Predicting the IMDb scores of the Netflix Movies and Series according to the genre and tags with the help of the NLP and ML algorithms

1.Introduction

Netflix Inc.(NFLX:NASDAQ) founded in 1997 by Marc Randolph and Reed Hastings as a rent-DVD-by-mail service that used the pay-for-rent model, and now turned into one of the most popular online entertainment platform.

As of 2022, there are 220 million Netflix subscribers, and half of the Americans prefer Netflix over any other video streaming service.

Netflix is heavily invested in developing machine learning research to maximize the user satisfaction and keep them hooked. And it can now set up 1,300 recommendation clusters, based on viewing preferences.

2.Data

The dataset is an artificial Kaggle dataset created by Ashish Gupta by using 4 APIs, includes 29 features, lastly updated in April 2021, has a license in public domain.

This dataset consists of one large cvs file and combines data sources from Netflix, Rotten Tomatoes, IMBD, posters, box office information, trailers on YouTube, and more. Note: There is no official Netflix API.

The dataset is great for curious intermediate level data scientists who wants to discover the correlations between different ratings, actors, directors, box office etc. It is also great for data scientists and engineers who wants to dive into deep learning areas like NLP, image processing etc.

3. Problem Statement

The goal of the project is to answer which tags & genres of series and movies streamed on Netflix achieve highest likeability/quality(IMDb score).

Note 1: IMDb is the world's most popular source for movie, TV and celebrity content.The answer of the question would be beneficial for Netflix to understand which categories of genre and tags are worth investing.

Note 2: IMDb score interpretation: If an IMDb score is 7.5, then it is within the top 25% of movies.

The original dataset has 29 columns and 15480 non-null entries.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15480 entries, 0 to 15479
Data columns (total 29 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Title                  15480 non-null  object 
 1   Genre                  13770 non-null  object 
 2   Tags                   15413 non-null  object 
 3   Languages              13545 non-null  object 
 4   Series or Movie        15480 non-null  object 
 5   Hidden Gem Score       13379 non-null  float64
 6   Country Availability   15461 non-null  object 
 7   Runtime                15479 non-null  object 
 8   Director               10772 non-null  object 
 9   Writer                 11150 non-null  object 
 10  Actors                 13555 non-null  object 
 11  View Rating            8456 non-null   object 
 12  IMDb Score             13381 non-null  float64
 13  Rotten Tomatoes Score  6382 non-null   float64
 14  Metacritic Score       4336 non-null   float64
 15  Awards Received        6075 non-null   float64
 16  Awards Nominated For   7661 non-null   float64
 17  Boxoffice              4007 non-null   object 
 18  Release Date           13373 non-null  object 
 19  Netflix Release Date   15480 non-null  object 
 20  Production House       5149 non-null   object 
 21  Netflix Link           15480 non-null  object 
 22  IMDb Link              13177 non-null  object 
 23  Summary                15471 non-null  object 
 24  IMDb Votes             13379 non-null  float64
 25  Image                  15480 non-null  object 
 26  Poster                 11842 non-null  object 
 27  TMDb Trailer           7194 non-null   object 
 28  Trailer Site           7194 non-null   object 
dtypes: float64(7), object(22)
memory usage: 3.4+ MB
```

The columns "Netflix Link", "IMDb Link", "Summary", "Image", "Poster", "TMDb Trailer", "Trailer Site" were found unnecessary for this analysis and eliminated from the dataframe.

As a rule of thumb, if a column has NAN values over %60-70, it should also be dropped. Following the investigation of the ratio of the NaN values, the columns "Metacritic Score" and "Boxoffice" got also eliminated.

The unique values in each remaining column are investigated. Tags are intended to describe the specific details about a movie/series that from time to time overlaps with genre or genre subcategories.

It turns out that the genre and tags have many subgroups. A movie or a series can be in multiple subgroups.

5.EDA

The necessary python libraries pandas, numpy, matplotlib, seaborn and plotly library of R were imported.

With descriptive statistics, the numerical features were investigated. Mean, standard deviation, min-max values as well as first quartile to third quantile were calculated.

```
#Descriptive statistics of the dataset
netflix.describe()
```

	Hidden Gem Score	IMDb Score	Rotten Tomatoes Score	Awards Received	Awards Nominated For	IMDb Votes
count	13379.000000	13381.000000	6382.000000	6075.000000	7661.000000	1.337900e+04
mean	5.937551	6.496054	59.523034	8.764444	13.983161	4.272841e+04
std	2.250202	1.146910	26.999173	18.311171	29.821052	1.257012e+05
min	0.600000	1.000000	0.000000	1.000000	1.000000	5.000000e+00
25%	3.800000	5.800000	38.000000	1.000000	2.000000	4.035000e+02
50%	6.800000	6.600000	64.000000	3.000000	5.000000	2.322000e+03
75%	7.900000	7.300000	83.000000	8.000000	12.000000	2.089050e+04
max	9.800000	9.700000	100.000000	300.000000	386.000000	2.354197e+06

For the next steps, I decided to create 2 seperate dataframes: one for TV Series, one for Movies.

One of the important steps of the EDA is examining the distribution of the numerical data and pointing out the outliers through visualization and mathematical approach. After removing the Nan values, the dataset becomes ready to check out the outliers.

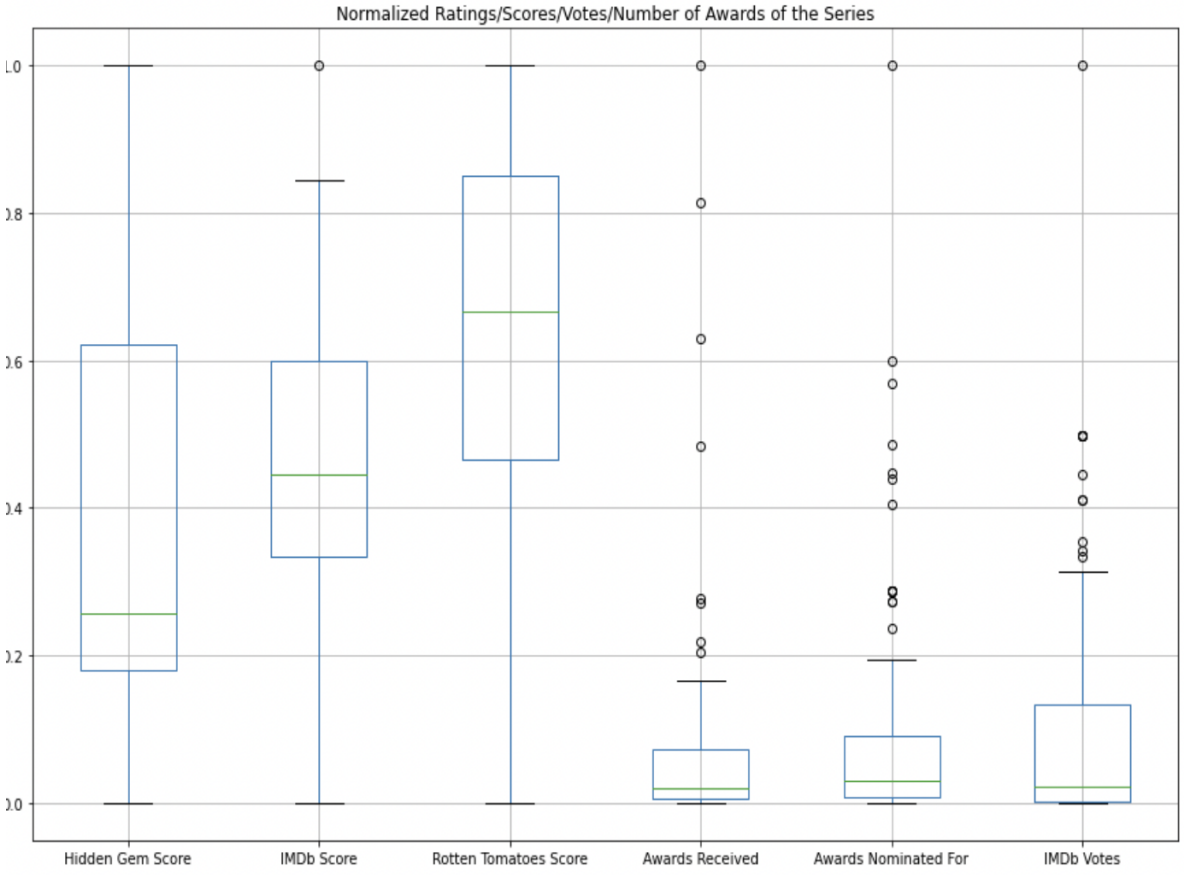
Note: Amputation methods like replacing the Nan values with mean, median, or mode would not work here since each movie's or series's rating is specific and defines its quality.

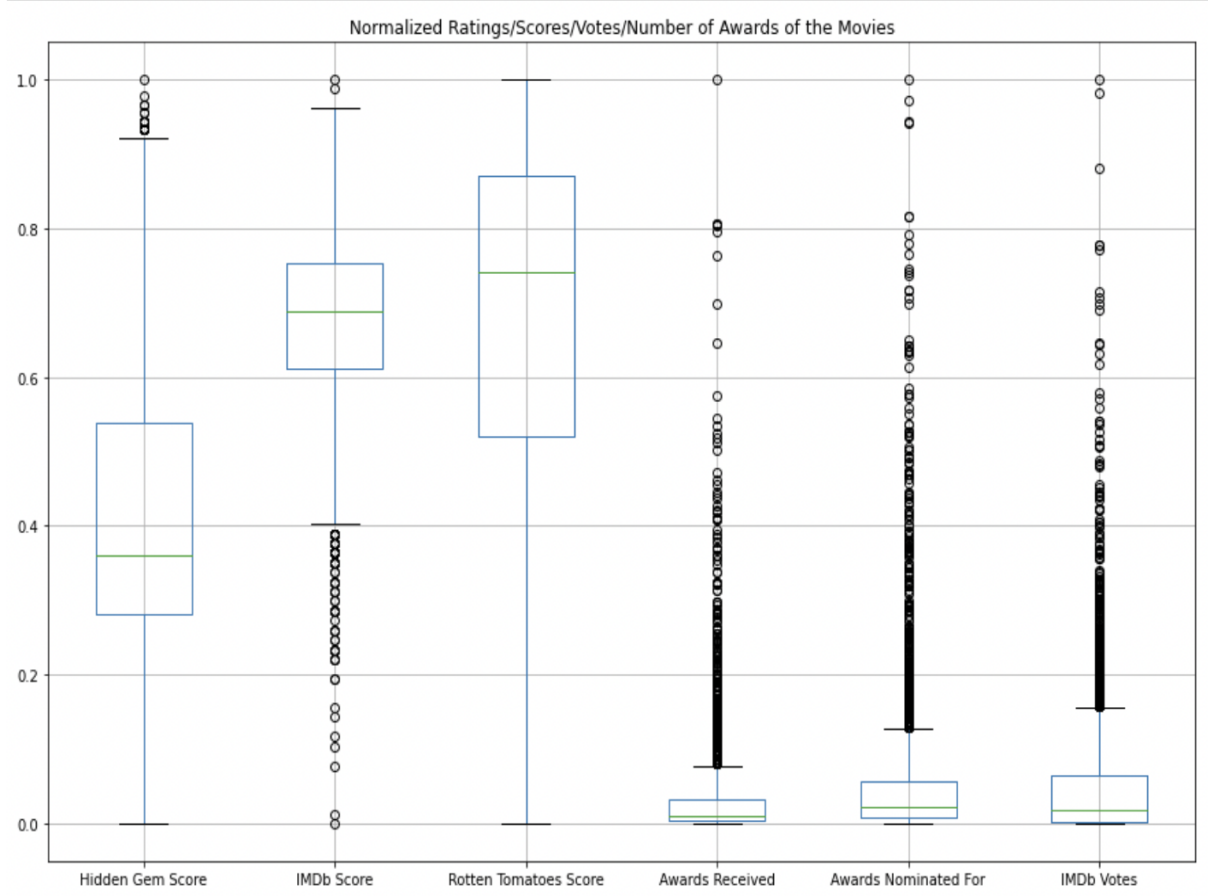
Boxplots can communicate a lot of information: min, max, outliers, percentiles, median...Z score describes the Taking a Z-score is simply mapping the data onto a distribution whose mean is defined as 0 and whose standard deviation is defined as 1.

The goal of taking Z-scores is to remove the effects of the location and scale of the data, allowing different datasets to be compared directly.

The intuition behind the Z-score method of outlier detection is that, once we've centred and rescaled the data, anything that is too far from zero (the threshold is usually a Z-score of 3 or -3) should be considered an outlier.

The data was rescaled through normalization. MinMaxScaler() from sklearn liibrary was used to transform all variables in the data to the same range. It did not solve the problem caused by outliers, so I dealt with the outliers as a next step.


Normalized Ratings/Scores/Votes/Number of Awards of the Series
1.0
0.8
0.6
0.4
0.2
0.0
Hidden Gem Score
IMDb Score
Rotten Tomatoes Score
Awards Received
Awards Nominated For
IMDb Votes


Normalized Ratings/Scores/Votes/Number of Awards of the Movies
1.0
0.8
0.6
0.4
0.2
0.0
Hidden Gem Score
IMDb Score
Rotten Tomatoes Score
Awards Received
Awards Nominated For
IMDb Votes

The presence of outliers in votes/awards/ratings actually makes sense. Good movies and series ratings/scores are way different than the mean ratings/scores, that is what makes them "good".

Heatmaps are an efficient way of demonstrating the correlation between two variables, one plotted on each axis.

By observing how cell colors change across each axis of the correlogram, you can observe if there are any patterns in value for one or both variables. The darker colors below represent higher correlation while lighter colors represent weaker correlation.

- The heatmap below shows the relationship among all the values of TV Series dataframe:

In TV Series, Hidden Gem Score and Rotten Tomatoes Score are positively correlated.

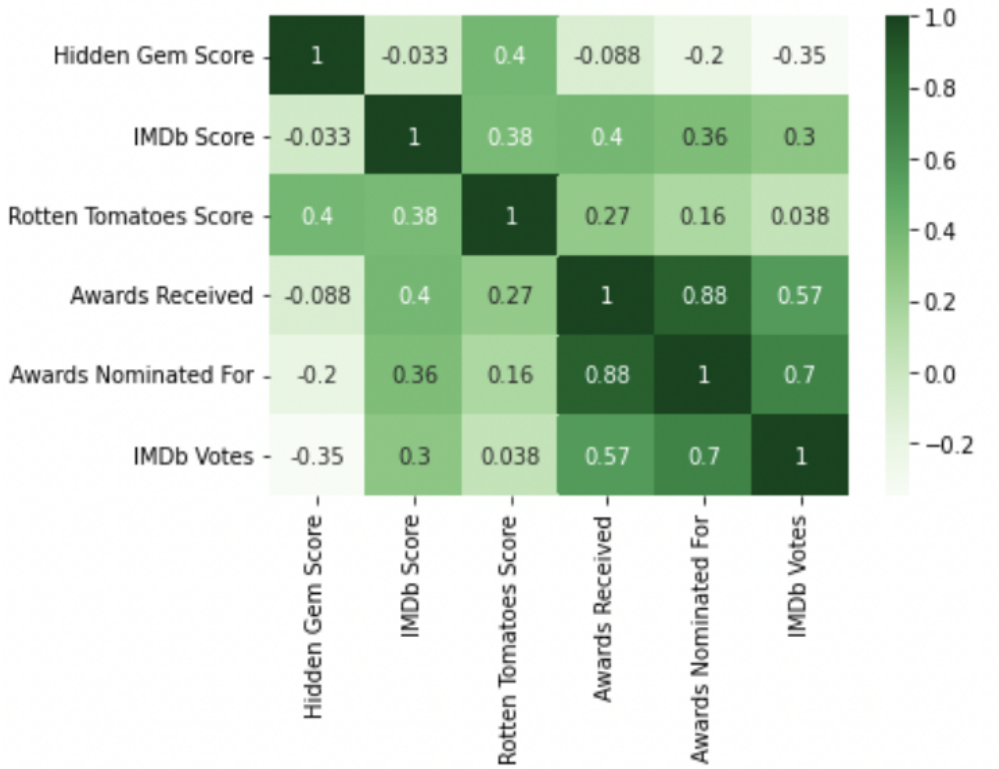
IMDb Score is strongly correlated with Rotten Tomatoes Score.

It is highly likely that a TV Series formerly nominated for award gets the award.

The series that were nominated and received award gets high IMDb score and IMDb votes.

Awards and nominations are not correlated with Hidden Gem Score.

The heatmap below shows the relationship among all the values of Movies dataframe:



In Movies, the Hidden Gem Score is positively correlated with Rotten Tomatoes Score.

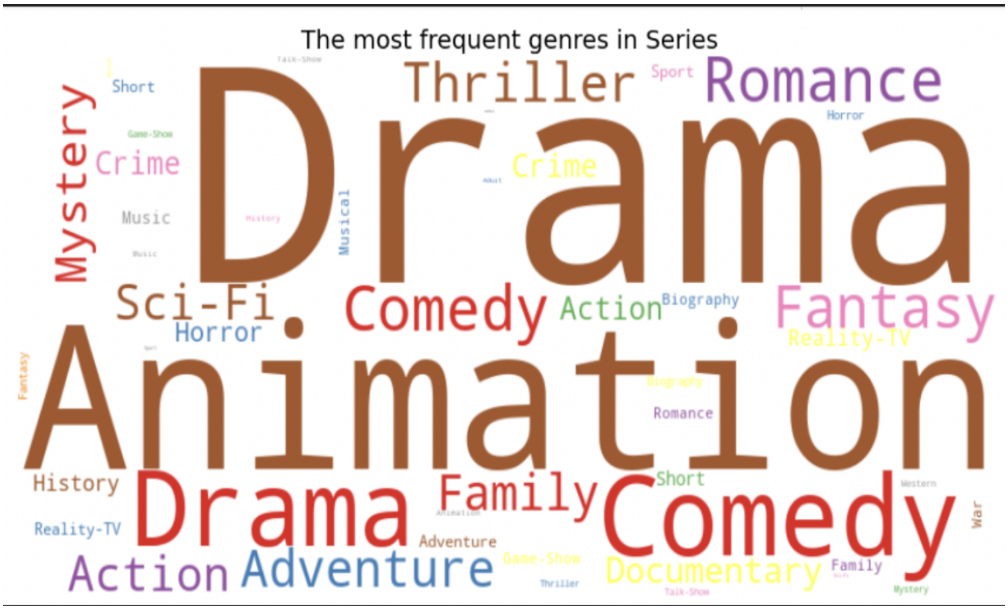
It is highly likely that a movie formerly nominated for award gets the award.

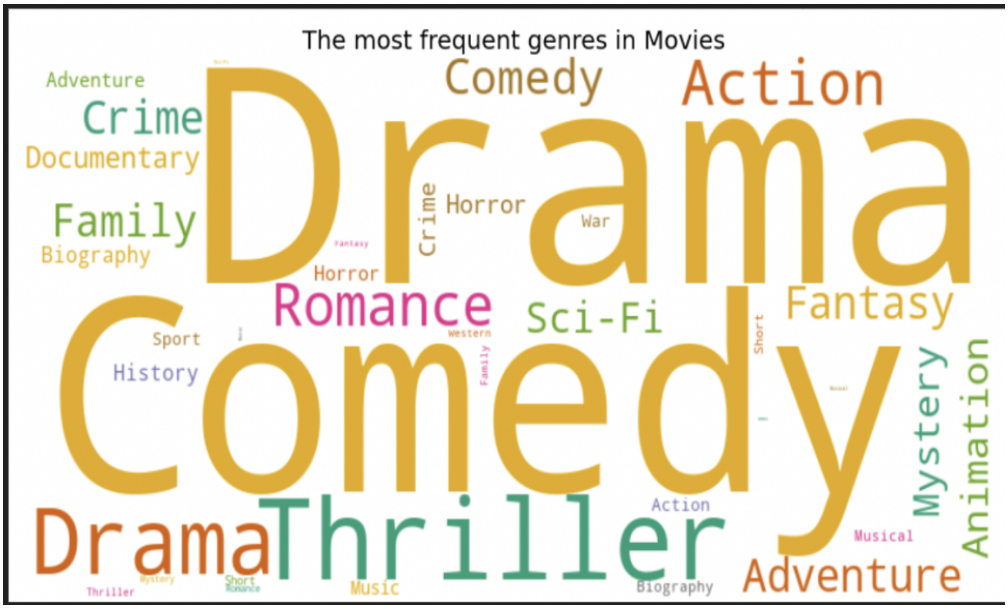
Nominated and rewarded movies gets high IMDb votes & Score.

Awards and nominations are not correlated with Hidden Gem Score and they are weakly positively correlated with Rotten Tomatoes Score.

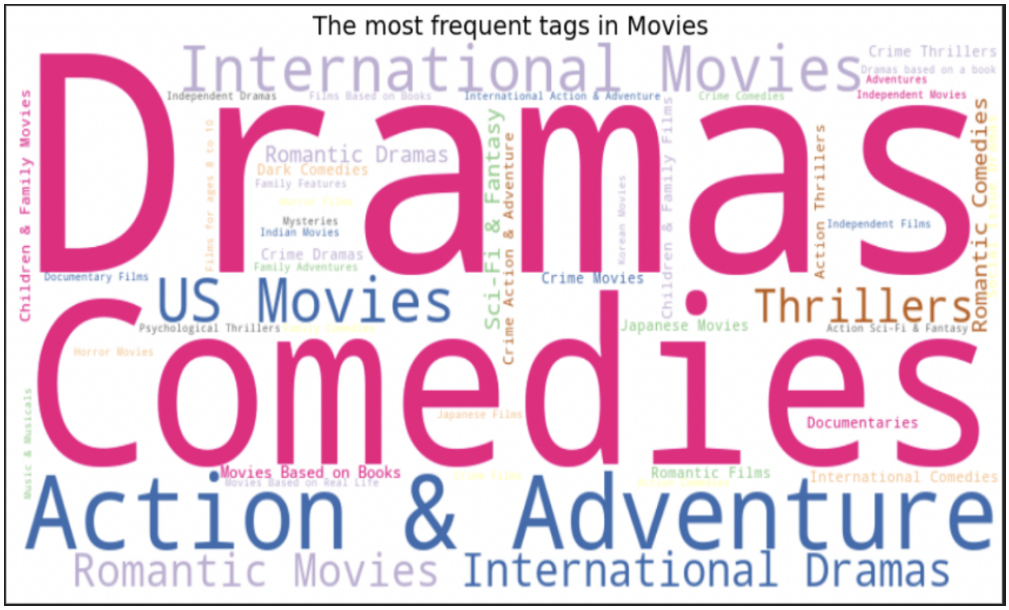
The next investigation is about finding the most frequently appearing genres and tags for both TV Series and Movies.

Wordcloud recently has become a very popular tool of data visualization. Although it might be insufficient for a deep analytical study, it is an easy and effective illustrative tool.It gives greater prominence to words that appear more frequently in a source text.


The most frequent genres in Series
Drama
Animation
Comedy
Drama
Thriller
Romance
Mystery
Crime
Crime
Sci-Fi
Comedy
Action
Fantasy
Horror
Reality-TV
Romance
History
Family
Comedy
Action
Adventure
Documentary
Short
Adventure
Musical
Music
Short
Sport
Horror
Game-Show
Biography
Western
War
Family
Mystery
Thriller
Talk-Show


The most frequent genres in Movies
Drama
Comedy
Drama
Thriller
Adventure
Crime
Documentary
Family
Biography
Comedy
Action
Horror
War
Crime
Horror
Romance
Sci-Fi
Fantasy
Sport
History
Mystery
Animation
Action
Musical
Music
Biography
Short
Western
Family
Short
Fantasy
Thriller
Mystery
Romance


The most frequent tags in Series
TV Dramas
TV Programmes
Anime Series
TV Comedies
Romantic TV Dramas
Crime TV Dramas
TV Cartoons
US TV Programmes
Docuseries
US TV Shows
Drama Programmes
Kids TV
TV Thrillers
TV Shows
Anime
Japanese TV Shows
Documentaries
Reality TV
Romantic TV Comedies
TV Shows Based on Books
Korean TV Shows
TV Action & Adventure
Romantic TV Shows
TV Mysteries
Period Pieces
