The Movie Database(TMDb) Analysis and Comparison of Predictive Models

A Preprocessing, Analytical and Modeling Case Study using Supervised ML Models

Jack Cox

Introduction: Motivation and Aim

- The movie industry has grown immensely over the past few decades generating approximately $10 billion of revenue for the stakeholders annually. 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.
- The goal of this project is to analyze the data and compare models to check the best one through evaluating metrics based on public data for movies extracted from a popular online movie database called The Movie Database (TMDb).

Table of Contents

01 Data Preprocessing

Defining Target Variables

- Create a new column in df called profitable, defined as 1 if the movie revenue is greater than the movie budget, and 0 otherwise.¶
- Define and store the outcomes we will use for regression and classification as such -- regression_target as the string 'revenue' & classification_target as the string 'profitable'

```
regression_target = 'revenue'
classification_target = 'profitable'
```

```
# Creating the 'profitable' column
df['profitable'] = df.revenue > df.budget
df['profitable'] = df['profitable'].astype(int)
```

```
# Printing the types of Value and their Count in the 'profitable' column
df["profitable"].value_counts()
```

```
1    2585
0    2218
Name: profitable, dtype: int64
```

Removing null, unwanted or infinite values

- Proceed by analyzing only the rows without any missing data. Replace the null values of all unimportant data to reduce data loss and then remove rows with any infinite or missing values.¶

```
In [16]: 1 df.isnull().sum()

Out[16]: budget              0
         genres              0
         id                  0
         keywords            0
         original_language   0
         overview            0
         popularity          0
         release_date        0
         revenue             0
         runtime             0
         status              0
         tagline             0
         title               0
         vote_average        0
         vote_count          0
         movie_id            0
         cast                0
         profitable          0
         return              0
         dtype: int64
```

Feature Engineering – Encoding Genre Column

- Add indicator columns for each genre.
- Determine all the genres in the genre column and make use of the strip() function on each genre to remove trailing characters
- Include each listed genre as a new column in the dataframe. Each element of these genre columns is 1 if the movie belongs to that particular genre, and 0 otherwise, keeping in mind, a movie may belong to several genres at once

	Action	Adventure	Fantasy	Science Fiction	Crime	Drama	Thriller	Animation	Family	Western	Comedy	Romance	Horror
0	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	1	0	0	0	0	0	0	0	0
3	1	0	0	0	1	1	1	0	0	0	0	0	0
4	1	1	0	1	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4788	0	0	0	0	1	0	0	0	0	0	1	0	1
4791	0	0	0	0	0	0	0	0	0	0	0	0	1
4792	0	0	0	0	1	0	1	0	0	0	0	0	1
4796	0	0	0	1	0	1	1	0	0	0	0	0	0
4798	1	0	0	0	1	0	1	0	0	0	0	0	0

Feature Engineering – Create Year Column

- Create a new column, year -- the year of production of a movie from the given format of release_date is extracted and stored

```
1 df['release_date'].head()
```

```
0    2009-12-10
1    2007-05-19
2    2015-10-26
3    2012-07-16
4    2012-03-07
Name: release_date, dtype: object
```

```
1 df['year'] = pd.to_datetime(df['release_date'], errors='coerce').apply(lambda x: str(x).split('-')[0])
```

```
1 df['year'].isnull().sum()
```

