A Capstone Project Report on

TMDB Popularity Prediction

Submitted by

C.Kevin Patrick

## Abstract

The movie is one of the integral components of our everyday entertainment. The worldwide movie industry is one of the most growing and significant industries and seizing the attention of people of all ages. It has been observed in the recent study that only a few of the movies achieve success. Uncertainty in the sector has created immense pressure on the film production stakeholder. Moviemakers and researchers continuously feel it necessary to have some expert systems predicting the movie success probability preceding its production with reasonable accuracy. A maximum of the research work has been conducted to predict the movie popularity in the post-production stage. To help the movie maker estimate the upcoming film and make necessary changes, we need to conduct the prediction at the early stage of movie production and provide specific observations about the upcoming movie. Using movie rating and voting information of similar movies machine learning model to build a multiclass movie popularity prediction system. We also proposed a system to predict the popularity of the upcoming movie among different audience groups. We have divided the audience group into four age groups junior, teenage, mid-age and senior. This study has used publicly available Internet Movie Database (IMDb) data and The Movie Database (TMDb) data. We had implemented a multiclass classification model and achieved 96.8% accuracy, which outperforms all the benchmark models. This study highlights the potential of predictive and prescriptive data analytics in information systems to support industry decisions.

### Acknowledgements

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this group project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, we were fortunate to have Mr.Anbu Joel as my mentor. He has readily shared his immense knowledge in data analytics and guide me in a manner that the outcome resulted in enhancing our data skills.

I certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: August 02, 2022 Name:C.KevinPatrick

## Certificate of Completion

I hereby certify that the project titled “TMDB Popularity Prediction” was undertaken and completed (July 2022)

Mentor: Mr. Anbu Joel Date: August 2, 2022

# Table Of Contents

|  |  |  |
| --- | --- | --- |
| Chapter Number | Title | Page Number |
| 1. | Introduction |  |
| 2. | Data preparation and understanding |  |
| 3. | Fitting models to data |  |
| 4. | Analysis using Machine learning |  |
| 5. | Conclusion |  |
| 6. | Reference |  |

## CHAPTER 1: INTRODUCTION

The movie industry has grown immensely over the past few decades generating approximately10 billion of revenue for the stakeholders annually. Nowadays, people can stream Movies online at the comfort of being at their home with the help of Netflix, YouTube and downloads. A Movie’s gross revenue prediction is a very important problem in the film industry because it determines all the financial decisions made by producers and investors. Furthermore, a prediction system to assess the success of new movies with the help of the predicted revenue can help the movie producers and directors take proper decisions when making the movie in order to increase the chance of profitability and success. Usually, these types of predictions are made using basic statistical techniques. While these methods are pretty common, they often only provide a very rough and not a specific estimate of revenue and profitability prediction before a film has been released. The goal of this project is to analyze the data and find the best computational model through evaluating metrics for predicting revenues and profitability rates of a movie based on public data for movies extracted from a popular online movie database called The Movie Database (TMDb).

Various Websites other than TMDb are present out there providing Movie Trend Analysis and most of them fail to provide proper Movie analysis. This is mostly due to the presence of various kinds of noise in their data. To showcase a series of Data Wrangling techniques needed in general for cleaning, preparing and transforming a raw Movie Box Office data from the source to a Proper Data ready and suitable for Analytical Methods to be implemented on them. Analysis and then explore few predictive models and compare them for the sake of improving the accuracy of the prediction

## CHAPTER 2: DATA PREPARATION AND UNDERSTANDING

Pre Processing of data, Preparation of data, Prediction of data and Tools for data visualization. The steps are explained below briefly:

### Collection of Data

The data that is feasible for analysis in TMDB data set has been used and the prediction has been carried out for the same.

### Pre processing of data

The pre processing of data involves 3 steps namely data cleaning, feature selection and data transformation. Each step is explained below: Data transformation comprises of two explanatory variables which can be transformed from binomial form into binary form to be much applicable for the chosen models.

The data cleaning step involves missing data imputation or handling. Some of the chosen algorithms cannot manage missing data that is why missing value can be transformed by median, mean or zero. However, the replacement of missing data by computed value statistically is a better choice. The used set of data involves missing values in certain numerical variables and two categorical variables. Before training of model, feature selection is one of the most essential .

### Prediction of data

In this project I will attempt to classify the movie database based on given data. I make use of linear regression, ada boost regressor, and random forest to do the prediction.

## CHAPTER 3: FITTING MODELS TO DATA

### Linear Regression

### **Linear Regression** is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.

### Logistic Regression

Logistic regression is the proper model of regression analysis to utilize when the dependent variable is binary. Logistic regression is a predictive examination used to describe the relation between an independent variable set and dependent binary variable. For churn of customer logistic regression has been used to estimate the probability of churn as a function of customers characters or variables set (Sahu et al, 2018). According to Hassouna et al (2016) Logistic regression is also used to find the customer churn occurrence probability. Logistic regression is based on a mathematically oriented method to examine the impact of variables on others. Prediction is made by comprising a group of equations linking values of input with the output field.

### AdaBoost Regression

AdaBoost is one of the first boosting algorithms to be adapted in solving practices. Adaboost helps you **combine multiple “weak classifiers” into a single “strong classifier”**.

### Random Forest

We applied Random Forest on the Training data set to validate if any further improvement of the model can be performed post the linear regression. Below were the parameters which were applied for Random Forest

For the Random Forest model Randomized Search CV is used to optimize for several hyper parameters including n\_estimators, max\_features, max\_depth, criterion and bootstrap .Random Forest algorithm was also trained, we optimized the number of trees hyper parameter. We experimented with building the model by changing the values of this parameter every time in 100, 200, 300, 400 and 500 trees. Te best results show that the best number of trees was 200 trees. Increasing the number of trees after 200 will not give a signifcant increase in the performance. GBM algorithm was trained and tested on the same data, we optimized the number of trees hyper-parameter with values up to 500 trees. Te best value after the experiment was also 200 trees. GBM gave better results than RF and DT.

### XGBoostRegressor

### XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg:linear, and that for binary classification is reg:logistics.

### CODE SAMPLE

In [1]:

**import** pandas **as** pd

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

In [2]:

df **=** pd**.**read\_csv(r"D:\Imarticus\Project\TMDb\_Movie\_Popularity\_Analysis-main\tmdb-movies.c

In [3]:

df

Out[3]:

**id imdb\_id popularity budget revenue original\_title cast**

**0** 135397 tt0369610 32.985763 150000000 1513528810

Jurassic World

Chris Pratt|Bryce Dallas Howard|Irrfan

Khan|Vi...

http

**1** 76341 tt1392190 28.419936 150000000 378436354

Mad Max: Fury Road

Tom Hardy|Charlize Theron|Hugh Keays-

Byrne|Nic...

http:/

**2** 262500 tt2908446 13.112507 110000000 295238201 Insurgent

Shailene Woodley|Theo James|Kate Winslet|Ansel...

http://www.thediverg

**3** 140607 tt2488496 11.173104 200000000 2068178225

Star Wars: The Force Awakens

Harrison Ford|Mark

Hamill|Carrie Fisher|Adam D...

http://www.sta

Furious 7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **4** | 168259 | tt2820852 | 9.335014 | 190000000 | 1506249360 |
| **...** | ... | ... | ... | ... | ... |
| **10861** | 21 | tt0060371 | 0.080598 | 0 | 0 |

Vin Diesel|Paul Walker|Jason Statham|Michelle ...

... ...

The Endless Summer

Michael Hynson|Robert August|Lord 'Tally

Ho' B...

**10862** 20379 tt0060472 0.065543 0 0 Grand Prix

James Garner|Eva Marie Saint|Yves Montand|Tosh...

**id imdb\_id popularity budget revenue original\_title cast**

**10863** 39768 tt0060161 0.065141 0 0

Beregis Avtomobilya

Innokentiy Smoktunovskiy|Oleg Efremov|Georgi Z...

**10864** 21449 tt0061177 0.064317 0 0 What's Up,

Tiger Lily?

Tatsuya Mihashi|Akiko Wakabayashi|Mie

Hama|Joh...

**10865** 22293 tt0060666 0.035919 19000 0

Manos: The Hands of

Fate

Harold P. Warren|Tom Neyman|John Reynolds|Dian...

# 10866 rows × 21 columns

In [4]:

Out[4]:

In [5]:

df**.**isnull()**.**sum()

id 0

imdb\_id 10

popularity 0

budget 0

revenue 0

original\_title 0

cast 76

homepage 7930

director 44

tagline 2824

keywords 1493

overview 4

runtime 0

genres 23

production\_companies 1030

release\_date 0

vote\_count 0

vote\_average 0

release\_year 0

budget\_adj 0

revenue\_adj 0

dtype: int64

df **=** df[df**.**genres**.**isnull()**==False**]

In [6]:

df **=** df[df**.**production\_companies**.**isnull()**==False**]

In [7]:

df **=** df**.**drop(['id','imdb\_id','original\_title','homepage','tagline','keywords','overview',

In [8]:

df **=** df**.**drop(['cast','production\_companies'],axis **=** 1)

In [42]:

plt**.**boxplot(df**.**popularity)

Out[42]:

In [10]:

con **=** []

**for** i **in** range(0,len(df)): p**=**df**.**genres**.**iloc[i]

p **=** str(p)

t **=** p**.**split('|')

**for** j **in** t:

**if** j **in** con:

**continue else**:

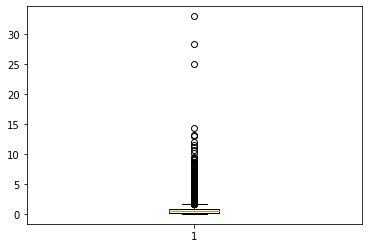
con**.**append(j)

{'whiskers': [<matplotlib.lines.Line2D at 0x172ec1c4c70>,

<matplotlib.lines.Line2D at 0x172ec1da100>], 'caps': [<matplotlib.lines.Line2D at 0x172ec1da370>,

<matplotlib.lines.Line2D at 0x172ec1da7c0>],

'boxes': [<matplotlib.lines.Line2D at 0x172ec1c49a0>], 'medians': [<matplotlib.lines.Line2D at 0x172ec1daac0>], 'fliers': [<matplotlib.lines.Line2D at 0x172ec1daf10>], 'means': []}



In [11]:

df[con] **=** 0

In [12]:

df**.**info()

<class 'pandas.core.frame.DataFrame'> Int64Index: 9827 entries, 0 to 10865 Data columns (total 31 columns):

# Column Non-Null Count Dtype

1. popularity 9827 non-null float64
2. budget 9827 non-null int64
3. revenue 9827 non-null int64
4. director 9807 non-null object
5. runtime 9827 non-null int64
6. genres 9827 non-null object
7. vote\_count 9827 non-null int64
8. vote\_average 9827 non-null float64
9. release\_year 9827 non-null int64
10. budget\_adj 9827 non-null float64
11. revenue\_adj 9827 non-null float64
12. Action 9827 non-null int64
13. Adventure 9827 non-null int64
14. Science Fiction 9827 non-null int64
15. Thriller 9827 non-null int64
16. Fantasy 9827 non-null int64

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 16 | Crime | 9827 | non-null | int64 |
| 17 | Western | 9827 | non-null | int64 |
| 18 | Drama | 9827 | non-null | int64 |
| 19 | Family | 9827 | non-null | int64 |
| 20 | Animation | 9827 | non-null | int64 |
| 21 | Comedy | 9827 | non-null | int64 |
| 22 | Mystery | 9827 | non-null | int64 |
| 23 | Romance | 9827 | non-null | int64 |
| 24 | War | 9827 | non-null | int64 |
| 25 | History | 9827 | non-null | int64 |
| 26 | Music | 9827 | non-null | int64 |
| 27 | Horror | 9827 | non-null | int64 |
| 28 | Documentary | 9827 | non-null | int64 |
| 29 | TV Movie | 9827 | non-null | int64 |
| 30 | Foreign | 9827 | non-null | int64 |

dtypes: float64(4), int64(25), object(2) memory usage: 2.4+ MB

In [13]:

**for** i **in** range(0,len(df)): p**=**df**.**genres**.**iloc[i]

p **=** str(p)

t **=** p**.**split('|')

**for** k **in** t:

**if** k **!=** '':

df[k]**.**iloc[i] **=** 1

C:\Users\kevpa\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWithCopyWa rning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_gu ide/indexing.html#returning-a-view-versus-a-copy

self.\_setitem\_single\_block(indexer, value, name)

In [14]:

df

Out[14]:

**popularity budget revenue director runtime genres vote\_count vote\_averag**

**0** 32.985763 150000000 1513528810 Colin

Trevorrow

124 Action|Adventure|Science

Fiction|Thriller

|  |  |
| --- | --- |
| 5562 | 6. |
| 6185 | 7. |
| 2480 | 6. |

**1** 28.419936 150000000 378436354 George Miller 120 Action|Adventure|Science

Fiction|Thriller

**2** 13.112507 110000000 295238201 Robert

Schwentke

119 Adventure|Science Fiction|Thriller

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **3** | 11.173104 | 200000000 | 2068178225 | J.J. Abrams | 136 | Action|Adventure|Science  Fiction|Fantasy | 5292 | 7. |
| **4** | 9.335014 | 190000000 | 1506249360 | James Wan | 137 | Action|Crime|Thriller | 2947 | 7. |
| **...** | ... | ... | ... | ... | ... | ... | ... |  |
| **10861** | 0.080598 | 0 | 0 | Bruce Brown | 95 | Documentary | 11 | 7. |
| **10862** | 0.065543 | 0 | 0 | John Frankenheimer | 176 | Action|Adventure|Drama | 20 | 5. |
| **10863** | 0.065141 | 0 | 0 | Eldar Ryazanov | 94 | Mystery|Comedy | 11 | 6. |
| **10864** | 0.064317 | 0 | 0 | Woody Allen | 80 | Action|Comedy | 22 | 5. |

**popularity budget revenue director runtime genres vote\_count vote\_averag**

**10865** 0.035919 19000 0 Harold P.

Warren

74 Horror 15 1.

# 9827 rows × 31 columns

In [16]:

Out[16]:

df['popularity']**.**index

Int64Index([ 0, 1, 2, 3, 4, 5, 6, 7, 8,

9,

...

10856, 10857, 10858, 10859, 10860, 10861, 10862, 10863, 10864,

10865],

dtype='int64', length=9827)

In [17]:

sns**.**scatterplot(df["popularity"]**.**index,df["popularity"])

Out[17]:

In [18]:

Out[18]:

In [19]:

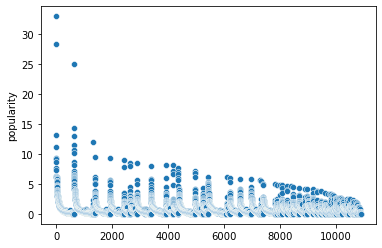
Out[19]:

In [20]:

C:\Users\kevpa\anaconda3\lib\site-packages\seaborn\\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positiona l argument will be `data`, and passing other arguments without an explicit keyword will re sult in an error or misinterpretation.

warnings.warn(

<AxesSubplot:ylabel='popularity'>



df["vote\_count"]**.**max()

9767

df**.**columns

Index(['popularity', 'budget', 'revenue', 'director', 'runtime', 'genres', 'vote\_count', 'vote\_average', 'release\_year', 'budget\_adj', 'revenue\_adj', 'Action', 'Adventure', 'Science Fiction', 'Thriller', 'Fantasy', 'Crime', 'Western', 'Drama', 'Family', 'Animation', 'Comedy',

'Mystery', 'Romance', 'War', 'History', 'Music', 'Horror', 'Documentary', 'TV Movie', 'Foreign'],

dtype='object')

col **=** df**.**columns

In [21]:

Out[21]:

|  |  |  |
| --- | --- | --- |
| 1 | 1 |  |
| 2 | 1 |  |
| 3 | 0 |  |
| 4 | 1 |  |
|  | .. |  |
| 10861 | 0 |  |
| 10862 | 0 |  |
| 10863 | 0 |  |
| 10864 | 0 |  |
| 10865 | 0 |  |
| Name: | Thriller, Length: 9827, dtype: | int64 |

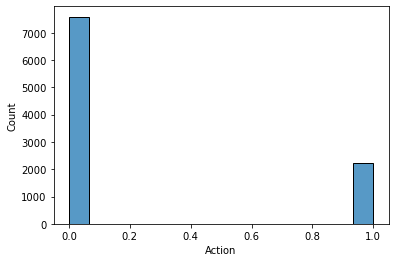
In [22]:

df**.**iloc[0:,14] 0 1

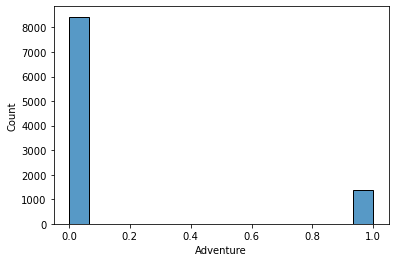
Action

**for** i **in** range(11,29): sns**.**histplot(df**.**iloc[0:,i]) print(col[i])

plt**.**show()

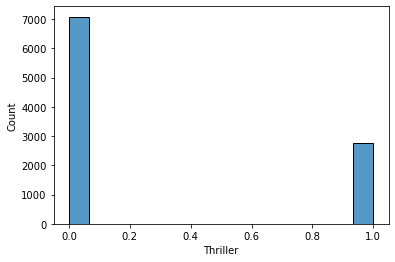


Adventure

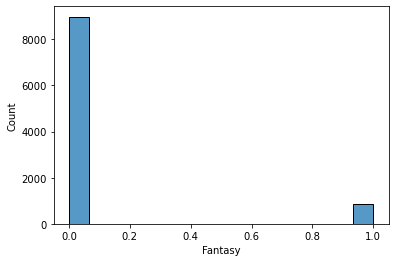


Science Fiction

Thriller

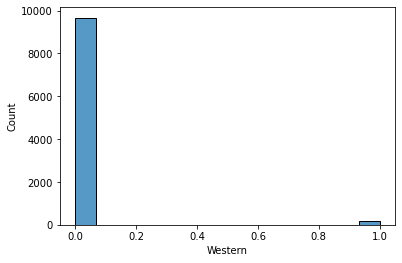


Fantasy

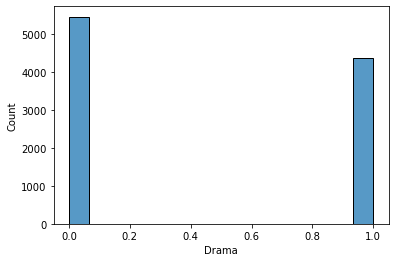


Crime

Western

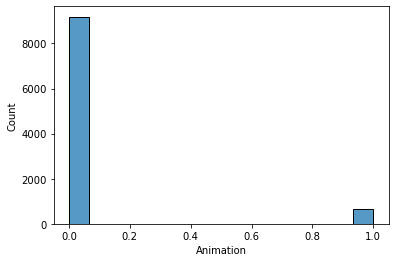


Drama

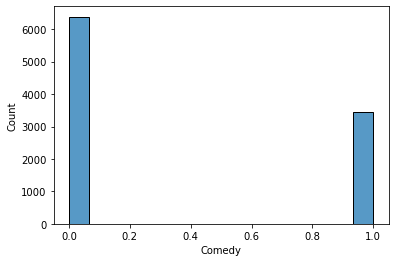


Family

Animation

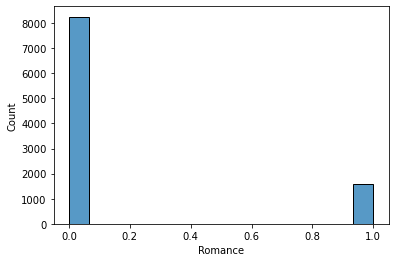


Comedy

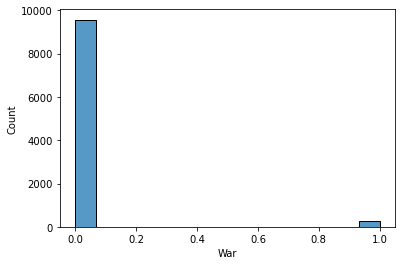


Mystery

Romance

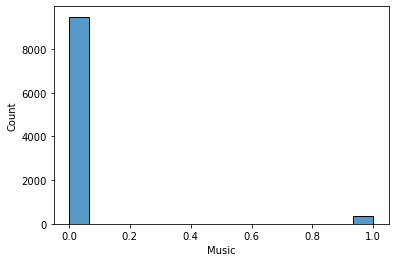


War

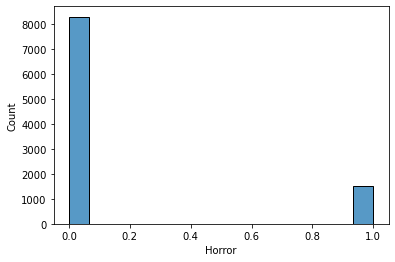


History

Music



Horror



Documentary

In [23]:

len(df**.**columns)

