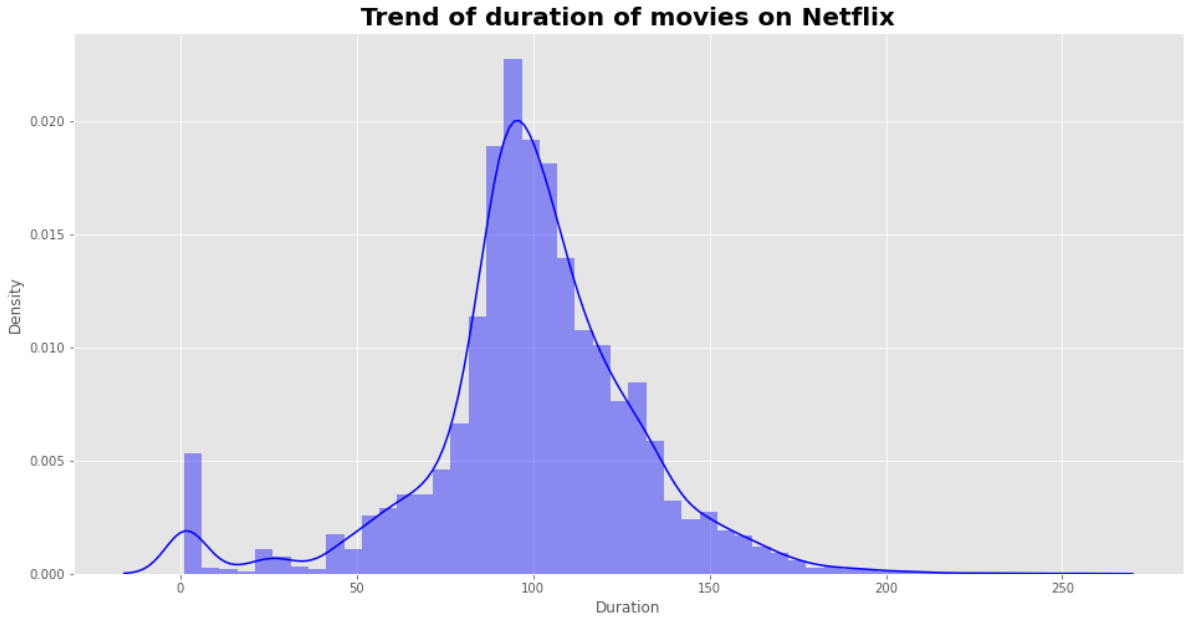


Conclusion: Movies are produced mostly in the month of October , January and December.

Trends in the length of movies

In [30]:

```
#Set size of visualization output
plt.figure(figsize = (16,8))

#Split column duration to get number of minutes only
data_two.minutes = data_two.duration.apply(lambda x : x.split(" ")[-2])

#Plot using seaborn
sns.distplot(data_two.minutes, color = 'blue')

#Create labels for tittle and X-axis
plt.title('Trend of duration of movies on Netflix', fontsize = 20, fontweight = 'bol
plt.xlabel('Duration')

#Visualize plot
plt.show()
```