Sitcoms
Japanese TV Programmes
Comedy Programmes
Sci-Fi & Fantasy Anime
TV Sci-Fi & Fantasy
Family Watch Together TVAnime Sci-Fi & Fantasy
British TV Shows
Asian TV Shows
Political TV Shows
Teen TV Shows
TV Shows Based on Comics
Action Anime
Japanese TV Films
British TV Programmes
TV Programmes Based on Books
Reality
Social Issue TV Dramas
Variety & Talk Shows
Crime TV Shows Based on Manga
School Anime



6.Feature Engineering & Pre-processing

The goal in this section is to use the Natural Language Processing(NLP) methods to prepare the data for machine learning modeling by feature extraction.

The necessary pyhton packages of sklearn library were used.

Text cleaning step of feature engineering varies from dataset to dataset. Here, the tags and genre categorizations were combined together. And then, strings were converted into lowercase and punctuation(comma) is removed. Then, the strings were splitted into "tokens".The stopwords, which were the words that were quite repetitive and have no meaning, got removed. One drawback of this dataset is that there are titles, genres and tags in multiple languages and the only stopwords that I removed are in English. Just like in English, there are stopwords in every language, and there are suitable packages to filter them out.

After tokenization, the next step was using the tf-idf vectorizer algorithm from scikit library.

A tf-idf vectorizer combines and performs two concepts: term frequency and inverse document frequency. A term frequency(tf) gives how many times a term has occured in a document(corpus). An inverse document frequency(idf) gives how significant a term is in a corpus. The mathematical formulas are like the following(Source:Medium link text):

IDF =Log[(# Number of documents) / (Number of documents containing the word)] and

TF = (Number of repetitions of word in a document) / (# of words in a document)

