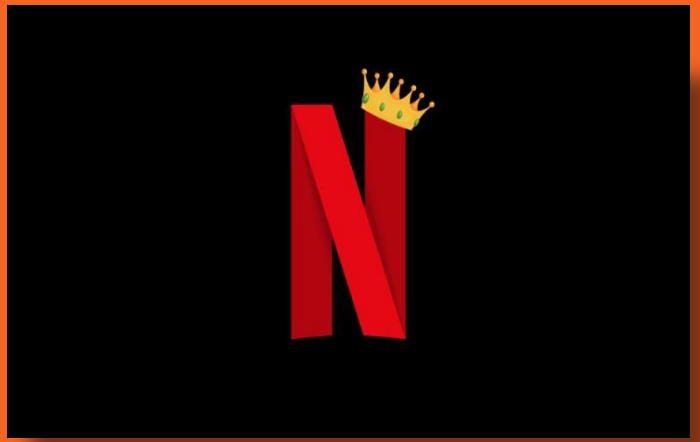
Netflix Movies and TV Shows



Group 3

What is the idea?

The main goal of this project is to bring the decision process on another level in which to be a guide for streaming investors in the Movie and TV distributors.

1. On Air

➔ **Satisfy**
Highlight what's new, unusual, or surprising with respect to the audience's demands.

➔ **Emotional**
Give people a reason to care. Why your platform? Such like Exxen, Amazon...

➔ **Detect**
Provide a couple more of the series from that specific director.

How many people do you need to know to **understand what the rest of the world needs?**

Tip

In this example, we're leading off with something **unexpected.**

While the audience is trying to come up with a specific style that they are interested in, we'll surprise them with the next slide.

More than one! Follow the style.

- **The main goal** of this project :
  - *is to make the potentials of the projects more visible*
  - *help them decide with more valid reasons.*
- Audience preference is important, and usually directors tend to keep their styles going on.
- All is made for the investors in the film industry, which anyone can visit anytime.

HOW TO IMPLEMENT THE ALGORITHM?

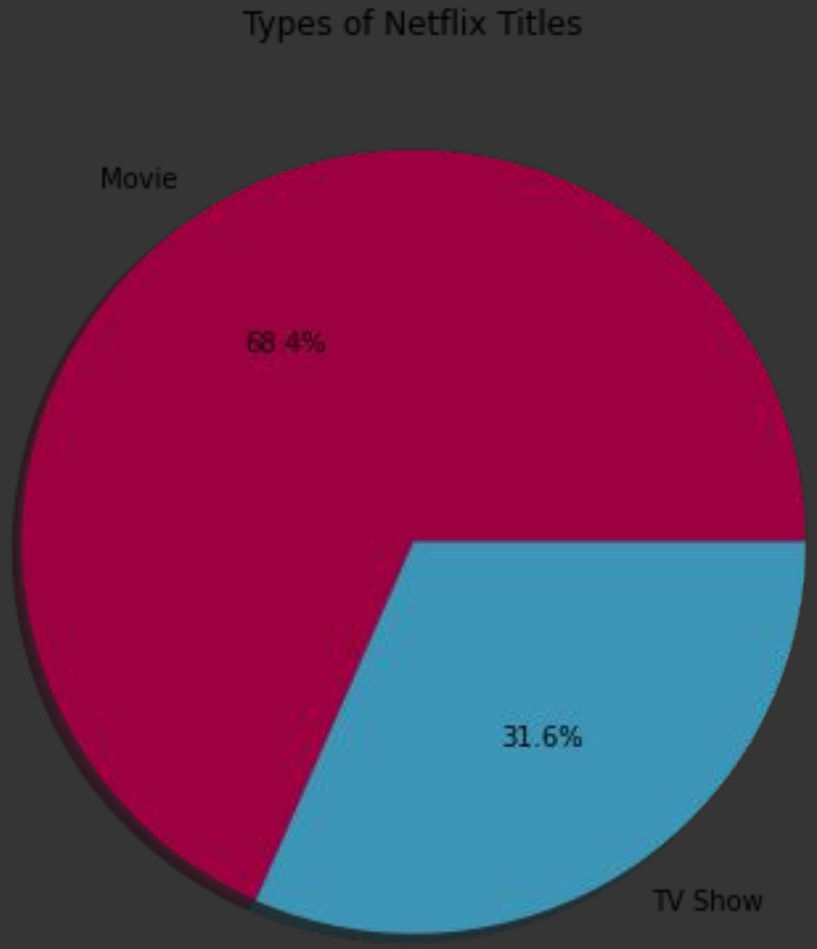
- Two different Machine Learning methods have been used, Decision Tree and Random Forest, respectively.
- IMDB Scores have been added into the data set and an unique IMDB Score level has been fixed in order to specify a valid score in this problem.
- Same data set applies for each methods, which have been separated as %80 for training, %10 for test and %10 to validate the data.
- Another change in the data set has been made by removing the actor names and titles in order to increase the efficiency of the algorithms.

