Predictive Modelling of Revenues of Modern American Movies

November 12, 2018

by Stephen Gou

Nov 12, 2018

Student Number: 1000382908

1 Introduction

A movie's box office is the most common metric to gauge its success. A good prediction of the revenue of a movie can guide production companies for building successful movies, and inform investors to pick out the most profitable movies. This project builds a model that predicts a movie's total revenue, given certain traits and facts about the movie. Only movies produced in the United States from 1990 to 2016 are considered, because the entertainment industry and economy changes over time. Movies produced after 2016 are not considered, because have not reached their full total revenue potential. Only movies produced in the U.S are considered, because the market characteristics vary over countries and the modelling of this aspect is beyond the scope of this project. This project aims to provide effective prediction as soon as the movies are released, which means that data like opening weekend box office, IMDb rating, social media sentiments cannot be used as features in the models.

To build an effective predictive model and gain insight, the project first explores and analyzes the major factors that affect a movie's revenue. And then, a model that best suits the case will be selected and trained. Its performance will be analyzed and compared to an alternative model. Last but not least, the model's limitations and potential improvements will be discussed.

2 Data Collection

This project makes use of several sources to collect data for analysis and training. Various types of data are collected that includes movie's revenue, budget, meta-data, cast, crews, rankings of actors and actresses and so on. The detail of all the datasets used is listed below.

1) TMDB 5000 Movies dataset. This is the main dataset which provides budget, revenue, runtime, genre, release-date and production country data. Source: https://www.kaggle.com/tmdb/tmdb-movie-metadata

2) New York Times Review dataset. This dataset includes data like whether a movie was picked by NYT critics, and review summaries. Source: NYT API

3) New York Times Review Articles sentiment data. This dataset includes the sentiment polarity score (from -1 to +1) for each movie. The articles were scraped from NYT movie reviews by BeautifulSoup. The sentiment polarity score was computed using the TextBlob library for NLP.

4) TMDB 5000 Crew dataset. This dataset has detailed cast and crew information, ranging from actor to writer, for each movie. Source: https://www.kaggle.com/tmdb/tmdb-movie-metadata

5) Top actors ranking data. This a Top 1000 Actors/Actresses Ranking published on IMDb. Source: https://www.imdb.com/list/ls058011111/

6) Top directors ranking data. This a Top Directors of All Time Ranking released published on IMDb. Source: https://www.imdb.com/list/ls006136043/

7) Annual CPI. This dataset lists the annual average CPI for U.S. Source: UsInflationCalculator.com

2.1 Data Cleaning

- Movies produced before 1990 and after 2016 were discarded.
- Movies produced outside of U.S were discarded.
- Movies with have zero revenue or budget, which might be a result of missing data or unreleased movie, were removed.
- Movies that do not have sentiment polarity score were assigned neutral score (0), after normalization.

3 Feature Selection and Transformation

There are a large amount of factors that might affect a movie's revenues ranging from movies' meta-data, to unemployment rate of the release year. Features that will be analyzed and incorporated into the predictive model are selected based on availability, informativeness, unambiguity, and interpretability. According to this criteria, the following features are selected: budget, runtime, critics-pick, genres, MPAA-rating, cast, and director. The following procedures and transformations of data are done to make data representable for modelling and to increase accuracy.

1) The cast of a movie is represented by a popularity score, which is calculated by the following rule. A percentile rank score for each cast is calculated according to the actors rank dataset. Then use 1 - percentage rank as the popularity score for a cast. So 1 is the highest one can get and 0 is the lowest (0 if cast not in the ranking). Then the cast popularity for the movie is calculated as following:

$$Cast\ Popularity\ Score = \sum_{i}^{N} \gamma^{i}(1 - PercentileRank(Cast\ i))$$

where gamma is a decay factor and N is the number of casts.

2) The director is represented by a popularity score, which is calculated by the following rule. A percentile rank score is calculated according to the directors rank dataset. Then use 1 - percentage rank as the popularity score. So 1 is the highest one can get and 0 is the lowest (0 if director not in the ranking).

3) The revenue and budget are adjusted for inflation according to the rule:

$$adjusted = \frac{CPI(2017)}{CPI(release\,year)} * unadjusted.$$

4) Genres are converted by one-hot encoding. Note that a movie can have multiple genres associated with it.

5) MPAA ratings are converted by one-hot encoding.

6) Runtime represented by a number and unchanged.

7) Critics pick is represented by 1 or 0 (1 repesents being picked)

8) New York Times review sentiment is represented by a polarity score. Negative score means negative sentiment and vice versa. The distribution of polarity scores of review articles have a positve mean. As a result, they are normalized to be more interpretable. For example, a negative score would mean that the review is more negative than the average sentiment polarity of all review articles.

4 Exploratory Data Analysis

Some observations can be made from the statistics of our wrangled dataset. - There are 2,009 movies in our final dataset. - 14% of the movies are picked by the critics. - Average runtime of a movie is 108 minutes while the lengthiest runs more than 4 hours, the shortest runs 46 mintues. - More than 66% of the movies are at least rated PG-13. - The most common genres are Drama, Comedy, and Action.