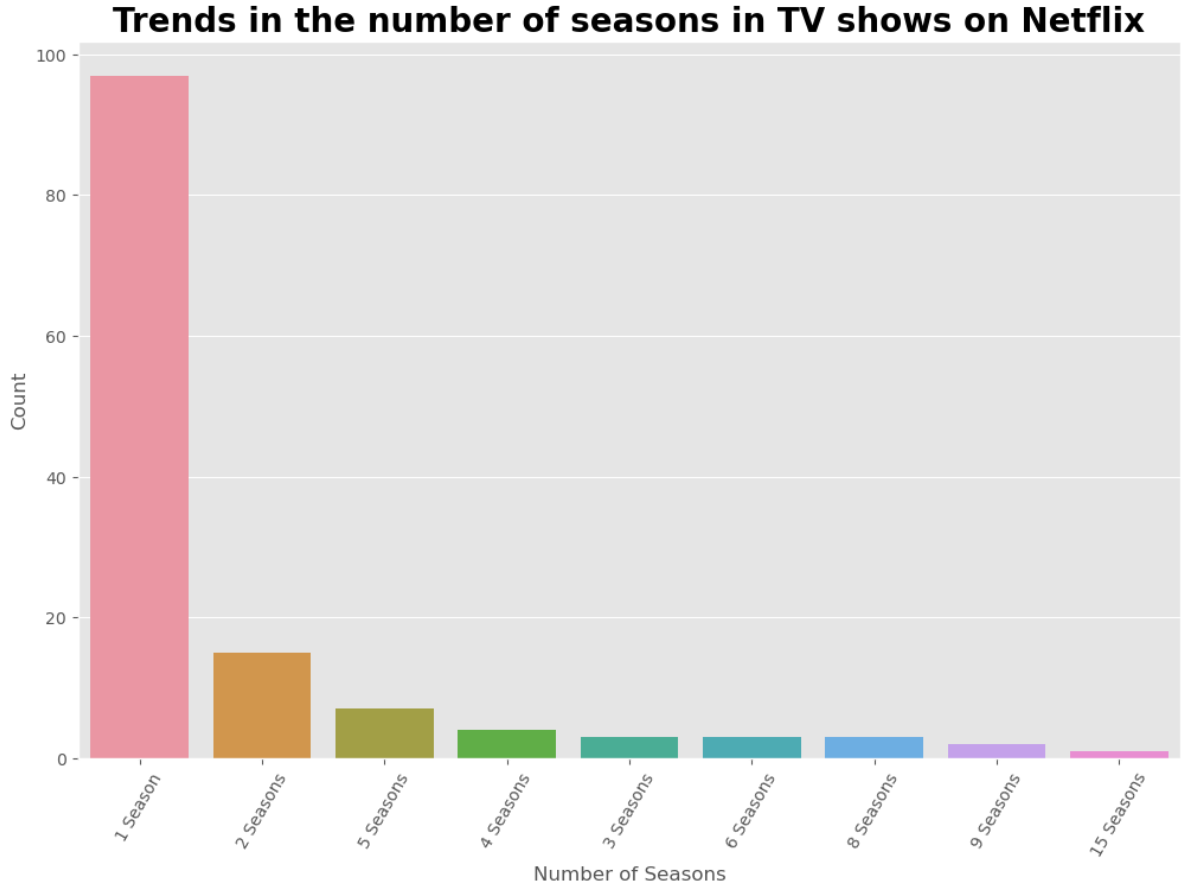


Conclusion: Majorities movies produced have a range of between 85 minutes to 120 minutes

Trends in the number of seasons in TV shows

In [31]:

```
#Define size of the visualization
plt.figure(figsize = (12,8), dpi = 100)



#Getting TV Shows seasons onlines
tv_dFrame = data_two[data_two.type == 'TV Show']

data_three = tv_dFrame.groupby(['duration'])['show_id'].count().reset_index().rename

# Plot
sns.barplot(x = 'duration' , y = 'Count' , data = data_three)

# Labelling
plt.title('Trends in the number of seasons in TV shows on Netflix', size = 20, fontw
plt.xlabel('Number of Seasons')
plt.xticks(rotation = 60)

# Show visualizaton
plt.show()
```



Conclusion:

Majority of the seasons consists of 1, 2 and 5 seasons.

Certain directors that netflix seems to like

In [32]:

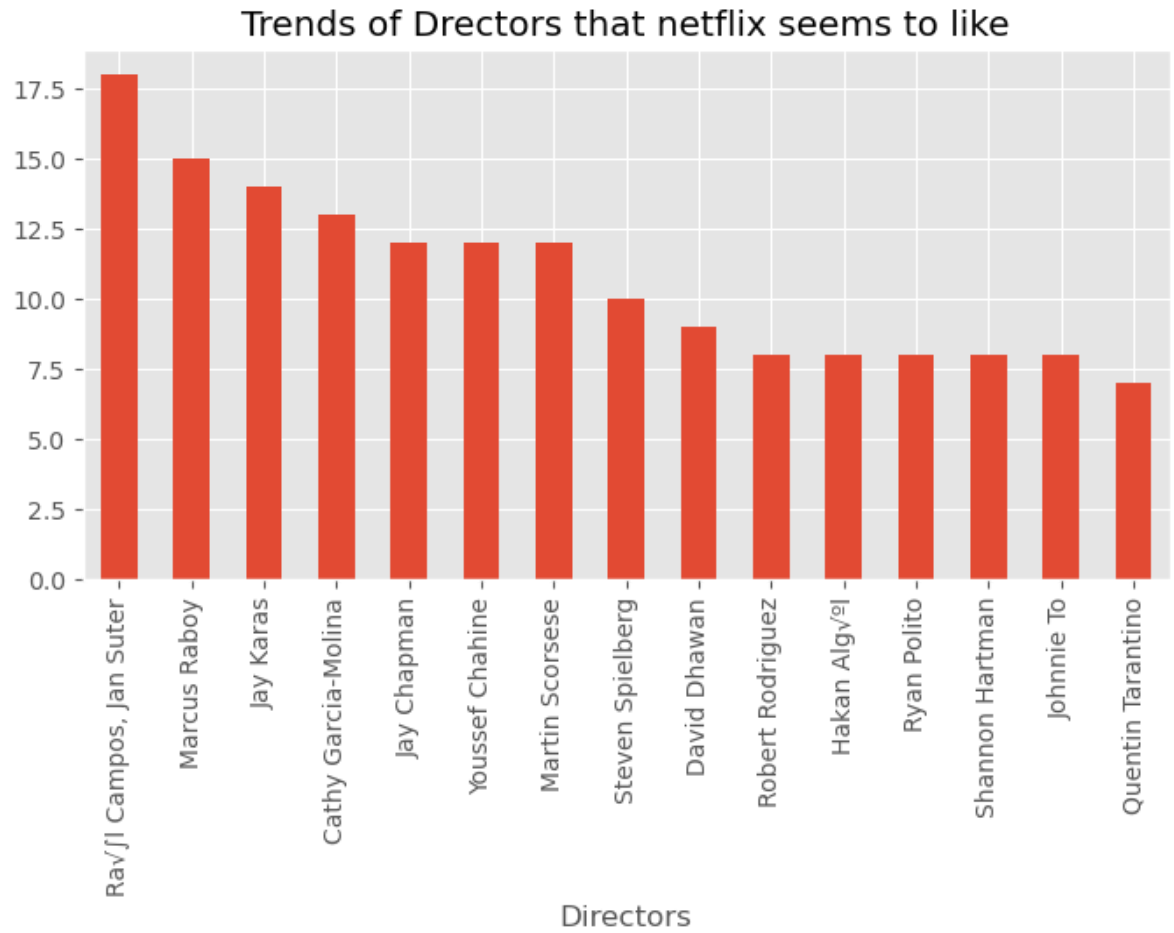
```
#To find unique actors
data_two.director.unique()
```

Out[32]:

```
array(['Jorge Michel Grau', 'Gilbert Chan', 'Shane Acker', ...,
       'Peter Hewitt', 'Josef Fares', 'Mozez Singh'], dtype=object)
```

In [33]:

```
# Detrmine size of the output(visualization)
plt.figure(figsize=(8,4), dpi = 100)

#Plot
director_data = data_two.director.value_counts().head(15)

director_data.plot(kind = 'bar')

# Create labels
plt.xlabel('Directors')
plt.title('Trends of Drectors that netflix seems to like')

# Show output
plt.show()
```



Conclusion: Raul Campus, Jan Suter and Marcus Raboy seems to be prefered by Netflix

MACHINE LEARNING

In [34]:

```
ml_df = data_two[["year_added" , "type"]]

#checking unie items in the type column
ml_df.type.unique()
```

Out[34]:

```
array(['Movie', 'TV Show'], dtype=object)
```

In [35]:

```
#convert the type data into numerical
le = LabelEncoder()
type_encoded = le.fit_transform(ml_df.type)

#print(type_encoded)
ml_df['encoded_type'] = type_encoded

#convert year_added object to numerical
#ml_df['year_added'] = ml_df['year_added'].astype(int) OR
ml_df['year_added'] = pd.to_numeric(ml_df['year_added'])
ml_df.head()
```

Out[35]:

	year_added	type	encoded_type
1	2016	Movie	0
2	2018	Movie	0
3	2017	Movie	0
4	2020	Movie	0
5	2017	TV Show	1

In [36]:

```
ml_df.dtypes
```

Out[36]:

```
year_added       int64
type            object
encoded_type     int64
dtype: object
```

LOGISTIC REGRESSION

In [37]:

```
#create object of Logistic regression
model = LogisticRegression()

#split dataset to train and split
X_train, X_test, y_train, y_test = train_test_split(ml_df[['year_added']], ml_df.enc

#train the dataset
model.fit(X_train, y_train)
```

Out[37]:

```
LogisticRegression()
```

In [38]:

```
model.predict(X_test)
```

Out[38]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [39]:

```
#Test Accuracy of the model
model.score(X_test, y_test)
```

Out[39]:

```
0.972972972972973
```

In [40]:

```
accuracy = accuracy_score(y_test, model.predict(X_test))
print('Accuracy: %.2f' % (accuracy*100))
```

```
Accuracy: 97.30
```

In [41]:

```
#Save model for later use
import pickle

with open("model_pickle", "wb") as f:
    pickle.dump(model, f)

#use the model
with open("model_pickle", "rb") as f:
    mp = pickle.load(f)

mp.predict([[2017]])
```

Out[41]:

```
array([0])
```

In [ ]:

In [ ]:

In [ ]:

ROUGH WORK

In [ ]: