Predictive_Model_of_movie_Revenue

October 29, 2018

1 Predictive Modelling of Revenues of Modern American Movies

by Stephen Gou

Oct 28, 2018

Student Number: 1000382908

1.1 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.

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.

1.2 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) 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

4) Top Actors/Actresses Rank. This the list of a Top 1000 Actors/Actresses Ranking released by IMDb. Source: IMDb

5) Top directors Rank. This the list of a Directors Ranking released by IMDb. Source: IMDb

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

1.2.1 Cleaning

Movies produced before 1990 and after 2016 are discarded. Movies produced outside of U.S are discarded. Some movies have zero revenue in the dataset, which might be a result of missing data or unreleased movie. These movies are removed.

1.3 Feature Selection and Mapping

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:

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

   CPI are from the Annual CPI data.

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

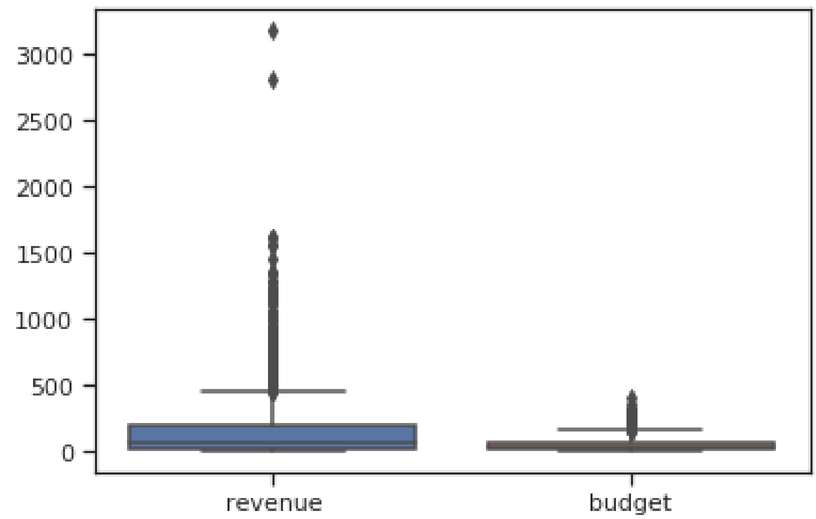
5) MPAA ratings are converted by one-hot encoding.

6) Runtime represented by a number and unchanged.

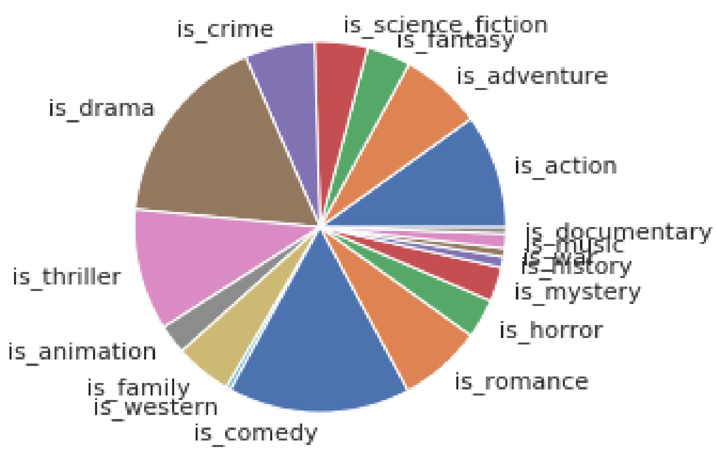
7) Critics pick is represented by 1 or 0 (1 repesents being picked)

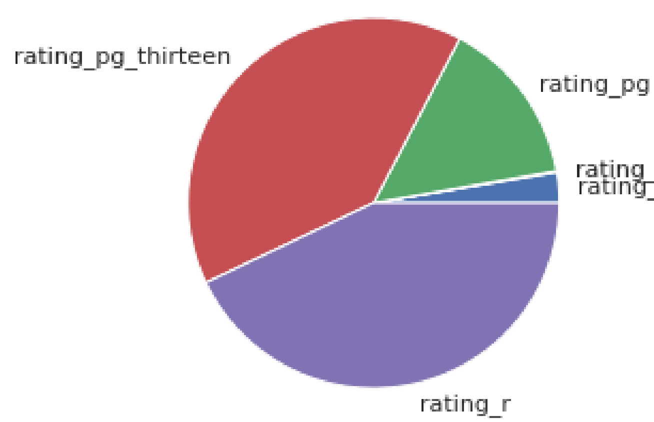
1.4 Exploratory Data Analysis

Some observations can be made from the statistics of our wrangled dataset. There are 2,033 movies in our final dataset. 17% 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.