DATA EXPLORATION

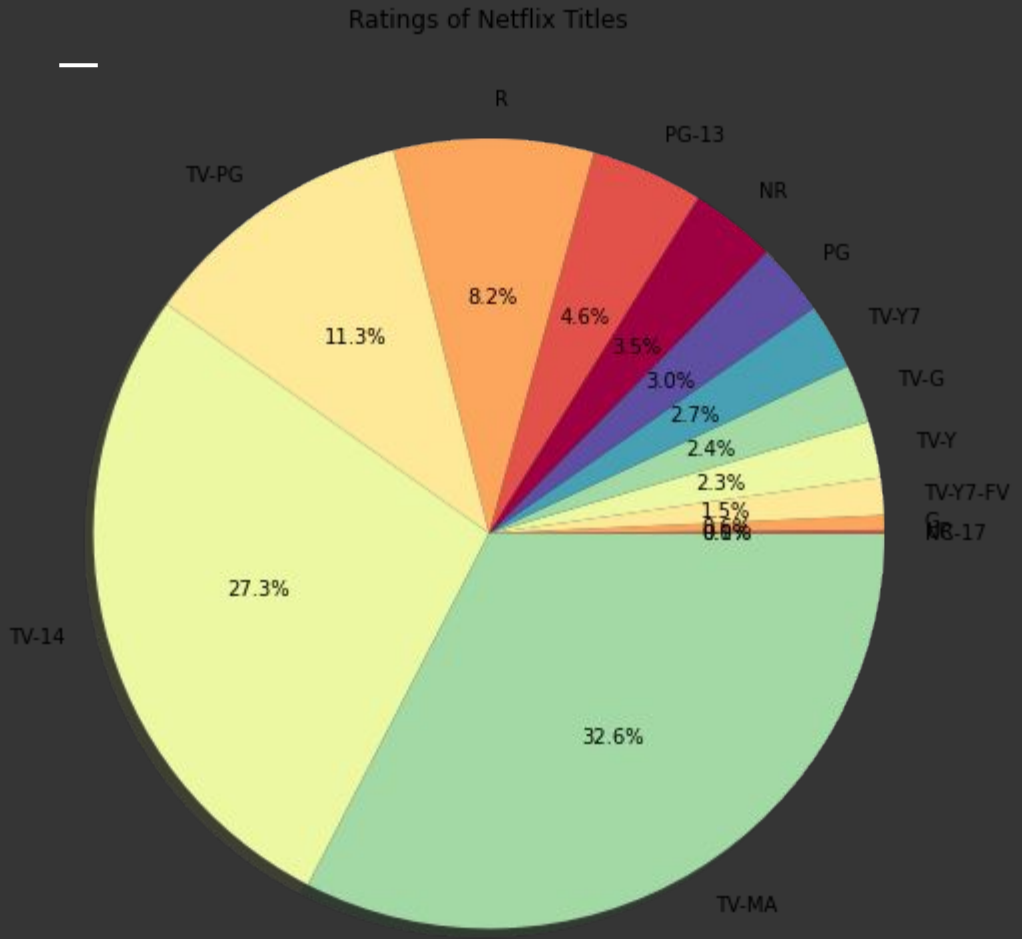
- Before developing a tool, an algorithm, adding data, we should first understand the data we already have.
- Important to understand the users tendency and desires.

- There is a need for a detailed exploration to find a causation and correlation.
- Pie charts and plots are to observe more clearly and understand better.

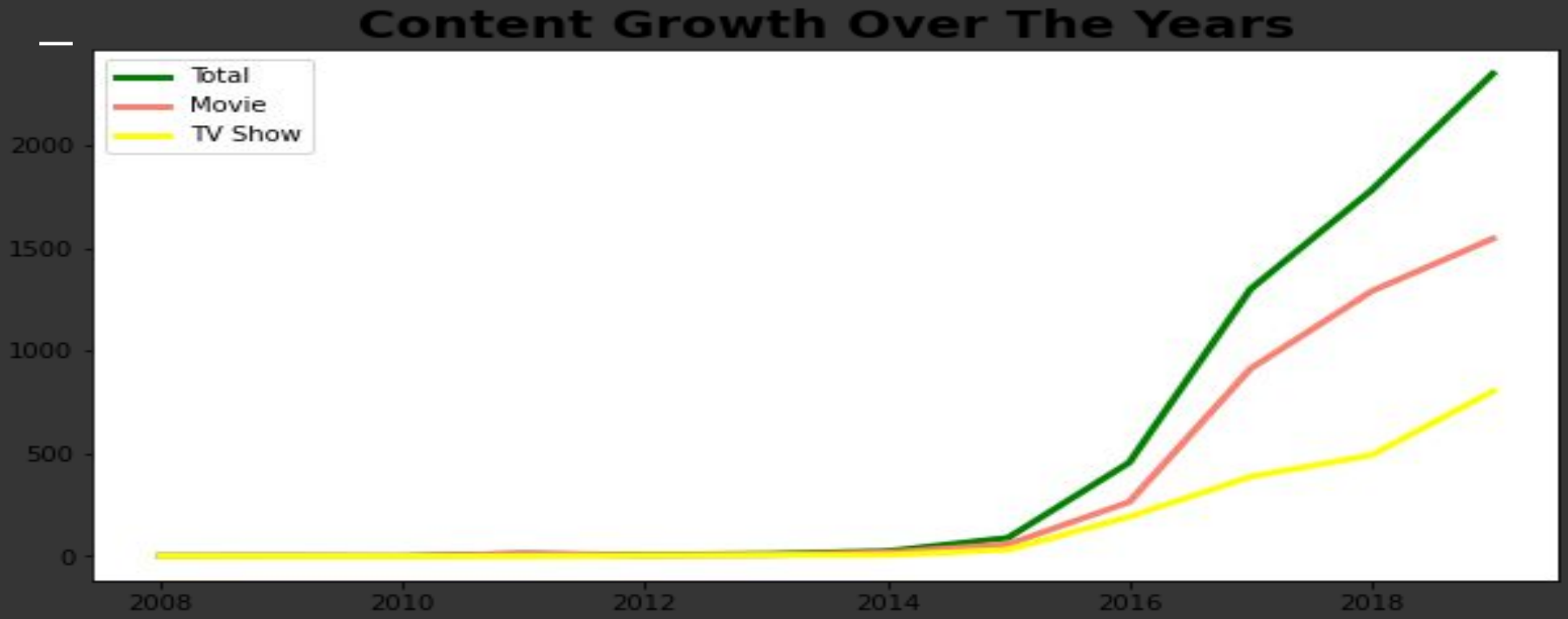
- After observing, we move to the step of modifying the data:
- Eliminating the null data
- Adding IMDB data to the previous given data



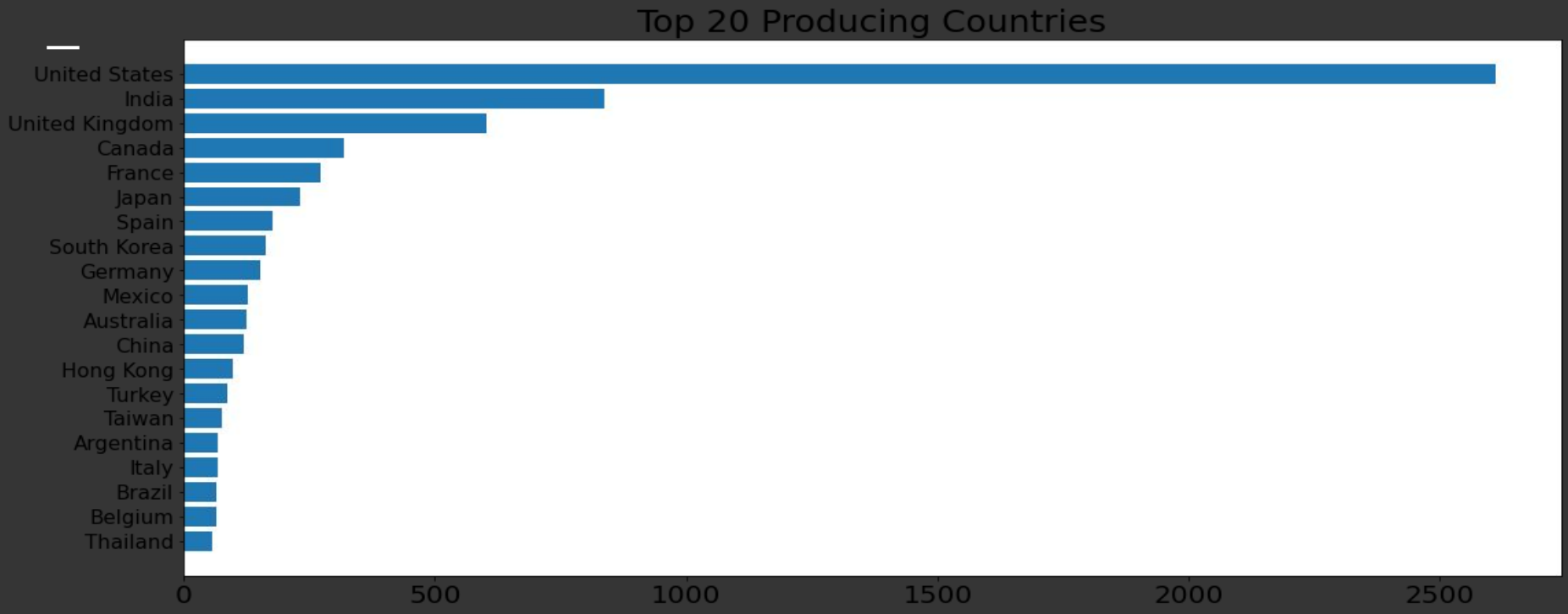
- Demonstrating the percentages of the types
- 68.4% movies
- 31.6% TV Show
- See the distribution of the users’ choices



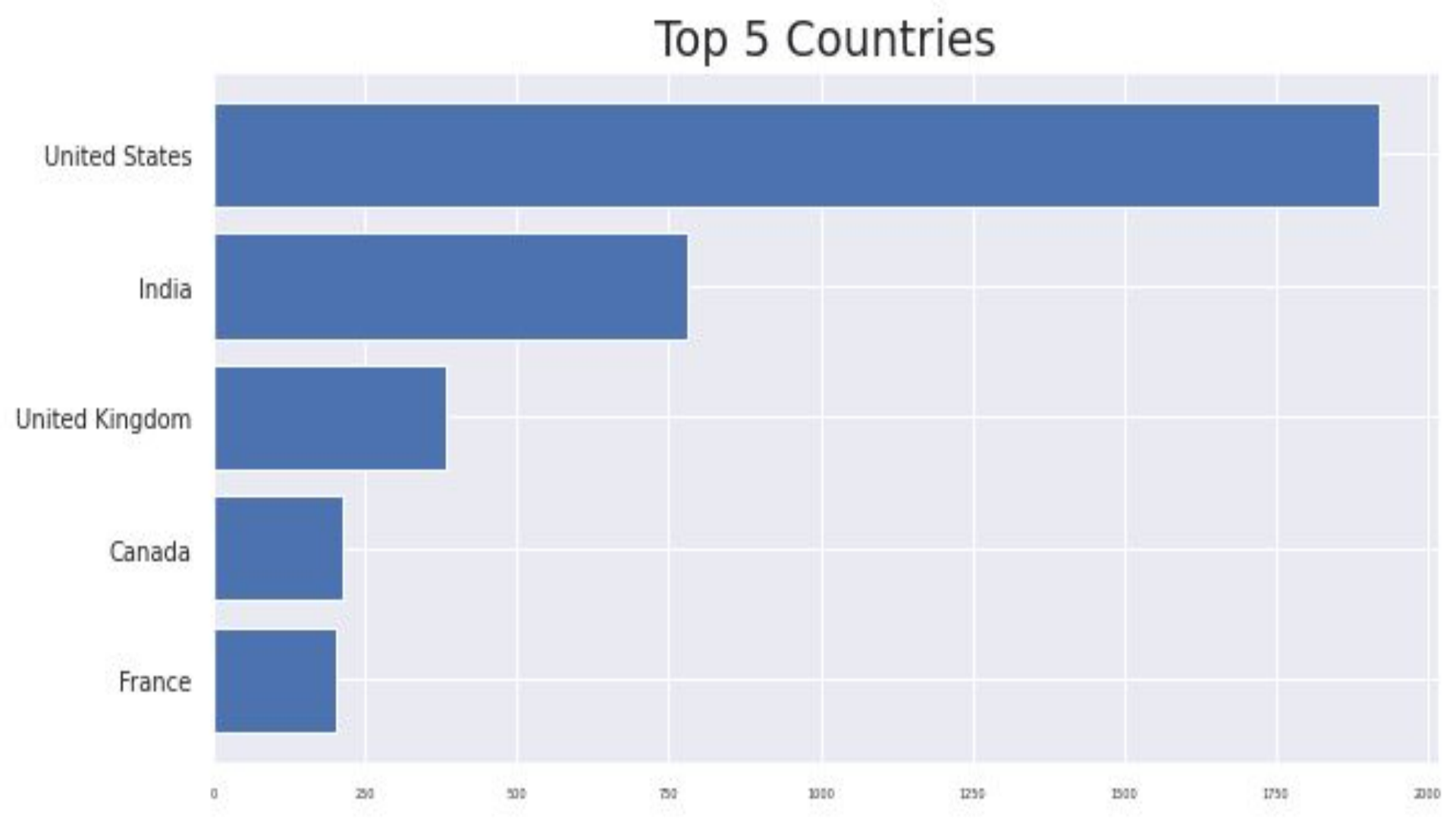
- Many different rating types that categorizes the suitable audiences for the Movies and TV Shows.
- TV-MA (‘Mature Audience only’) and TV-14 (‘May not be suitable for children under 14 years of age’) holds the majority.

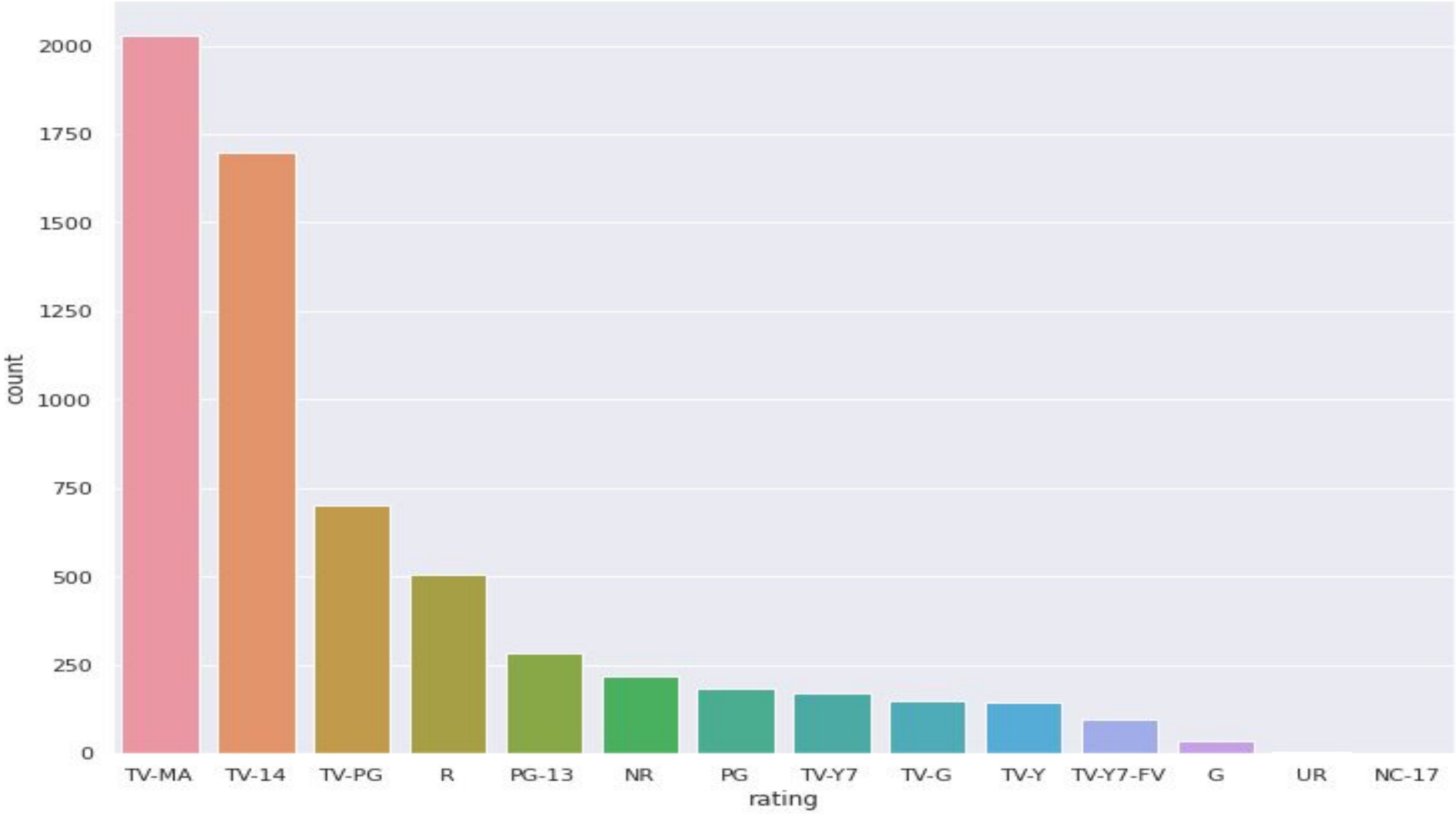


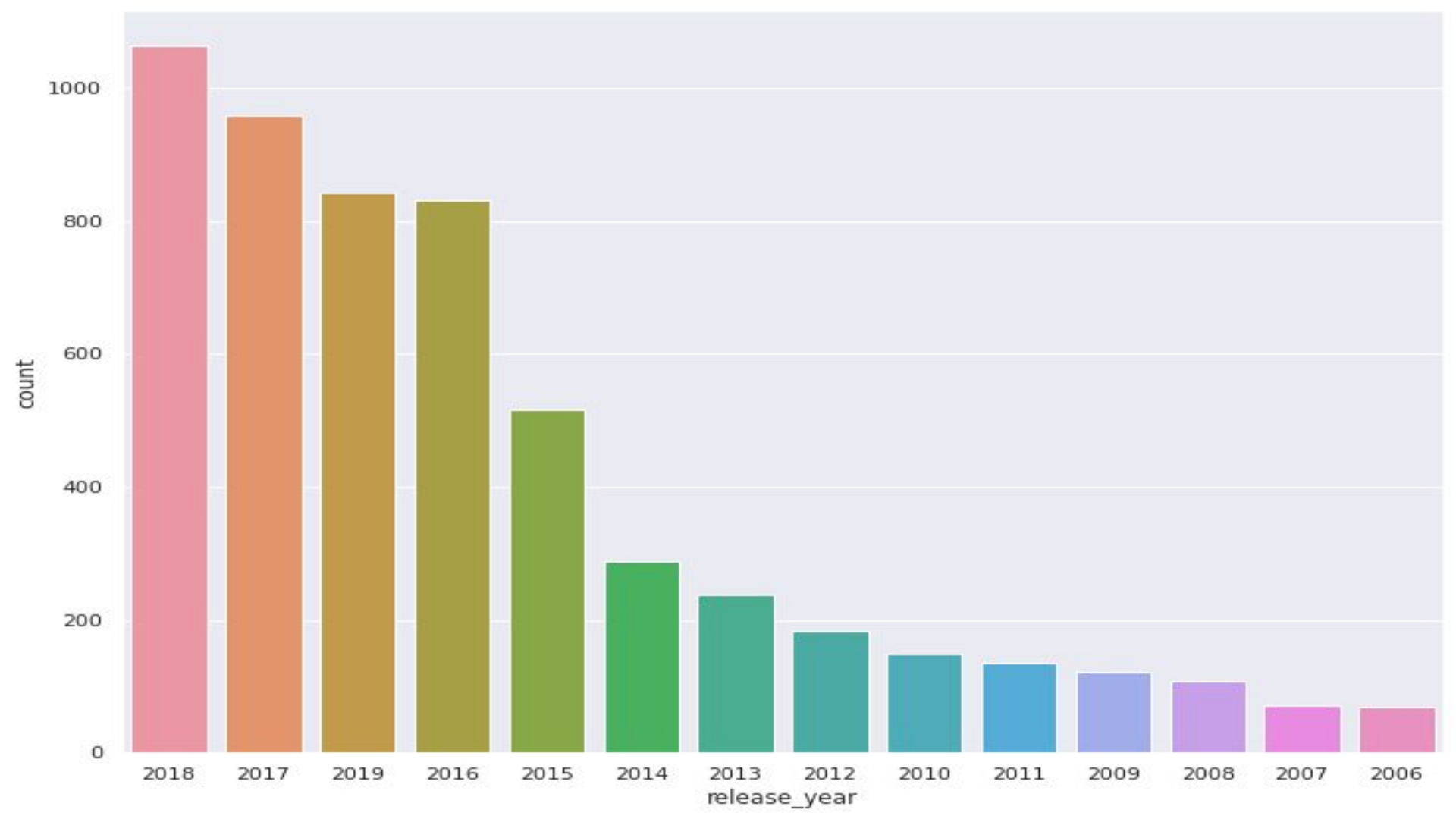
- Over the years, interests for the Movie content has increased more than the TV Shows have.
- This may lead the investors to focus on putting more money to the Movie industry.



- Important to see the leading countries of the film industry.
- Leading country is United States


Top 5 Countries
United States
India
United Kingdom
Canada
France
0
250
500
750
1000
1250
1500
1750
2000


2000
1750
1500
1250
1000
750
500
250
0
count
TV-MA
TV-14
TV-PG
R
PG-13
NR
PG
TV-Y7
TV-G
TV-Y
TV-Y7-FV
G
UR
NC-17
rating


1000
800
600
400
200
0
count
2018
2017
2019
2016
2015
2014
2013
2012
2010
2011
2009
2008
2007
2006
release_year

The IMDB Ratings

- We used omdbapi.com API to retrieve the IMDB scores of the titles.
- Main issue was one API key only allowed for 1000 requests per day
- Had to get different API keys with different mail addresses
- Re-ran the code with the retrieved API keys

```
df = pd.read_csv(join(path_prefix, fname))

title_list = df.title.T.values.tolist()
array_for_title = np.array(title_list)
print(array_for_title.reshape(6234,1))

print(" ")
year_list = df.release_year.T.values.tolist()
array_for_year = np.array(year_list)
print(array_for_year.reshape(6234,1))

import requests
endpoint = "http://www.omdbapi.com/?apikey=8e22b2aa&t=Norm%20of%20the%20North:%20King%20Sized%20Adventure&y=2019"
res = requests.get(endpoint)
print(res.json())

k = 0
for i in range( len(array_for_title)):
  show = array_for_title[i]
  year = array_for_year[i]
  key = "7055d37f"
  endpoint = "http://www.omdbapi.com/?apikey={}&t={}&y={}".format(key, show, year)
  res = requests.get(endpoint)
  # get the imdb rating from retrieved json object
  try:
    df["imdbRating"][i] = res.json()["imdbRating"]
    #rows.append({"imdbRating": res.json()["imdbRating"]})
  except:
    df["imdbRating"][i] = "NaN"
    #rows.append({"imdbRating": "NaN"})
  k+=1
  if k == 900:
    break

print(df)
compression_opts = dict(method='zip',
                        archive_name='out_final.csv')
df.to_csv('out_finallll.zip', index=False,
          compression=compression_opts)
!cp out_finallll.zip "drive/My Drive/"
```

Machine Learning

- There are two implementations of machine learning models which are Decision Tree and Random Forest Classifiers.
- Both models were classified according to the projects' IMDB score and its' release year, which then finds the director accordingly.