Out[23]: 31

In [24]:

df **=** df**.**drop(['genres'],axis **=** 1)

In [25]:

**from** sklearn.preprocessing **import** LabelEncoder

In [26]:

df['director'] **=** LabelEncoder**.**fit\_transform(df,df['director'])

C:\Users\kevpa\anaconda3\lib\site-packages\sklearn\preprocessing\\_label.py:117: UserWarnin g: Pandas doesn't allow columns to be created via a new attribute name - see https://panda s.pydata.org/pandas-docs/stable/indexing.html#attribute-access

self.classes\_, y = \_unique(y, return\_inverse=True)

In [27]:

df **=** df**.**drop('release\_year',axis**=**1)

In [28]:

**from** sklearn.model\_selection **import** train\_test\_split, cross\_val\_score

X **=** df**.**drop('popularity',axis**=**1) y **=** df['popularity']

In [29]:

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.2, random\_state**=**0)

In [30]:

**from** sklearn.ensemble **import** RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegr

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.svm **import** SVR

**from** xgboost **import** XGBRegressor

**from** sklearn.tree **import** DecisionTreeRegressor

In [31]:

linreg **=** LinearRegression() linreg**.**fit(X\_train,y\_train)

Out[31]:

In [32]:

LinearRegression()

Out[32]:

In [33]:

xgb **=** XGBRegressor(n\_estimators**=**50) xgb**.**fit(X\_train,y\_train)

linreg**.**score(X\_test,y\_test) 0.7348247436783566

Out[33]:

In [34]:

Out[34]:

In [35]:

ada **=** AdaBoostRegressor(random\_state**=**0, n\_estimators**=**100)**.**fit(X\_train,y\_train)

XGBRegressor(base\_score=0.5, booster='gbtree', callbacks=None, colsample\_bylevel=1, colsample\_bynode=1, colsample\_bytree=1, early\_stopping\_rounds=None, enable\_categorical=False, eval\_metric=None, gamma=0, gpu\_id=-1, grow\_policy='depthwise', importance\_type=None, interaction\_constraints='', learning\_rate=0.300000012, max\_bin=256, max\_cat\_to\_onehot=4, max\_delta\_step=0, max\_depth=6, max\_leaves=0, min\_child\_weight=1, missing=nan, monotone\_constraints='()', n\_estimators=50, n\_jobs=0, num\_parallel\_tree=1, predictor='auto', random\_state=0, reg\_alpha=0, reg\_lambda=1, ...)

xgb**.**score(X\_test,y\_test)

0.5836697781105052

In [36]:

Out[36]:

In [37]:

rf **=** RandomForestRegressor(n\_estimators**=**1000, random\_state**=**0) rf**.**fit(X\_train,y\_train)

0.6363811855675683

ada**.**score(X\_test,y\_test)

Out[37]:

In [38]:

Out[38]:

In [ ]:

In [ ]:

In [ ]:

RandomForestRegressor(n\_estimators=1000, random\_state=0)

rf**.**score(X\_test,y\_test)

0.7598590839467816

CONCLUSION

The dataset isn't a collection of clean data points. I dropped the NULL values because their size doesn't drastically affect the dataset but dropping the zero valued data points of budget and revenue affect the dataset. Dropping them means losing half of the data points collected. Instead, I filtered out the data points with zero values for both budget and revenue columns and filled them with the mean. To make the distribution easy, I computed the mean of budget and revenue columns for each release\_year, and filled the data points with their respective release\_year. The data point with either the budget or revenue values of zeros got dropped. In the budget and revenue part, I found out that there is a huge difference in budgeting between flopped and successful movies. I also found out that most of the movies with high profit as compared to their budget couldn't make it to the successful movies. I analysed the correlation between profit success and different factors and I found out profit is dependent 6% on top directors, 14% on top star casts, 54% on top production companies, and 48% on the popularity of the genre. However, this analysis does provide a lot of insight into what features are characteristic of existing positions. This paper details only a small portion of the information about positions that can be inferred from this analysis.

Reference

1.TMDB movie analysis – <https://github.com/Sumaya52/TMDb_Movie_Popularity_Analysis/blob/main/TMDb_popularity_analysis.ipynb>

2.Kaggle reference – <https://www.kaggle.com/code/dhainiksuthar/movie-box-office-prediction>