Distributions of Data Revenues and budgets of movies are concentrated in low values, \$ 78m and \$ 37m respectively. There are large number of outliers in both cases. However, revenue has very long-tail towards higher values and outliers with more extreme values.





There are quite diverse and evenly distributed number of genres in the data. And majority of movies are at least PG-13.

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.

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.

Another interesting observation is that New York Time's critics' picks appear to be uncorrelated with most of the features of a movie, meaning that they are not favoring a particular subsets 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.

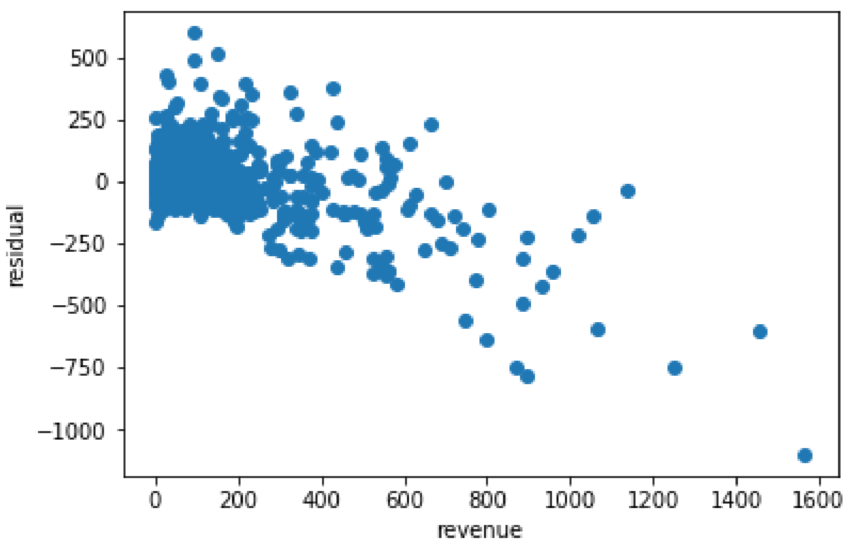
From these observations, runtime and mpaa-rating of a movie could be potentially discarded, because they usually depend on other features of the movie.

1.5 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, Ridge/Lasso regression and Random Forest regression are the candidate models. 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. Model is trained on training set, which is 70% of the dataset.

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



1.5.1 Feature Analysis and Engineering

The goal is to improve the predictive power of models through finding and creating better features. 1) find features that have significant influence on our dependable variable, revenue 2) discard features that heavily correlate with other features to reduce multi-collinearity 3) brainstorm and discover interactions between features 4) consider cofounders and higher order terms of the features. OLS regression and Logistic regression are used to perform the analysis.

- Like suggested in EDA, mpaa-rating is discarded because it depends on other features.

- Intuitively, revenues are more than linearly related with star and director power and budget. Thus,quadratic terms of cast score, director score, budget score are added to the model, and it improves the fitting of the model. Exponential terms are experimented with too, and it gives similar result to quadratic terms.
- Intuitively, the cast's influence on the revenue might depend on the genres of the movie as well. Interaction between cast and each genres are added to the model, but most of them have very large p-values except for animation and adventure, which have significant p-values.

1.5.2 Predictive Modelling

pre release model, post release model regression regression tree KNN ## Results we found that xxxx are major factors, yyy not so much. thus we built a model with ... with zzz model. Multi-collinearity (dummy varaible trap) ## Conclusion ### Future Work Actors and directors representation Incorporate economy/meta data, e.g GDP growth rate of the year of release, unemployment rate.