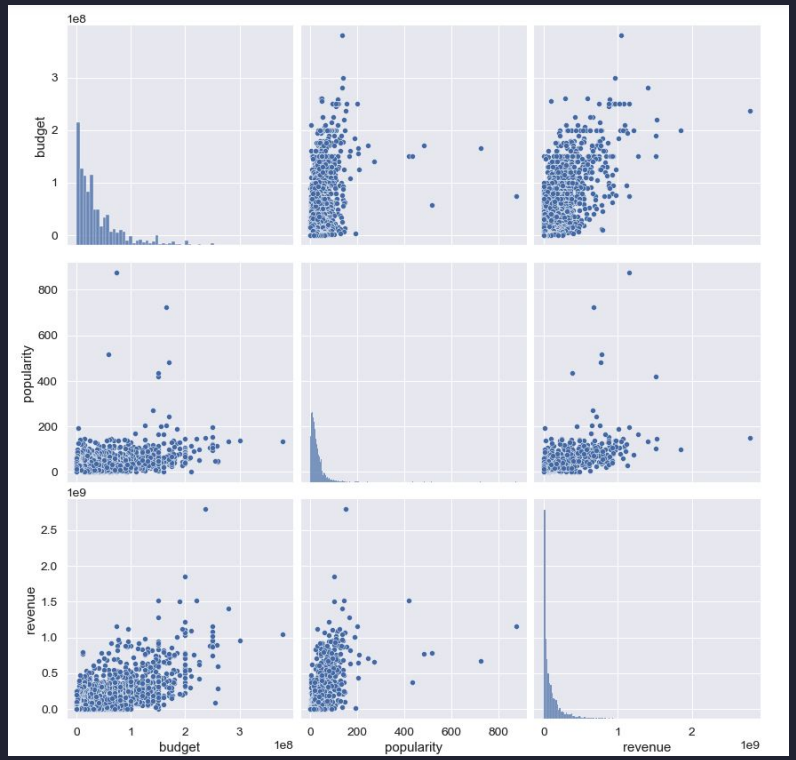
```
0
```

Feature Selection & Feature Transformation

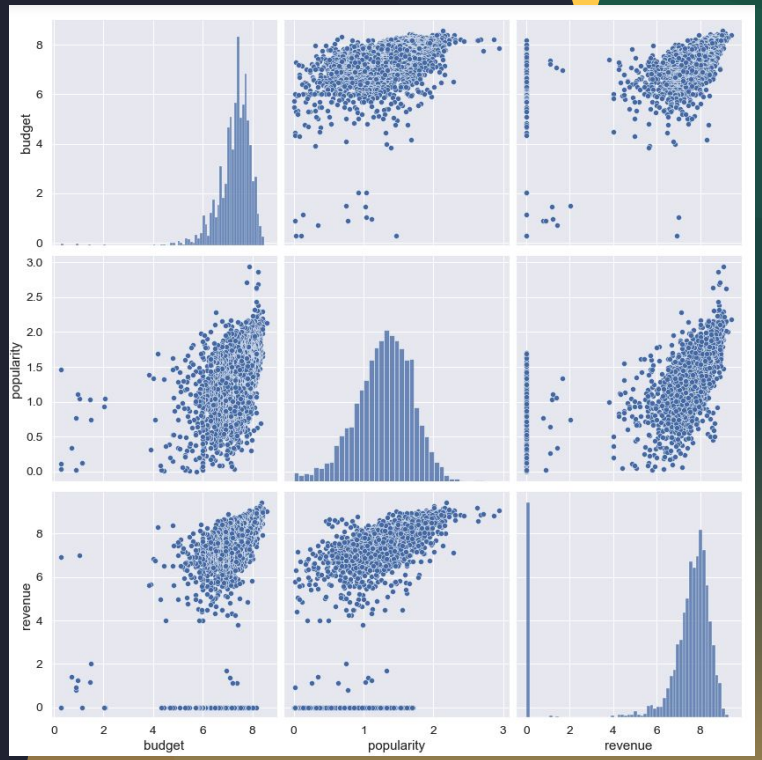
- Some variables in the dataset are already numeric and useful for regression and classification. Store the names of these variables for future use. Also view some of the continuous variables and outcomes by plotting each pair in a scatter plot and evaluate the skew of each variable.
- After selection of the variables and skew evaluation there will be a need to transform these variables to eliminate this skewness, using the np.log10() method. Because some of these variable values are exactly 0, a small positive value will be added to each to ensure it is defined; this is necessary because log(0) is negative infinity.

Feature Selection & Feature Transformation

Pre-skew elimination



After eliminating skew

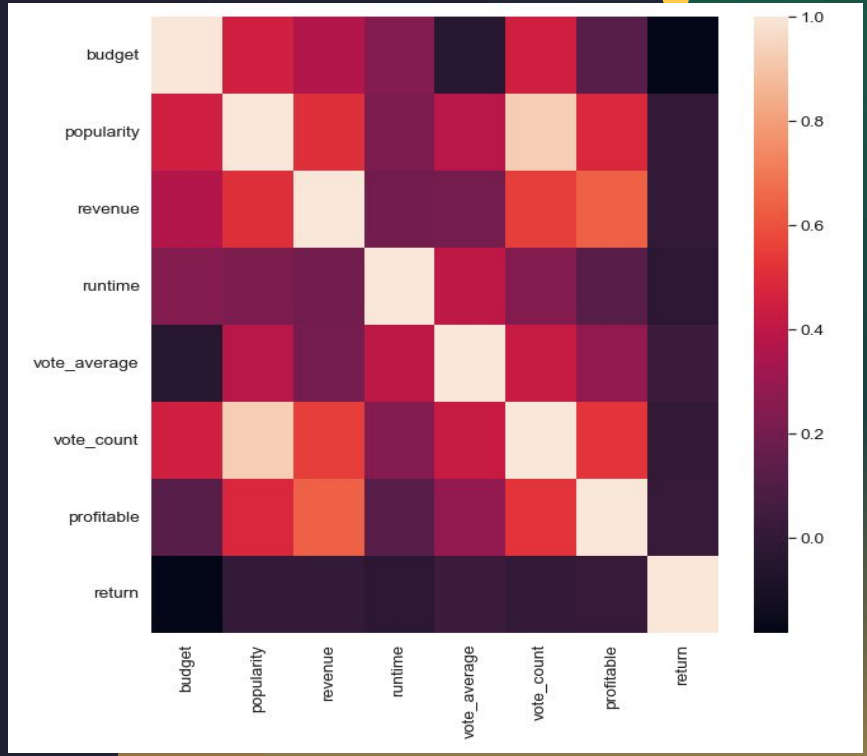


02

Exploratory Data Analysis

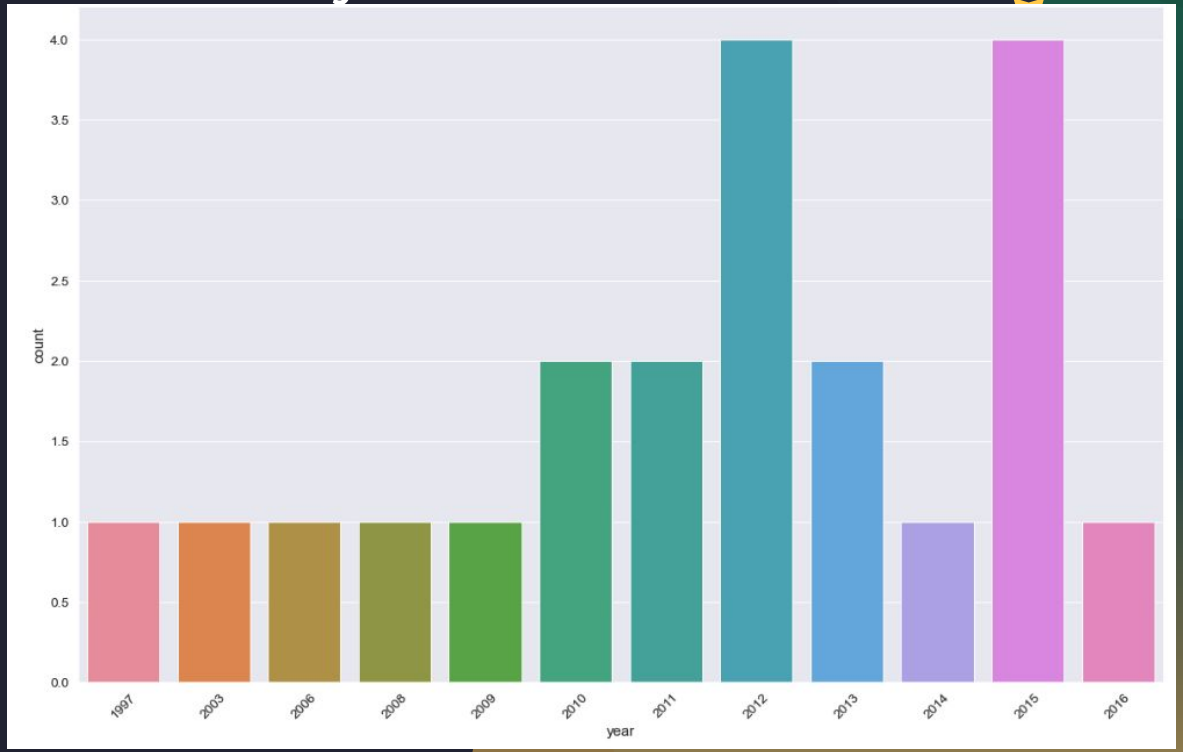
Correlation Analysis

- Most of the features have Pearson's Correlation Coefficient between 0.3 and 0.7 meaning that they have some relation between them



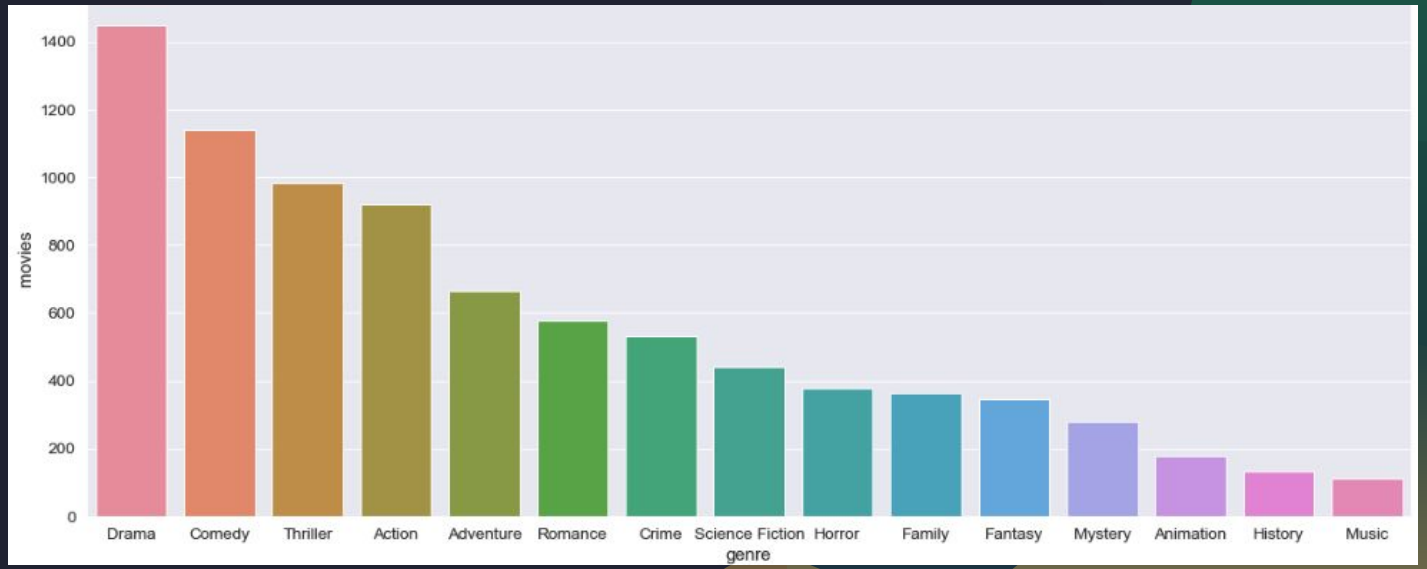
High-revenue and high-vote_count movies according to their release year