A tf-idf is the multiplication and normalizaiton of both values together.And the output looks as the following:

7.Modeling

7.1 Logistic Regression

Logistic Regression Model is considered as an efficient model for binary classification problems. Unlike linear regression, a linear relationship between dependent and independent variable is not necessary.

I used the logistic regression model to find out the probability of tags/genre categories end up as 0(%75 or unpopular) or 1(25% or popular).

7.1.1.Logistic Regression Classification Report

Classification Report for TV Series:

	precision	recall	f1-score	support
0	0.68	0.85	0.75	486
1	0.55	0.30	0.39	285
accuracy			0.65	771
macro avg	0.61	0.58	0.57	771
weighted avg	0.63	0.65	0.62	771

Classification Report for Movies:

	precision	recall	f1-score	support
0	0.88	1.00	0.94	2246
1	1.00	0.01	0.01	306
accuracy			0.88	2552
macro avg	0.94	0.50	0.47	2552
weighted avg	0.90	0.88	0.83	2552

Accuracy and precision are two measures of observational error. Accuracy is how close or far off a given set of measurements (observations or readings) are to their true value, while precision is how close or dispersed the measurements are to each other(Wikipedia).

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that were retrieved. Both precision and recall are therefore based on relevance(Wikipedia).

The f-1 score which is the harmonic mean of precision and recall. F1 tells if the classifier is actually good at identifying members of a class, or if it is just identifying everything as a member of a large class.

How to interpret these values?

The classification report for TV Series shows that the recall is higher than the precision when it comes to categorizing the 0's. High recall means that an algorithm returns most of the relevant results (whether or not irrelevant ones are also returned).A high precision relates to a low false positive rate, and high recall relates to a low false negative rate. There is lower false positives in classifying 0's than 1's.

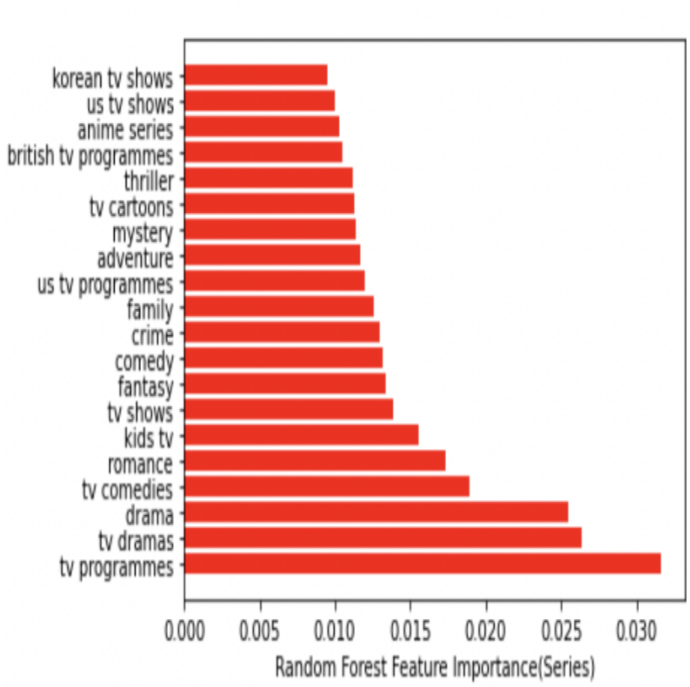
The classification report for Movies shows that the performance of the classification is ideal(1.0).

7.2 Random Forest Classifier

7.2.1. Random Forest Classifier Feature Importance Rankings & Model Evaluations

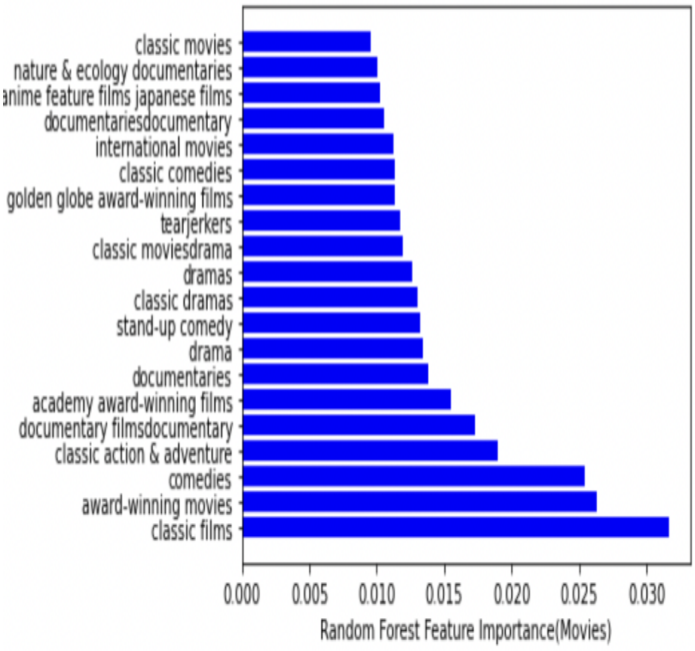
Ranking for TV Series:

	feature	score
1180	tv programmes	0.031638
1144	tv dramas	0.026343
7	drama	0.025464
1131	tv comedies	0.018932
18	romance	0.017303
604	kids tv	0.015544
1212	tv shows	0.013867
9	fantasy	0.013377
5	comedy	0.013180
6	crime	0.013009
8	family	0.012619
1292	us tv programmes	0.011944
1	adventure	0.011694
15	mystery	0.011357
1124	tv cartoons	0.011328
28	thriller	0.011211
172	british tv programmes	0.010516
111	anime series	0.010297
1300	us tv shows	0.010034
653	korean tv shows	0.009517



Ranking for Movies:

	feature	score
11799	classic films	0.008916
7083	award-winning movies	0.004707
12407	comedies	0.003704
11500	classic action & adventure	0.003480
21560	documentary filmsdocumentary	0.002951
885	academy award-winning films	0.002807
20934	documentaries	0.002794
302	drama	0.002673
60969	stand-up comedy	0.002628
11681	classic dramas	0.002362
21649	dramas	0.002182
12221	classic moviesdrama	0.002058
62407	tearjerkers	0.002021
31875	golden globe award-winning films	0.001992
11602	classic comedies	0.001882
38559	international movies	0.001847
21373	documentariesdocumentary	0.001799
5852	anime feature films japanese films	0.001771
48700	nature & ecology documentaries	0.001716
12054	classic movies	0.001695



7.2.2. Random Forest Classifier Confusion Matrix & Classification Report

Confusion Matrix & Classification Report for TV Series:

```
[[415  71]
 [191  94]]
              precision    recall  f1-score   support

           0       0.68      0.85      0.76       486
           1       0.57      0.33      0.42       285

    accuracy                           0.66       771
   macro avg       0.63      0.59      0.59       771
weighted avg       0.64      0.66      0.63       771

0.6601815823605707
```

Confusion Matrix & Classification Report for Movies:

```
[[2229   17]
 [ 282   24]]
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      2246
           1       0.59      0.08      0.14       306

    accuracy                           0.88      2552
   macro avg       0.74      0.54      0.54      2552
weighted avg       0.85      0.88      0.84      2552

0.8828369905956113
```

The classification report of Random Forest Classifier for TV Series highlights that the model correctly identifies 85% of the 0's and 33% of the 1's. The model predicts %68 at a time that a feature results in class 0, %57 at a time that a feature results in class 1.

The classification report of Random Forest Classifier for Moviesbrings out that the model correctly identifies 99% of 0's and 8% of 1's. This is the case due to the imbalance between the number of 0 and 1's.

Note that accuracy is not a good measure of classifier performance when the classes are so imbalanced. Also note that the precision and recall are inversely related.

8.Future Improvements & Conclusion

8.1. Improvements for Feature Engineering & Preprocessing

There is a lot of room for improvement:

1)Some features are very repetitive in the dataset. Some genre and tags are similar, so when tags and genres combined, their weight among the dataset increase dramatically.

2)The tf-idf vectorizer function can include max_df, min_df and max_features and stopwords of other languages.

8.2. Improvements for Modeling

Both logistic regression and random forest classifier are considered as efficient models in NLP problems.

While logistic regression tends to be more sensitive to overfitting and outliers, random forest model is more robust to overfitting and outliers. In this project, random forest classifier and logistic regression performed similarly.

A cleaner dataset would allow the ML models to learn the significant features and not biased toward certain repetitive features.

Defining a maximum depth of the trees, changing the number of estimators(trees), setting a limit to the numbeer of features at each node split could significantly improve the random forest model.

Doing cross-validation after each change would help me to identify which changes are improving the accuracy and overall performance of the model.