1.6 Python Code

```
In [27]: 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 import linear_model
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
```

1.6.1 Cleaning and Feature Mapping

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

        #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']]

        #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'].to
        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 += (0.8 ** 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.merge(ny_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)
```

```python
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'] = main_df['revenue'] * 0.000001
main_df['budget'] = main_df['budget'] * 0.000001
```

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

```
In [3]: print('wrangled dataset: ' + str(main_df.shape))
        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.to_csv('wrangled_dataset.csv')
        main_df.rename(columns={'rating_not rated':'rating_not_rated'},inplace=True)
        main_df.head()
```

```
wrangled dataset: (2033, 28)
```

```
Out[3]:        budget      revenue  runtime                                     title  \
        0  270.771526  3185.238655    162.0                                    Avatar
        1  354.684562  1136.172881    169.0  Pirates of the Caribbean: At World's End
        2  266.936095  1158.437625    165.0                     The Dark Knight Rises
        3  277.613539   303.387927    132.0                               John Carter
        4  305.028724  1053.261376    139.0                              Spider-Man 3

           critics_pick  is_action  is_adventure  is_fantasy  is_science_fiction  \
        0           1.0          1             1           1                   1
        1           0.0          1             1           1                   0
        2           1.0          1             0           0                   0
        3           0.0          1             1           0                   1
        4           0.0          1             1           1                   0

           is_crime       ...        is_war  is_music  is_documentary  rating_g  \
        0         0       ...             0         0               0         0
        1         0       ...             0         0               0         0
        2         1       ...             0         0               0         0
        3         0       ...             0         0               0         0
        4         0       ...             0         0               0         0

           rating_nc_seventeen  rating_pg  rating_pg_thirteen  rating_r  cast_score  \
        0                    0          0                   1         0    1.141077
        1                    0          0                   1         0    1.943331
        2                    0          0                   1         0    3.147587
        3                    0          0                   1         0    1.339893
        4                    0          0                   1         0    1.381308

           director_score
        0        0.836364
        1        0.400000
        2        0.709091
        3        0.000000
        4        0.490909
```

```
[5 rows x 30 columns]
```

1.7 EDA

In [26]: main_df.describe()

Out[26]:

```
            budget      revenue      runtime  critics_pick    is_action  \
count  2035.000000  2035.000000  2035.000000   2035.000000  2035.000000
mean     54.405731   162.734096   108.094840      0.138084     0.258968
std      53.526064   238.730589    18.560045      0.345072     0.438176
min       0.000000     0.000015    46.000000      0.000000     0.000000
25%      16.446526    25.164840    95.000000      0.000000     0.000000
50%      37.632211    78.458092   105.000000      0.000000     0.000000
75%      77.706677   198.331272   118.000000      0.000000     1.000000
max     414.154688  3185.238655   254.000000      1.000000     1.000000

       is_adventure   is_fantasy  is_science_fiction     is_crime  \
count   2035.000000  2035.000000         2035.000000  2035.000000
mean       0.186241     0.099754            0.120885     0.158722
std        0.389397     0.299746            0.326073     0.365507
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

          is_drama       ...            is_war     is_music  is_documentary  \
count  2035.000000       ...       2035.000000  2035.000000     2035.000000
mean      0.441769       ...          0.019165     0.032924        0.014742
std       0.496720       ...          0.137137     0.178481        0.120548
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%       1.000000       ...          0.000000     0.000000        0.000000
max       1.000000       ...          1.000000     1.000000        1.000000

          rating_g  rating_nc_seventeen    rating_pg  rating_pg_thirteen  \
count  2035.000000          2035.000000  2035.000000         2035.000000
mean      0.019656             0.000983     0.108600            0.296806
std       0.138849             0.031342     0.311213            0.456963
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            1.000000
max       1.000000             1.000000     1.000000            1.000000

          rating_r   cast_score  director_score
```

```
count  2035.000000  2035.000000     2035.000000
mean      0.320393     1.083142        0.072986
std       0.466742     0.770948        0.209531
min       0.000000     0.000000        0.000000
25%       0.000000     0.407677        0.000000
50%       0.000000     1.056600        0.000000
75%       1.000000     1.636858        0.000000
max       1.000000     3.307431        1.000000

[8 rows x 29 columns]
```

In [ ]:
```
sum_df = main_df.apply(np.sum,axis = 0)
#genres pie chart
plt.pie(sum_df[5:-7],labels = sum_df.index[5:-7])
plt.show()
```

In [ ]:
```
#mpaa ratings pie chart
plt.pie(sum_df[-7:-2],labels = sum_df.index[-7:-2])
plt.show()
```

In [ ]:
```
sns.boxplot(data = main_df[['revenue','budget']])
plt.show()
```

In [ ]:
```
sns.boxplot(data = main_df[['budget']])
plt.show()
```

In [ ]:
```
sns.set(style="ticks", color_codes=True)
#plt.scatter(main_df['budget'],main_df['revenue'])
sns.pairplot(main_df[['budget','revenue','runtime','director_score','cast_score','criti
```

In [ ]:
```
corr = main_df.corr()
plt.figure(figsize = (28,28))
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(2
```

1.8 Feature Analysis

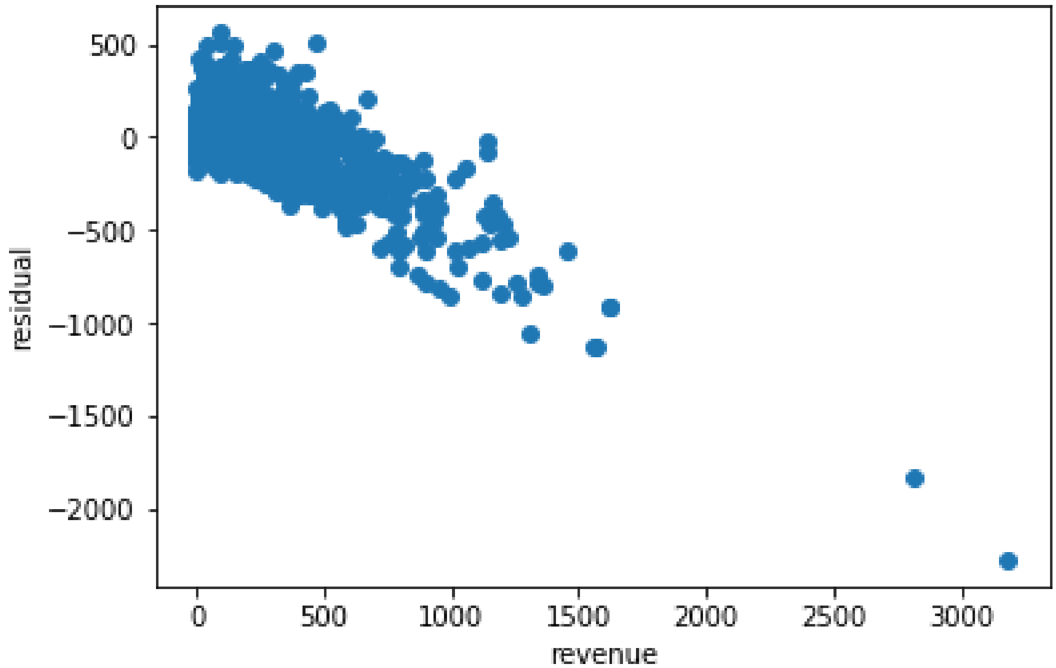
In [118]:
```
formula = '''revenue ~
                    budget + runtime + critics_pick
                    + is_action + is_adventure+is_fantasy+is_science_fiction + is_cri
                    + cast_score + director_score
                    '''
results = smf.ols(formula, data=main_df).fit()
print(results.summary())
print(results.summary())
plt.scatter(main_df['revenue'], results.fittedvalues - main_df['revenue'])
plt.xlabel('revenue')
plt.ylabel('residual')
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.500
Model:                            OLS   Adj. R-squared:                  0.494
Method:                 Least Squares   F-statistic:                     87.39
Date:                Mon, 29 Oct 2018   Prob (F-statistic):         5.65e-282
Time:                        16:49:56   Log-Likelihood:                -13324.
No. Observations:                2035   AIC:                         2.670e+04
Df Residuals:                    2011   BIC:                         2.683e+04
Df Model:                          23
Covariance Type:            nonrobust
======================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
Intercept           -174.6824     30.933     -5.647      0.000    -235.347    -114.018
budget                 2.3961      0.103     23.338      0.000       2.195       2.597
runtime                1.7769      0.283      6.275      0.000       1.222       2.332
critics_pick          40.7434     11.550      3.528      0.000      18.092      63.395
is_action            -14.3928     11.155     -1.290      0.197     -36.269       7.484
is_adventure          38.9542     12.024      3.240      0.001      15.374      62.534
is_fantasy             3.4313     13.605      0.252      0.801     -23.249      30.112
is_science_fiction     7.0768     13.014      0.544      0.587     -18.446      32.599
is_crime              -5.8771     11.657     -0.504      0.614     -28.739      16.985
is_drama             -43.6419      9.731     -4.485      0.000     -62.727     -24.557
is_thriller          -17.5477     10.747     -1.633      0.103     -38.624       3.529
is_animation         114.8815     21.011      5.468      0.000      73.677     156.086
is_family              7.5511     15.930      0.474      0.636     -23.689      38.791
is_western          -152.0575     36.854     -4.126      0.000    -224.333     -79.782
is_comedy             -1.1345      9.990     -0.114      0.910     -20.726      18.457
is_romance            22.5176     10.532      2.138      0.033       1.862      43.173
is_horror             29.4117     15.429      1.906      0.057      -0.847      59.671
is_mystery            -6.6904     14.905     -0.449      0.654     -35.922      22.541
is_history           -99.1458     26.411     -3.754      0.000    -150.941     -47.351
is_war               -28.9469     29.215     -0.991      0.322     -86.242      28.348
is_music              -1.9619     21.613     -0.091      0.928     -44.349      40.425
is_documentary        13.7423     32.835      0.419      0.676     -50.652      78.136
cast_score            13.2025      5.905      2.236      0.025       1.621      24.784
director_score        72.6542     19.889      3.653      0.000      33.650     111.659
==============================================================================
Omnibus:                     1597.218   Durbin-Watson:                   1.424
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            74477.217
Skew:                           3.277   Prob(JB):                         0.00
Kurtosis:                      31.903   Cond. No.                     1.24e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.24e+03. This might indicate that there are
```

strong multicollinearity or other numerical problems.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.500
Model:                            OLS   Adj. R-squared:                  0.494
Method:                 Least Squares   F-statistic:                     87.39
Date:                Mon, 29 Oct 2018   Prob (F-statistic):          5.65e-282
Time:                        16:49:56   Log-Likelihood:                -13324.
No. Observations:                2035   AIC:                         2.670e+04
Df Residuals:                    2011   BIC:                         2.683e+04
Df Model:                          23
Covariance Type:            nonrobust
======================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------
Intercept           -174.6824     30.933     -5.647      0.000    -235.347    -114.018
budget                 2.3961      0.103     23.338      0.000       2.195       2.597
runtime                1.7769      0.283      6.275      0.000       1.222       2.332
critics_pick          40.7434     11.550      3.528      0.000      18.092      63.395
is_action            -14.3928     11.155     -1.290      0.197     -36.269       7.484
is_adventure          38.9542     12.024      3.240      0.001      15.374      62.534
is_fantasy             3.4313     13.605      0.252      0.801     -23.249      30.112
is_science_fiction     7.0768     13.014      0.544      0.587     -18.446      32.599
is_crime              -5.8771     11.657     -0.504      0.614     -28.739      16.985
is_drama             -43.6419      9.731     -4.485      0.000     -62.727     -24.557
is_thriller          -17.5477     10.747     -1.633      0.103     -38.624       3.529
is_animation         114.8815     21.011      5.468      0.000      73.677     156.086
is_family              7.5511     15.930      0.474      0.636     -23.689      38.791
is_western          -152.0575     36.854     -4.126      0.000    -224.333     -79.782
is_comedy             -1.1345      9.990     -0.114      0.910     -20.726      18.457
is_romance            22.5176     10.532      2.138      0.033       1.862      43.173
is_horror             29.4117     15.429      1.906      0.057      -0.847      59.671
is_mystery            -6.6904     14.905     -0.449      0.654     -35.922      22.541
is_history           -99.1458     26.411     -3.754      0.000    -150.941     -47.351
is_war               -28.9469     29.215     -0.991      0.322     -86.242      28.348
is_music              -1.9619     21.613     -0.091      0.928     -44.349      40.425
is_documentary        13.7423     32.835      0.419      0.676     -50.652      78.136
cast_score            13.2025      5.905      2.236      0.025       1.621      24.784
director_score        72.6542     19.889      3.653      0.000      33.650     111.659
==============================================================================
Omnibus:                     1597.218   Durbin-Watson:                   1.424
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            74477.217
Skew:                           3.277   Prob(JB):                         0.00
Kurtosis:                      31.903   Cond. No.                     1.24e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
[2] The condition number is large, 1.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```