4.1 Distributions of Data

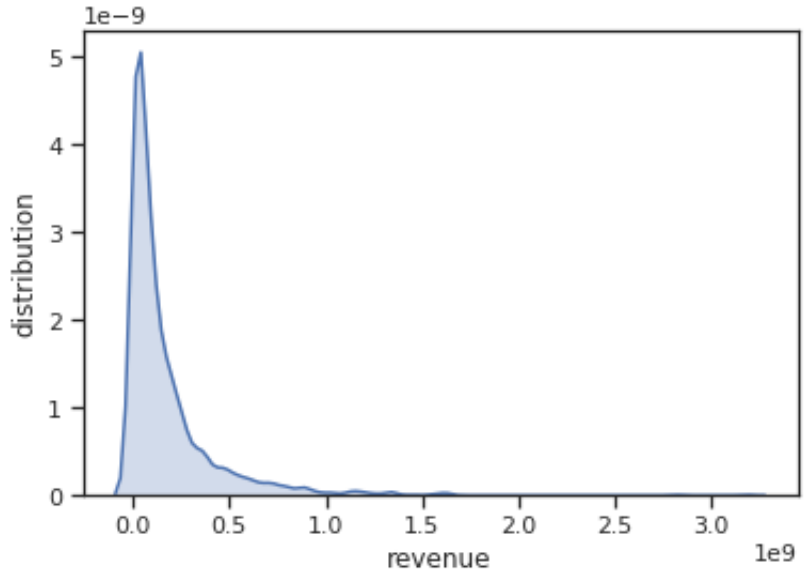
Both revenue and budget have wide range of values and high positive skewness. Therefore, a log transformation were applied before modelling.

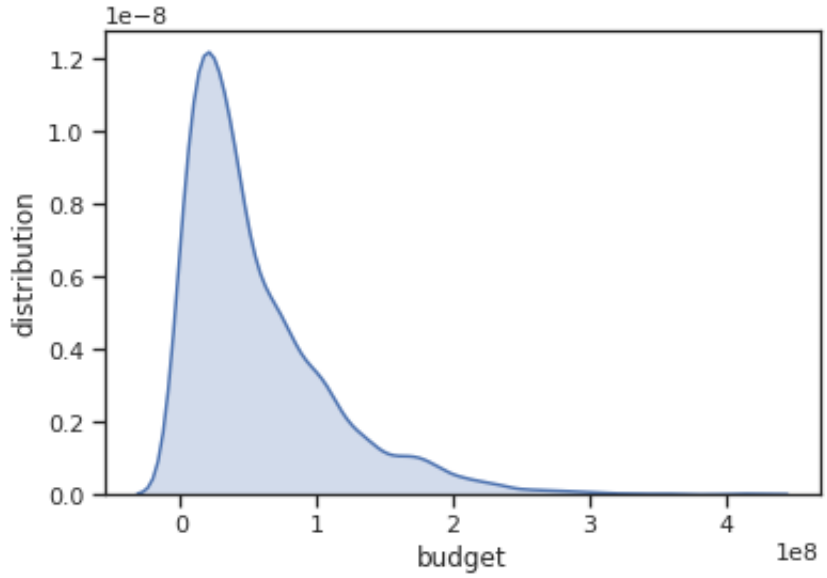
Both revenue and budget have large number of outliers. Revenue has outliers with more extreme values.

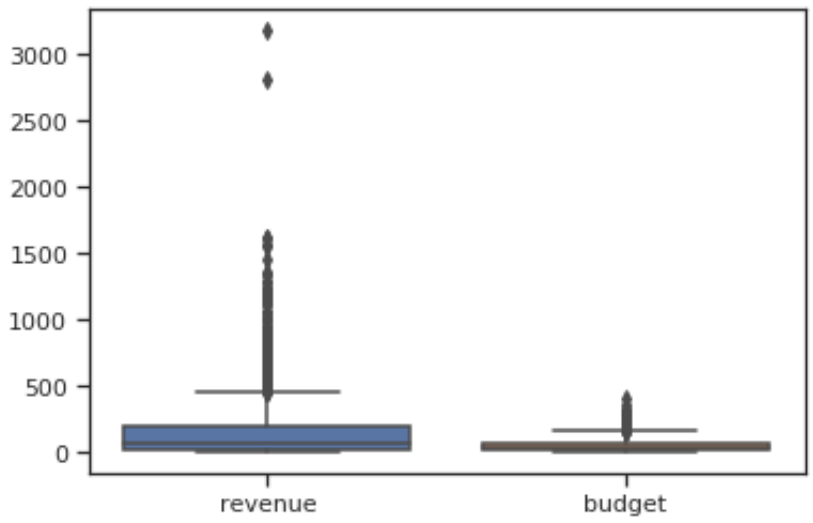
4.2 Correlations Between Features

A heatmap of correlation between features is plotted to spot features that have strong relationships with each other, so that redundant features can be discarded to reduce multicollinearity.

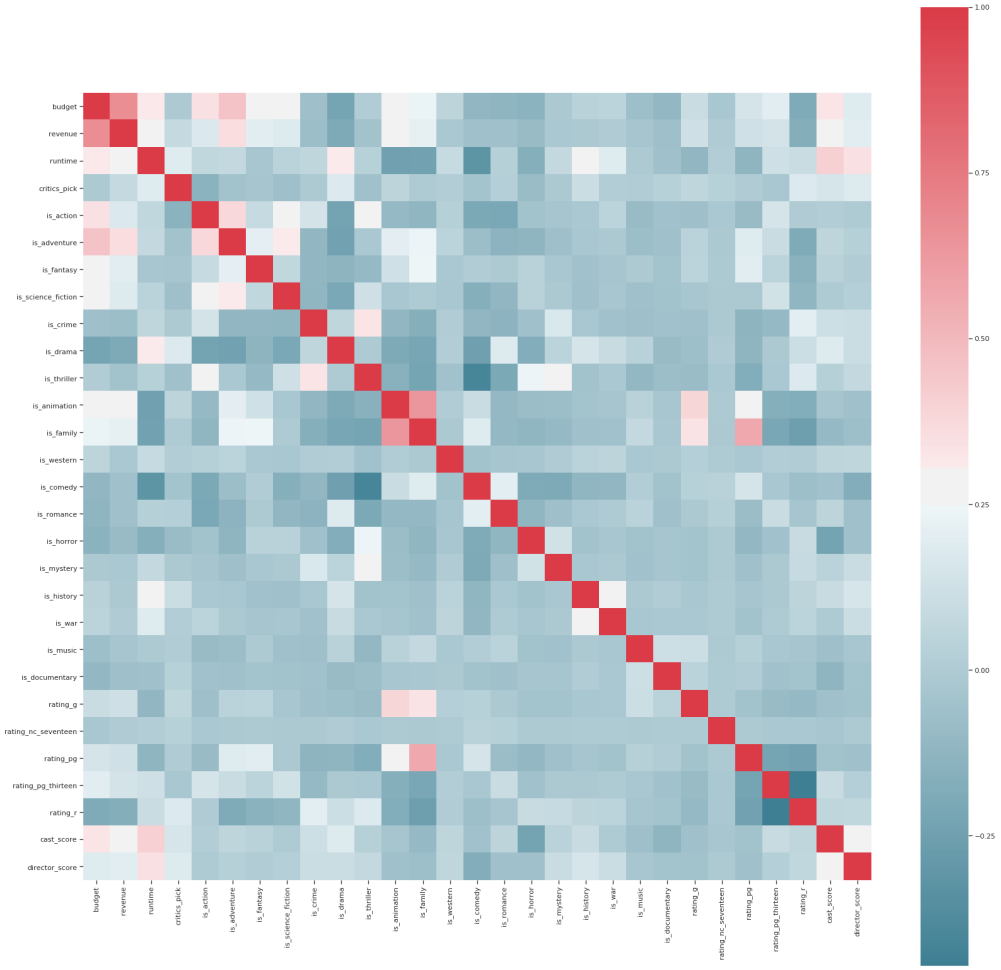
- Genres and mpaa-rating tend to have strong correlations. From the plot, it's clear that movies that have "family" as a genre is also very likely to have "animation" as a genre as well. Family and animation movies also usually have PG or G rating.
- The quality of the cast appear to be uncorrelated with most of the genres of movies except for horror, where quality of cast drops significantly.


1e−9
5
4
3
2
1
0
distribution
0.0
0.5
1.0
1.5
2.0
2.5
3.0
revenue
1e9


1e−8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
distribution
0
1
2
3
4
budget
1e8



revenue and budget



- Intuitively, the runtime of a movie has correlations with its genres, which is confirmed by the heatmap. The runtime also correlates with budget and quality of director and cast.
- New York Time's critics' picks appear to be uncorrelated with most of the features of a movie, meaning that critics do not favor particular types of movies over the others. Action and thriller movies are marginally less likely to be picked, but that could just be a result of noise.

From these observations, runtime and mpaa-rating of a movie could be potentially discarded, because they usually depend on other features of the movie.

4.3 New York Times Critics Pick and Review Sentiment

A regression of the normalized sentiment polarity score based on critics picks gives a extremely small p-value, and coefficient of 0.25. This means that critics pick does have some amount of effects on the sentiment of their review article, but the effects are very small. Since they do not show strong collinearity, both are kept as features.

5 Analysis and Modelling

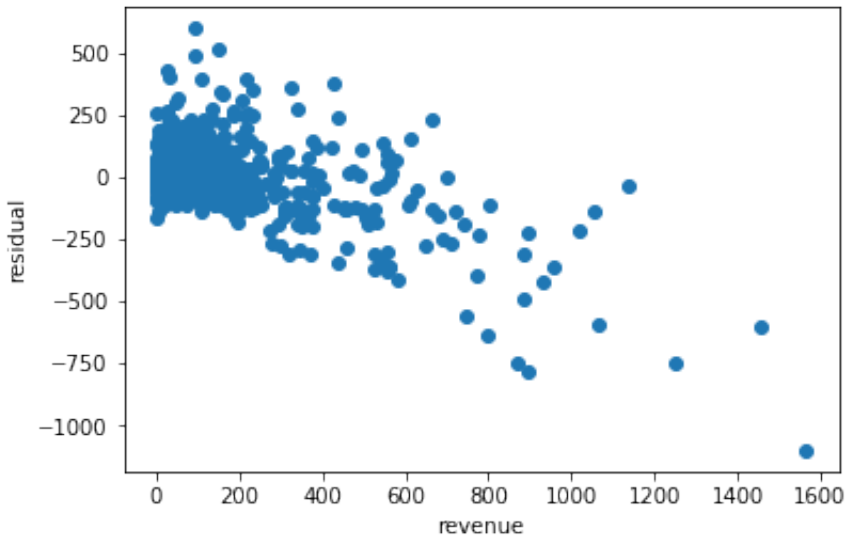
Since the goal is to predict revenue, a continuous value over a wide range, regression models are considered. More specifically, OLS regression, Regression Tree, k-Nearest-Neibours Regressor and Multilayer Perceptrons are the candidate models.

5.1 Model Performance Evaluation

The wrangled dataset is split into 70% training and 30% testing.

Initially, the models' performance are evaluated based on the R-Squared statistic and the residual plot. An OLS regression model that simply includes all the features without adding higher order terms and interactions is fitted and its result is used as a baseline.

It obtained a **R-Squared score of 0.492** on the test set and the residual plot as below:



However, the R-Squared statistic is not very informative. Revenue is an unbounded number with high variance in nature, and there is a large number of outliers in the dataset. Moreover, the

residual plot displays a clear linear relationship between residual and the revenues of the movies, meaning that there is a significant pattern of revenues of movies that it is not explained by the features. However, there is inherently large uncertainly of movie's revenue and given limited information there are about the movies. It's necessary to find a more effective metric to evaluate the models.

The alternative metric is defined as the percentage of predictions that are within 20% error from the true value when converted back to normal scale from log scale. It will be referred to as "accuracy". The baseline is **61.6%**, which is obtained by simply use the mean revenue for every prediction.

5.2 OLS Linear Regression

The following steps are taken to improve the performance, interpretability and reduce overfitting.

- Like suggested in EDA, mpaa-rating is discarded because it depends on other features.
- Genres are discarded as well. OLS regression shows an extremely large condition number (> 10^ 10) with genres included, meaning there are strong multi-collinearity. In addition, it caused certain terms to have extremely large weights, even when L1/L2 reguarizers are added. Lastly, in introduced too many potential interactions between each other and other features like directors, actors and budget.
- Naturally removing outliers from dataset was considered, but removing them did not improve any model's performances. Therefore, outliers are kept.
- A brute force model that includes a large number of interactions between between features, and certain second order terms (102 total terms in regression formula) is fitted. However, it did not improve model accuracy.

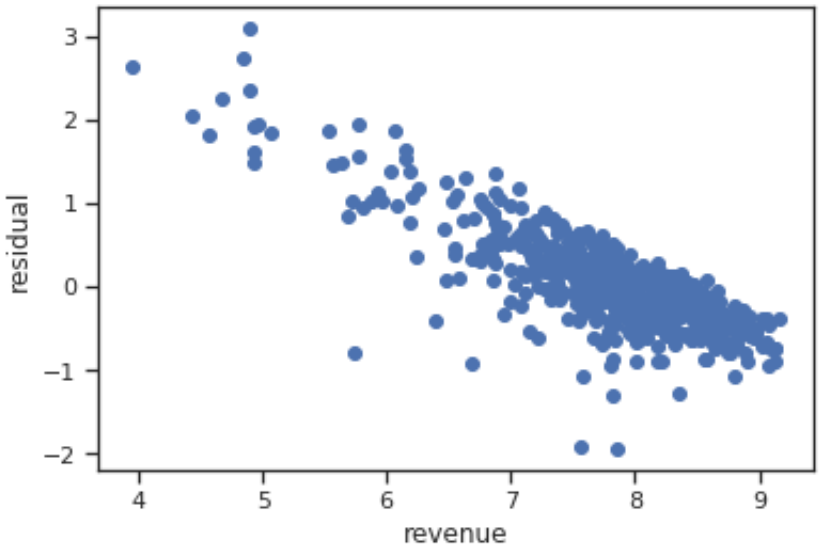
The resulting formula for regression is

$$revenue = Intercept + w_0 * runtime + w_1 * budget + w_2 * \text{critics pick} + w_4 * \text{cast score} + w_5 * \text{director score} + w_6 * \text{se}$$

It achieved an accuracy of **61.5%**, which is almost identical to the baseline accuracy by just predicting the mean of log revenue. Therefore, this linear model did not offer much explaining power. However, the analyzing the coefficeitns of the model does provide some insight into how the features affect revenue. Budget, cast score, director score, and critics all have coefficients with significant p-values, but small coefficients. Runtime doesn't have a coefficient with extremely large p-value. Simply including sentiment polarity will produce a non-sigfinicant coefficient, but including interaction between sentiment polarity and budget result in significant coefficients. It means that positive sentiment polarity itself might not increase revenue, but it does have an effect if the budget of the movie is big.

5.3 Non-Linear Regression Models

Given the large number of potential interactions and non-linear relationship between certain features and revenue, it is extremely hard manually select features. Thus, non-linear models like regression tree and multilayer perceptron (neutral networks with only fully connected hidden layers) are considered. For these models, all available features are included in the input. Hyper

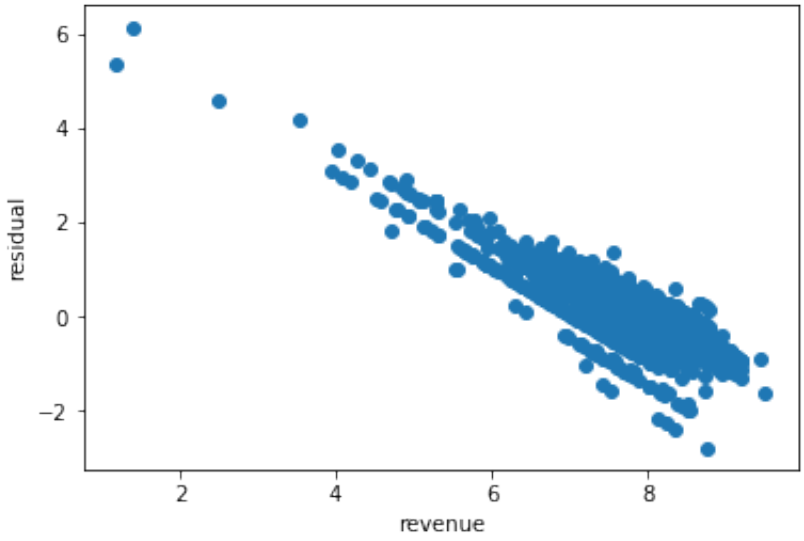


parameters were tuned by informal search based on their performance in a 5-fold cross-validation test.

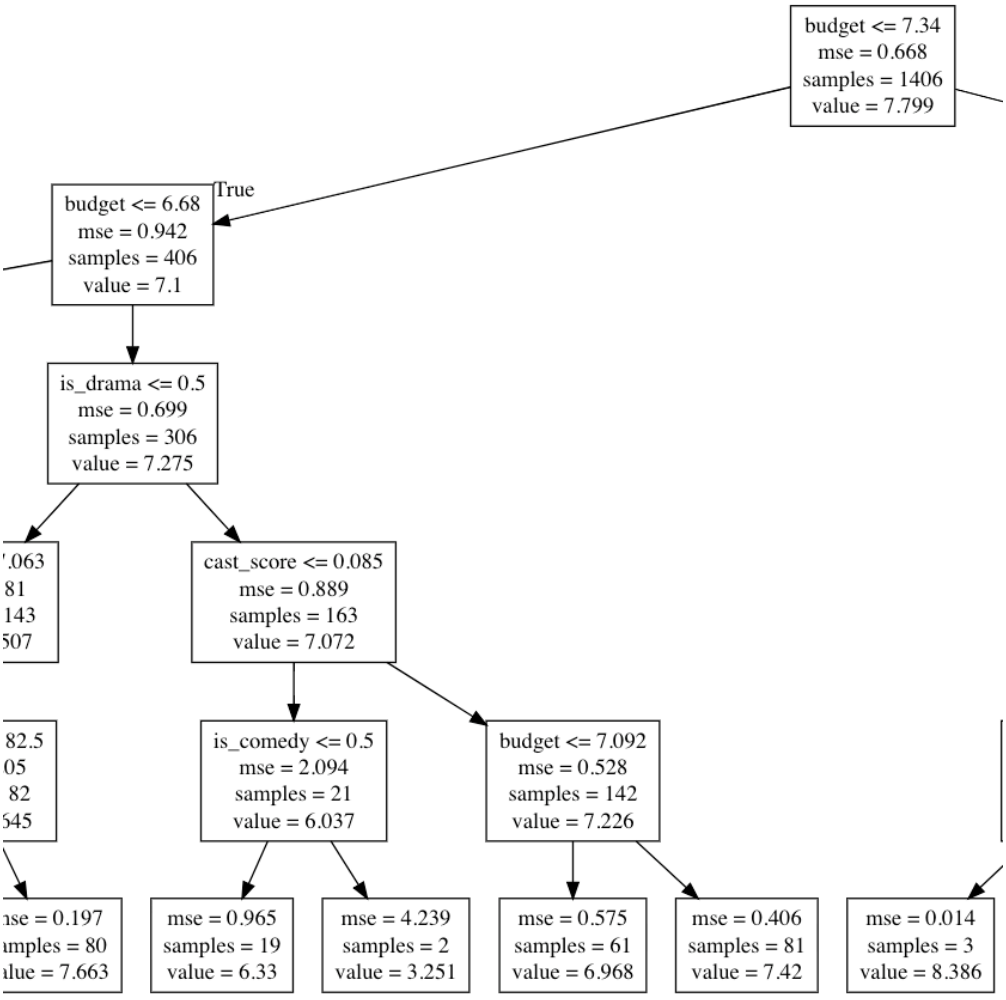
Input Features: budget(log transformed), runtime(in minutes), critics pick(0-1), sentiment polarity score, genres(one-hot encoded), mpaa rating(one-hot encoded), cast score, director score.

5.3.1 Regression Tree

A regression tree with max depth of 5 achieved accuracy of **63.3%** and R-Squared of **0.387**.

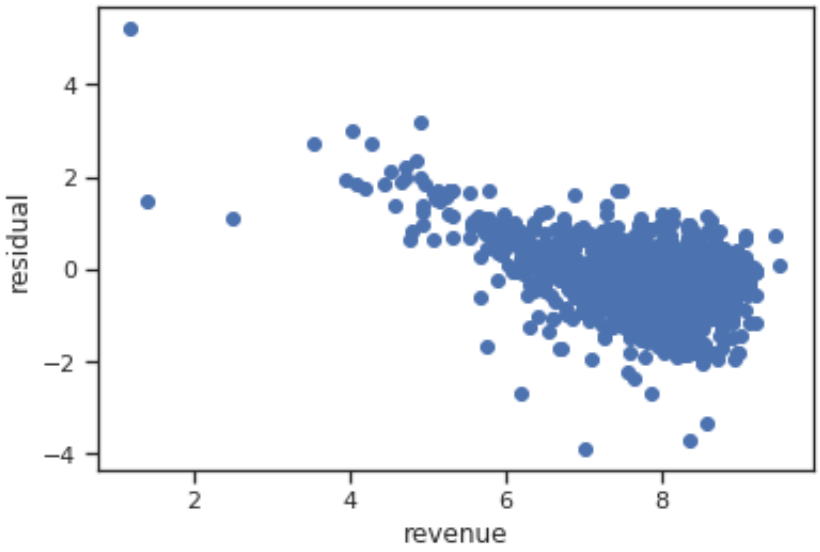


The top left section of the tree is



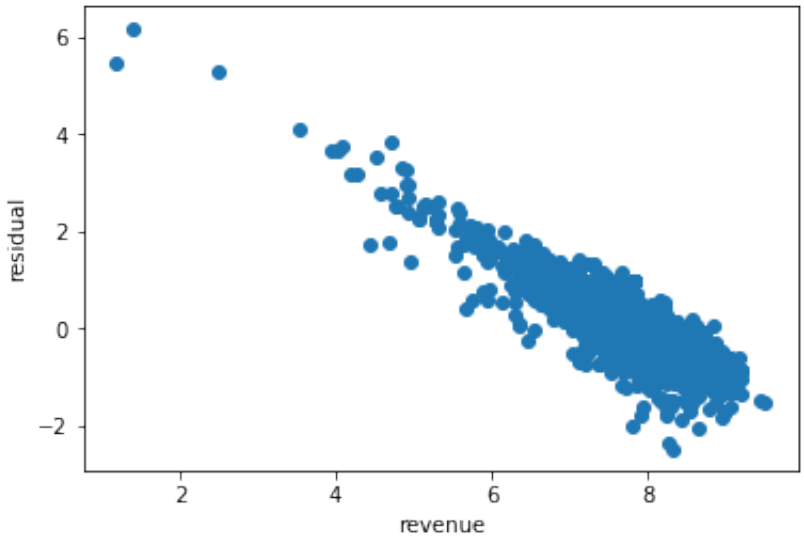
shown as below:

Budget appears to provide the most information gain, as it's the root of the tree and it appears most frequently in top layers. More interestingly, certain genres, for example drama, and comedy provide considerable information gain. For example, in the graph above, there is a path that can be interpreted as the following: for a low-medium budget drama movie, if it has poor cast (cast score < 0.085), it only averages revenue of \$ 1,088,930, otherwise it averages \$ 16,826,740.



5.3.2 kNN

A kNN model with k=5 obtained achieved accuracy of **61.4%** and R-Squared of **0.336**.



5.3.3 Neural Network

A neural network with two fully connected hidden layers, each of size 64 with ReLU activations, was trained for 2,000 iterations multiple times. The best network achieved accuracy of **74.2%** and R-Squared of **0.392**.

The neural network achieved the best prediction accuracy out of all the other models by a wide marge, and beat the baseline by 13%. The residuals are more evenly distributed and have less linear downward trend, comparing to that of other models, indicating that the neural network is able to better

6 Results

The project built a model (Multilayer-Perceptron), that is able to predict the lifetime revenue of a modern American movie within a 20% error margin **74.2%** of times. To build the model, the project constructed a dataset of American movies produced between 1990 and 2016 that include features: budget, runtime, NYT critics pick, review sentiment polarity, genres, mpaa_rating, cast_score, and director_score.

Other models including OLS linear regression, regression tree, and k-Nearest-Neighbours were exmained and trained, but did not achieve better result than baseline.

The OLS regression model shows that budget, cast score, director score, and critics all have coefficients with significant p-values, but small coefficients. Sentiment polarity score has more influence when budget of the movie is higher.

While the regression model failed to beat the baseline, neural network outperformed it by a 13% margin. This suggests that the features in our dataset do have correlations with the revenue but wasn't captured by the regression model. The reason could be that non-linear relationships, higher order terms of the features or complicated multi-way interactions between features were not accurately included in the OLS regression model.

7 Limitations

The models were trained only on data of American movies produced between 1990 and 2016. It will not apply to movies produced in other countries, and or too far from this time period into the future. The model cannot predict the revenue before the release of a movie because it utilizes the critics picks and movie review sentiments, which are usually only available after release.

8 Future Work

8.0.1 Find better feature data

Since the residual shows that there is a significant linear pattern for the revenues was not explained by the model, more relevant data and data that have more explaining power might be explored and incorporated into the models. For example, the number of views of movie's trailers before release, social media influence of casts, writer of the movie, production company, season and so on. Another possibility is to loosen the assumption so that more post-release data can be incoporated like opening weekend box office, IMDb rating, hashtag counts and so on.

8.0.2 Improve existing features

There are room for improvements of the features currently used in the models. For example, how the cast and director score is calculated could be improved. Instead of rank based, one could include more revenue related traits for example, actors' social media following, revenues of their past 3 movies, and so on.

8.0.3 Improve modelling

Since the models are systemically predicting overestimated revenue for low revenue movies and underestimated revenue for high revenue movies, there might be opportunity to take advantage

of this observation. For example, use locally weighted regression, kNN or an ensemble of models so that movies in differernt levels can be modelled separately.

8.1 Python Code

```
In [1]: import json
        import requests
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns;
        from pandas.io.json import json_normalize
        import statsmodels.api as sm
        import statsmodels.formula.api as smf
        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import cross_val_predict
        from sklearn.model_selection import cross_val_score
        from sklearn import linear_model
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.neural_network import MLPClassifier
        from sklearn.neural_network import MLPRegressor
        from sklearn.ensemble import AdaBoostRegressor
        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.ensemble import BaggingRegressor
        from sklearn import tree
```

8.1.1 Cleaning and Feature Mapping

```
In [2]: github_raw_root = 'https://raw.githubusercontent.com/gouzhen1/Moives-Data-Analysis/mast

        #NY Reviews Dataset
        ny_df = pd.read_csv(github_raw_root + 'NY_movie_reviews.csv')
        ny_df.rename(columns={'display_title':'title'},inplace=True)
        ny_df = ny_df[['title','mpaa_rating','critics_pick']]

        ny_sentiment_df = pd.read_csv(github_raw_root + 'ny_reviews_sentiment.csv')[['title','s
        ny_sentiment_df['sentiment_polarity'] = (ny_sentiment_df['sentiment_polarity'] - ny_ser

        #Wrangle actors and director
        #TMDB Credits Dataset (for cast and director)
        tmdb_credits_df = pd.read_csv(github_raw_root + 'tmdb_5000_credits.csv')
        actors_rank = pd.read_csv(github_raw_root + 'Top_actors_rank.csv')['Name'].tolist()
        directors_rank = pd.read_csv(github_raw_root +'All_time_director_rank.csv')['Name'].tol
        total_actors = len(actors_rank)
        total_directors = len(directors_rank)
```

```
def transform_cast(df):
    cast_json = df['cast']
    parsed_cast = json.loads(cast_json)
    score = 0.
    count = 0
    for cast in parsed_cast:
        actor = cast['name']
        if actor in actors_rank:
            #discounted for later casts
            score += (0.9 ** count) * (1. - (actors_rank.index(actor)/total_actors))
        count += 1
    return score
tmdb_credits_df['cast_score'] = tmdb_credits_df.apply(transform_cast, axis = 1)

def transform_crew(df):
    crew_json = df['crew']
    parsed_crew = json.loads(crew_json)
    score = 0.
    for crew in parsed_crew:
        if crew['department'] == 'Directing' and crew['job'] == 'Director':
            director = crew['name']
            if director in directors_rank:
                score += (1. - (directors_rank.index(director)/total_directors))
            break
    return score

tmdb_credits_df['director_score'] = tmdb_credits_df.apply(transform_crew, axis = 1)
tmdb_credits_df = tmdb_credits_df[['title','cast_score','director_score']]

#TMDB Main Dataset
main_df = pd.read_csv(github_raw_root + 'tmdb_5000_movies.csv')
main_df['release_date'] = pd.to_datetime(main_df['release_date'])
main_df.drop(main_df[main_df['release_date'].dt.year < 1990].index, inplace=True)
main_df.drop(main_df[main_df['release_date'].dt.year > 2016].index, inplace=True)
main_df = main_df[main_df['revenue'] > 0]
main_df = main_df[main_df['budget'] > 0]
main_df = main_df.merge(ny_df,how='left')
main_df = main_df.merge(ny_sentiment_df, how='left')

#process and filter countries
def process_country(df):
    country_json = df['production_countries']
    parsed_country = json.loads(country_json)
    if len(parsed_country) > 0:
        return parsed_country[0]['name']
    else:
        return None
main_df['production_countries'] = main_df.apply(process_country, axis = 1)
```

```
main_df = main_df[main_df['production_countries'] =='United States of America']
main_df.drop(columns='production_countries',inplace=True)

#wrangle genre
genre_dict = {}
def transform_genre(df):
    genre_json = df['genres']
    parsed_genre = json.loads(genre_json)
    result = []
    for genre in parsed_genre:
        genre_name = genre['name'].replace(' ','_')
        result.append(genre_name)
        if genre_name not in genre_dict:
            genre_dict[genre_name] = 1
        else:
            genre_dict[genre_name] += 1

    return result
main_df['genres'] = main_df.apply(transform_genre, axis = 1)
#drop very low rare genres
del genre_dict['Foreign']
for genre in genre_dict:
    main_df['is_' + genre] = main_df['genres'].transform(lambda x: int(genre in x))
main_df.drop(columns=['genres'],inplace=True)

#map mpaa rating
rating_df = pd.get_dummies(main_df['mpaa_rating'],prefix='rating')
main_df = main_df.merge(rating_df,left_index=True,right_index=True)
main_df.drop(columns=['mpaa_rating','rating_Not Rated'],inplace=True) #drop one catego

#adjust revenue and budget for inflation
cpi_df = pd.read_csv(github_raw_root + 'Annual_CPI.csv')
cpi_df = cpi_df.set_index('DATE')
cpi_dict = cpi_df.to_dict()['CPIAUCSL']
def get_cpi_adjusted_revenue(df):
    year = df['release_date'].year
    revenue = df['revenue']
    return cpi_dict['2017-01-01']/cpi_dict['{}-01-01'.format(year)] * revenue

def get_cpi_adjusted_budget(df):
    year = df['release_date'].year
    budeget = df['budget']
    return cpi_dict['2017-01-01']/cpi_dict['{}-01-01'.format(year)] * budeget

main_df['revenue'] = main_df.apply(get_cpi_adjusted_revenue,axis=1)
main_df['budget'] = main_df.apply(get_cpi_adjusted_budget,axis=1)
main_df['revenue'] = np.log10(main_df['revenue'])
main_df['budget'] = np.log10(main_df['budget'])
```

```
main_df = main_df.drop(columns = ['release_date','original_language','popularity','home
```

In [3]:
```
main_df = main_df.merge(tmdb_credits_df,how='left')
main_df.columns = map(str.lower, main_df.columns)
main_df.rename(columns={'rating_pg-13':'rating_pg_thirteen','rating_nc-17':'rating_nc_s
main_df['critics_pick'].fillna(0,inplace=True)
main_df['sentiment_polarity'].fillna(0.0, inplace=True)
main_df.to_csv('wrangled_dataset.csv')
main_df.rename(columns={'rating_not rated':'rating_not_rated'},inplace=True)
main_df.head()
```

Out[3]:

	budget	revenue	runtime	title
0	8.432603	9.503142	162.0	Avatar
1	8.549842	9.055444	169.0	Pirates of the Caribbean: At World's End
2	8.426407	9.063873	165.0	The Dark Knight Rises
3	8.443441	8.481998	132.0	John Carter
4	8.484341	9.022536	139.0	Spider-Man 3

	critics_pick	sentiment_polarity	is_action	is_adventure	is_fantasy
0	1.0	0.562590	1	1	1
1	0.0	-0.892757	1	1	1
2	1.0	-0.538564	1	0	0
3	0.0	1.567143	1	1	0
4	0.0	0.915226	1	1	1

	is_science_fiction	...	is_war	is_music	is_documentary
0	1	...	0	0	0
1	0	...	0	0	0
2	0	...	0	0	0
3	1	...	0	0	0
4	0	...	0	0	0

	rating_g	rating_nc_seventeen	rating_pg	rating_pg_thirteen	rating_r
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	1	0

	cast_score	director_score
0	1.412578	0.836364
1	2.403865	0.400000
2	4.555502	0.709091
3	1.787542	0.000000
4	2.124545	0.490909

[5 rows x 31 columns]

8.2 EDA

In [4]:
```
main_df.describe()
```

Out[4]:

	budget	revenue	runtime	critics_pick
count	2009.000000	2009.000000	2009.000000	2009.000000
mean	7.508704	7.792729	108.490791	0.141364
std	0.611931	0.812651	18.208264	0.348483
min	1.022147	1.192343	63.000000	0.000000
25%	7.274876	7.440466	96.000000	0.000000
50%	7.600529	7.921382	106.000000	0.000000
75%	7.901559	8.310837	118.000000	0.000000
max	8.617163	9.503142	214.000000	1.000000

	sentiment_polarity	is_action	is_adventure	is_fantasy
count	2009.000000	2009.000000	2009.000000	2009.000000
mean	-0.006764	0.268790	0.197113	0.101543
std	0.887636	0.443441	0.397917	0.302122
min	-4.252729	0.000000	0.000000	0.000000
25%	-0.386116	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000
75%	0.478528	1.000000	0.000000	0.000000
max	3.889671	1.000000	1.000000	1.000000

	is_science_fiction	is_crime	...	is_war
count	2009.000000	2009.000000	...	2009.000000
mean	0.126431	0.158785	...	0.018915
std	0.332417	0.365567	...	0.136258
min	0.000000	0.000000	...	0.000000
25%	0.000000	0.000000	...	0.000000
50%	0.000000	0.000000	...	0.000000
75%	0.000000	0.000000	...	0.000000
max	1.000000	1.000000	...	1.000000

	is_music	is_documentary	rating_g	rating_nc_seventeen
count	2009.000000	2009.000000	2009.000000	2009.000000
mean	0.030861	0.012942	0.020408	0.000996
std	0.172984	0.113051	0.141427	0.031544
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000

	rating_pg	rating_pg_thirteen	rating_r	cast_score
count	2009.000000	2009.000000	2009.000000	2009.000000
mean	0.106521	0.303136	0.328522	1.343395
std	0.308580	0.459728	0.469793	0.968162
min	0.000000	0.000000	0.000000	0.000000