Decision Tree

- X dataset was given the IMDB scores and release years.
- Y dataset was given the directors' names.
- Then we had trained 80% of the data and then split the remaining data by half and created validation and test datasets.
- Later on Decision Tree was trained with our X_train and y_train datasets.

```
X_train, X_remaining, y_train, y_remaining = train_test_split(X, y, random_state=1, test_size=0.20)
X_val, X_test, y_val, y_test = train_test_split(X_remaining, y_remaining, random_state=1, test_size=0.50)

model = DecisionTreeClassifier(criterion="entropy")
model.fit(X_train, y_train)
```

Decision Tree

- We use the cross_val_score function to perform cross validation for Decision Tree Model.
- Then we checked the accuracy of the trained data and the test data.
- Accuracy of our trained data came out to be 63.98%, and the accuracy of our test data came out to be 48.12%.

```
[ ]  1 cross_val_score(model, X_train, y_train, scoring="accuracy")
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=5.
  % (min_groups, self.n_splits)), UserWarning)
array([0.58984375, 0.63671875, 0.62890625, 0.6015625 , 0.58984375])
```

```
[ ]  1 print("Decision Tree Train Accuracy:"+str(accuracy_score(y_train, model.predict(X_train))))
```

```
Decision Tree Train Accuracy:0.63984375
```

```
[ ]  1 dt_predictions = model.predict(X_test)
     2 accuracy_score(y_test, dt_predictions)
```

```
0.48125
```

Random Forest

- The same datasets that we used in Decision Tree Model were used in Random Forest Model as well, in order to check the difference in accuracies among both models.
- Then we checked the accuracy of the trained data and the test data.
- Accuracy of our trained data came out to be 63.98%, and the accuracy of our test data came out to be 48.12%. Which was the same results that was seen in the Decision Tree Model.

```
[ ]  1 model_rf = RandomForestClassifier(random_state=1, n_estimators=100)
     2 model_rf.fit(X_train, y_train)
     3 cross_val_score(model_rf, X_train, y_train, scoring="accuracy")
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=5.
  % (min_groups, self.n_splits)), UserWarning)
array([0.59375   , 0.640625  , 0.62890625, 0.61328125, 0.58984375])
```

```
[ ]  1 cross_val_score(model_rf, X_train, y_train, scoring="accuracy")
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=5.
  % (min_groups, self.n_splits)), UserWarning)
array([0.59375   , 0.640625  , 0.62890625, 0.61328125, 0.58984375])
```

```
[ ]  1 rf_predictions = model_rf.predict(X_test)
     2 rf_acc = accuracy_score(y_test, rf_predictions)
     3 print("Random Forest Train Accuracy:"+str(accuracy_score(y_train, model_rf.predict(X_train))))
     4 print("Random Forest Test Accuracy:"+str(rf_acc))
```

```
Random Forest Train Accuracy:0.63984375
Random Forest Test Accuracy:0.48125
```

Random Forest

- Then, Grid Search was used to find the best parameter for the dataset.
- After finding the parameters, we checked the accuracy of validation dataset and found it to be 59.37%.

```
param_grid = {
    'random_state': [32, 61],
    'n_estimators': [50, 75],
    'min_samples_split': [2, 4],
    'max_depth': [16, 32]
}

rf_predictions = RandomForestClassifier()

rf_grid = GridSearchCV(estimator=rf_predictions, param_grid=param_grid)
rf_grid.fit(X_train, y_train)
```

```
[ ]  1 rf_grid.best_params_
```

```
{'max_depth': 16,
 'min_samples_split': 2,
 'n_estimators': 75,
 'random_state': 61}
```

```
[ ]  1 grid_predictions = rf_grid.predict(X_val)
     2 accuracy_score(y_val, grid_predictions)
```

```
0.59375
```

Work Division

Suzan and Zeynep have worked on the data processing such as visualisation the data

Baran have worked on getting the IMDB score data via web scraping

Baran, Göktuğ and Alpaslan have worked on the Machine Learning process