- After Titanic (1997) which was the first movie to reach 1 billion dollar mark, many more movies passed the threshold in 2010, 2011, 2012, 2013 and 2015



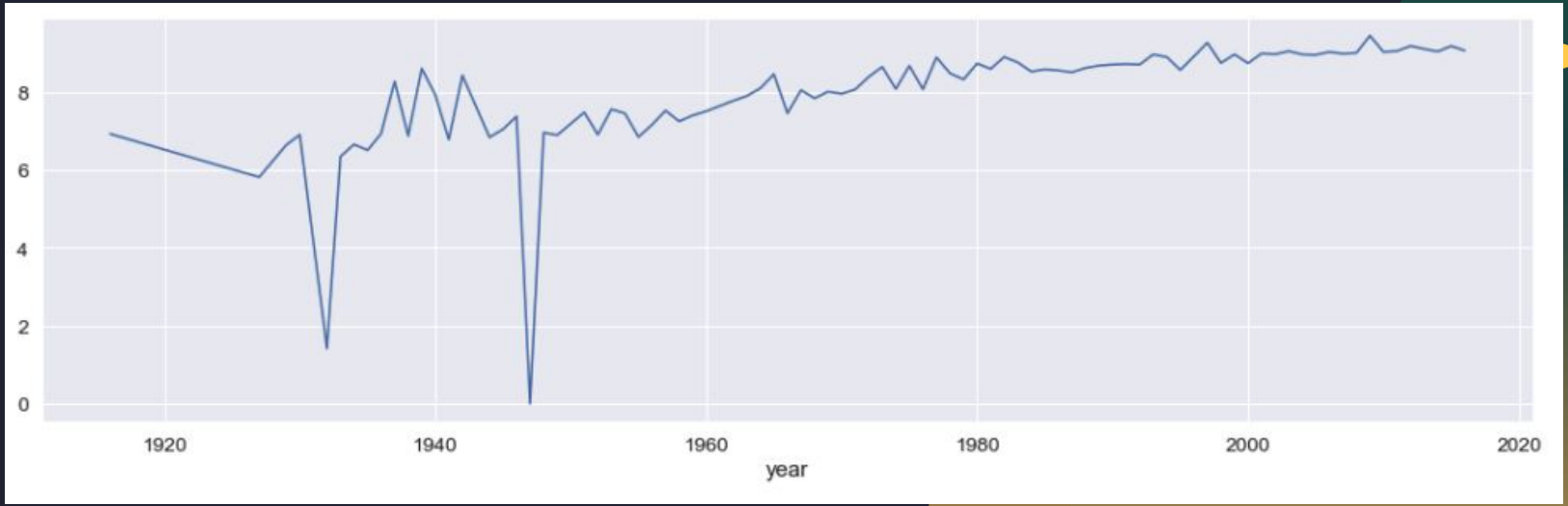
Genre feature Analysis

	genre	movies
0	Drama	1449
1	Comedy	1140
2	Thriller	984
3	Action	919
4	Adventure	666
5	Romance	576
6	Crime	533
7	Science Fiction	442
8	Horror	378
9	Family	362
10	Fantasy	347
11	Mystery	278
12	Animation	178
13	History	132
14	Music	112
15	War	108
16	Western	57
17	Documentary	30
18	Foreign	4
19	TV Movie	2



Revenue trend year over year

- Since the late 1940's the movie business has almost consistently increased their revenue year to year



03

Part 3: Predictive and Comparison Modeling

BalancedRandomForestClassifer

- The BRFC model was the first model chosen and did not perform badly at a 76% accuracy score but did not achieve a satisfactory level of accuracy

In [54]:
```
# Resample the training data with the BalancedRandomForestClassifier
from imblearn.ensemble import BalancedRandomForestClassifier
# Instantiate
brfc = BalancedRandomForestClassifier(n_estimators=100, random_state=1)

# Fit
brfc.fit(X_train, y_train)
```

Out[54]:
```
BalancedRandomForestClassifier(random_state=1)
```

In [55]:
```
# Calculated the balanced accuracy score
y_pred = brfc.predict(X_test)
balanced_accuracy_score(y_test, y_pred)
```

Out[55]:
```
0.7563359636144447
```

In [56]:
```
from imblearn.metrics import classification_report_imbalanced
print(classification_report_imbalanced(y_test, y_pred))
```

	pre	rec	spe	f1	geo	iba	sup
0	0.58	0.78	0.73	0.67	0.76	0.57	264
1	0.87	0.73	0.78	0.80	0.76	0.57	553
avg / total	0.78	0.75	0.76	0.76	0.76	0.57	817

EasyEnsembleClassifer

- The EEC model perform almost identically to BRFC model with a 76% accuracy score which makes sense as they are similar models

```
In [58]:
from imblearn.ensemble import EasyEnsembleClassifier
# Instantiate
eec = EasyEnsembleClassifier(n_estimators=100, random_state=1)

# Fit
eec.fit(X_train, y_train)
```

```
Out[58]: EasyEnsembleClassifier(n_estimators=100, random_state=1)
```

```
In [59]:
y_pred = eec.predict(X_test)
balanced_accuracy_score(y_test, y_pred)
```

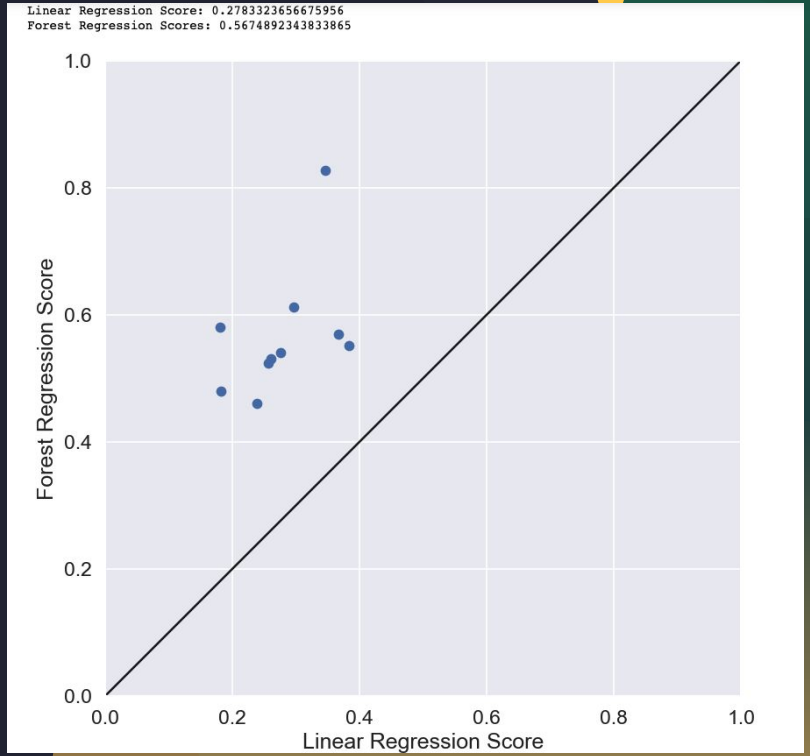
```
Out[59]: 0.7608567592744808
```

```
In [60]:
print(classification_report_imbalanced(y_test, y_pred))
```

	pre	rec	spe	f1	geo	iba	sup
0	0.59	0.78	0.74	0.67	0.76	0.58	264
1	0.88	0.74	0.78	0.80	0.76	0.58	553
avg / total	0.78	0.75	0.77	0.76	0.76	0.58	817

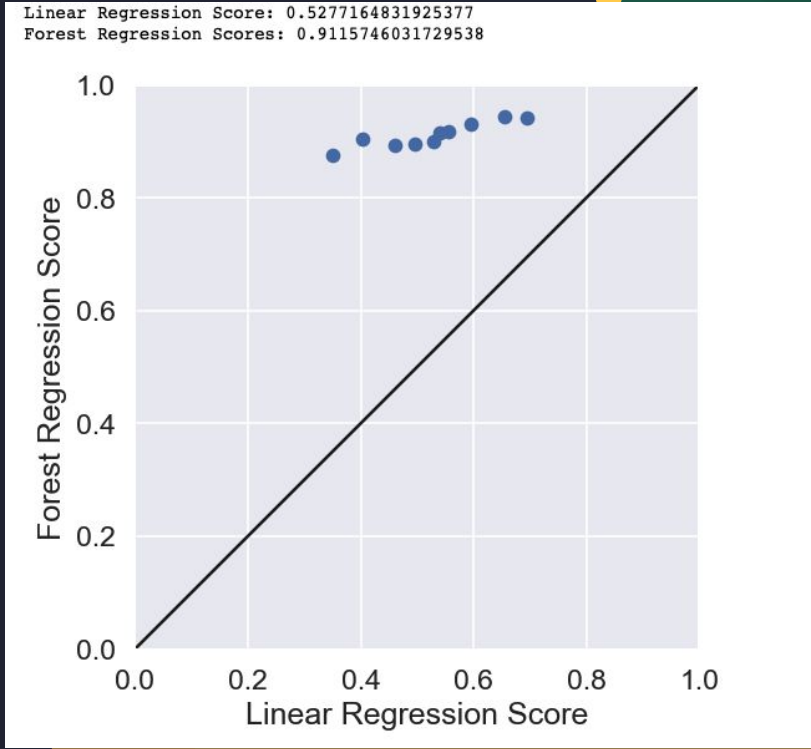
Linear Regression and Forest Regression

- The decision was made to take a different approach run Linear Regression and Forest regression models and the first attempt was underwhelming which accuracy scores of 27% and 57% respectively



Optimized Linear Regression and Random Forest Regression

- After the lackluster performance of the previous model there was an optimization step needed
- Movies that generated 0 revenue were excluded, and the models were rerun to determine if the fits improve
- They did! Forest Regression turned out to be the most successful best fit model ran (91% accuracy) and will be used for analysis in the future



Factors determining Profitability

- This analysis shows that most important factor in determining profitability is the amount of votes a movie receives on IMDB not as much the rating of those votes
- Budget makes sense as the second most important factor as more expensive movies tend to make more money

```
[('TV Movie', 0.0),
 ('Foreign', 0.0001951957463406404),
 ('Documentary', 0.0007567776409663647),
 ('History', 0.0011471648822844514),
 ('Western', 0.0016407820960412307),
 ('Animation', 0.002397877539520455),
 ('Music', 0.002946669490284031),
 ('War', 0.003304843375298297),
 ('Family', 0.004043559232404994),
 ('Fantasy', 0.00416928892040611),
 ('Horror', 0.004290840721428187),
 ('Adventure', 0.004416956660460765),
 ('Mystery', 0.004598713235178292),
 ('Romance', 0.004823908282913866),
 ('Action', 0.005377041323100831),
 ('Comedy', 0.006156013393299106),
 ('Crime', 0.006168237475322609),
 ('Science Fiction', 0.0062933246215915875),
 ('Thriller', 0.007034725373585503),
 ('Drama', 0.007446236132483877),
 ('vote_average', 0.052919822266391706),
 ('runtime', 0.05936054636151042),
 ('popularity', 0.08400605646173301),
 ('budget', 0.29309635284196606),
 ('vote_count', 0.4334090659254874)]
```

04

Conclusions

Conclusions

- There was success in determining if a movie would be profitable and the factors which determined profitability were found. More model optimization could be done as well as additional data points as the data set used only goes through 2017. It would be interesting to see if the same factors still determine profitability in a post-pandemic moviegoing world

Technologies, Languages, and Tools

- Python
- HTML
- JavaScript
- Jupyter Notebook
- VS Code

- SciKit Learn
- Pandas
- NumPy
- TensorFlow
- Matplotlib
- Seaborn

Data Source and Citations

- Web Data Source: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
- Github Repo:

  https://github.com/coxjack/Final-Project_Movie-Database