```
In [120]: formula = '''revenue ~
                        budget + runtime + critics_pick
                        + is_action + is_adventure+is_fantasy+is_science_fiction + is_cri
                        + cast_score + director_score

                        + is_action * cast_score
                        + is_science_fiction * cast_score
                        + is_crime * cast_score
                        + is_drama * cast_score
                        + is_thriller * cast_score
                        + is_western * cast_score
                        + is_comedy * cast_score
                        + is_romance * cast_score
                        + is_horror * cast_score
                        + is_mystery * cast_score
                        + is_history * cast_score
                        + is_war * cast_score
                        + is_music * cast_score
                        + is_documentary * cast_score
                        + is_adventure * cast_score
                        + is_fantasy * cast_score
```

```
+ is_animation * cast_score

+ budget * cast_score
+ budget * director_score
+ budget * runtime
+ budget * critics_pick
+ is_action * budget
+ is_science_fiction * budget
+ is_crime * budget
+ is_drama * budget
+ is_thriller * budget
+ is_western * budget
+ is_comedy * budget
+ is_romance * budget
+ is_horror * budget
+ is_mystery * budget
+ is_history * budget
+ is_war * budget
+ is_music * budget
+ is_documentary * budget
+ is_adventure * budget
+ is_fantasy * budget
+ is_animation * budget

+ director_score * cast_score
+ director_score * runtime
+ budget * director_score
+ director_score * critics_pick
+ is_action * director_score
+ is_science_fiction * director_score
+ is_crime * director_score
+ is_drama * director_score
+ is_thriller * director_score
+ is_western * director_score
+ is_comedy * director_score
+ is_romance * director_score
+ is_horror * director_score
+ is_mystery * director_score
+ is_history * director_score
+ is_war * director_score
+ is_music * director_score
+ is_documentary * director_score
+ is_adventure * director_score
+ is_fantasy * director_score
+ is_animation * director_score

+ critics_pick * cast_score
+ critics_pick * runtime
```

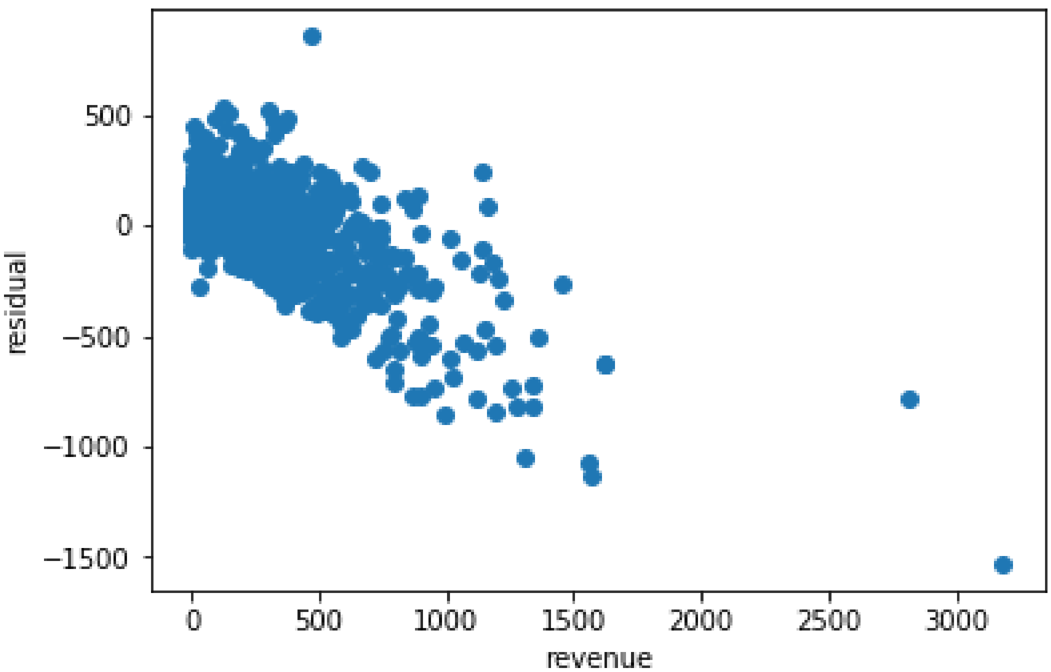
```
                     + budget * critics_pick
                     + director_score * critics_pick
                     + is_action * critics_pick
                     + is_science_fiction * critics_pick
                     + is_crime * critics_pick
                     + is_drama * critics_pick
                     + is_thriller * critics_pick
                     + is_western * critics_pick
                     + is_comedy * critics_pick
                     + is_romance * critics_pick
                     + is_horror * critics_pick
                     + is_mystery * critics_pick
                     + is_history * critics_pick
                     + is_war * critics_pick
                     + is_music * critics_pick
                     + is_documentary * critics_pick
                     + is_adventure * critics_pick
                     + is_fantasy * critics_pick
                     + is_animation * critics_pick

                     + np.power(budget,2)
                     + np.power(cast_score,2)
                     + np.power(director_score,2)
                     '''
results = smf.ols(formula, data=main_df).fit()
print(results.summary())
plt.scatter(main_df['revenue'], results.fittedvalues - main_df['revenue'])
plt.xlabel('revenue')
plt.ylabel('residual')
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.587
Model:                            OLS   Adj. R-squared:                  0.565
Method:                 Least Squares   F-statistic:                     26.87
Date:                Mon, 29 Oct 2018   Prob (F-statistic):          1.84e-297
Time:                        16:52:36   Log-Likelihood:                -13131.
No. Observations:                2035   AIC:                         2.647e+04
Df Residuals:                    1932   BIC:                         2.705e+04
Df Model:                         102
Covariance Type:            nonrobust
=================================================================================
                                  coef    std err          t      P>|t|      [0.025
---------------------------------------------------------------------------------
Intercept                      -8.2496     99.306     -0.083      0.934    -203.008
```

budget	-2.1288	0.559	-3.810	0.000	-3.225
runtime	-0.2940	0.429	-0.686	0.493	-1.134
critics_pick	-39.3401	54.169	-0.726	0.468	-145.576
is_action	-19.1678	19.741	-0.971	0.332	-57.883
is_adventure	3.8144	23.893	0.160	0.873	-43.045
is_fantasy	3.3479	27.052	0.124	0.902	-49.707
is_science_fiction	-60.0700	22.555	-2.663	0.008	-104.305
is_crime	8.5133	21.461	0.397	0.692	-33.577
is_drama	-3.4285	17.106	-0.200	0.841	-36.977
is_thriller	-19.3470	18.869	-1.025	0.305	-56.353
is_animation	103.0374	40.301	2.557	0.011	23.999
is_family	25.6001	15.494	1.652	0.099	-4.786
is_western	38.2231	111.015	0.344	0.731	-179.499
is_comedy	8.3522	17.392	0.480	0.631	-25.756
is_romance	-26.3208	19.007	-1.385	0.166	-63.598
is_horror	22.2913	23.284	0.957	0.338	-23.373
is_mystery	54.1982	26.514	2.044	0.041	2.199
is_history	-4.9612	68.923	-0.072	0.943	-140.133
is_war	-11.2830	59.616	-0.189	0.850	-128.201
is_music	31.9027	36.164	0.882	0.378	-39.021
is_documentary	-33.8198	46.652	-0.725	0.469	-125.314
cast_score	17.9364	18.532	0.968	0.333	-18.409
director_score	13.5848	195.846	0.069	0.945	-370.506
is_action:cast_score	-3.8687	16.646	-0.232	0.816	-36.514
is_science_fiction:cast_score	-10.8369	18.494	-0.586	0.558	-47.108
is_crime:cast_score	-3.1351	17.118	-0.183	0.855	-36.706
is_drama:cast_score	-17.1004	13.536	-1.263	0.207	-43.647
is_thriller:cast_score	15.7519	14.853	1.061	0.289	-13.378
is_western:cast_score	-15.9765	75.175	-0.213	0.832	-163.408
is_comedy:cast_score	-16.1764	12.947	-1.249	0.212	-41.568
is_romance:cast_score	9.4579	14.379	0.658	0.511	-18.743
is_horror:cast_score	-41.4428	26.065	-1.590	0.112	-92.562
is_mystery:cast_score	16.8417	19.446	0.866	0.387	-21.296
is_history:cast_score	27.9812	37.018	0.756	0.450	-44.619
is_war:cast_score	-6.7768	48.423	-0.140	0.889	-101.743
is_music:cast_score	-4.8048	29.006	-0.166	0.868	-61.691
is_documentary:cast_score	2.9443	74.484	0.040	0.968	-143.133
is_adventure:cast_score	32.1859	16.659	1.932	0.054	-0.486
is_fantasy:cast_score	12.5476	18.065	0.695	0.487	-22.881
is_animation:cast_score	-58.2943	23.195	-2.513	0.012	-103.783
budget:cast_score	0.4574	0.141	3.234	0.001	0.180
budget:director_score	0.7532	0.471	1.599	0.110	-0.171
budget:runtime	0.0305	0.005	5.948	0.000	0.020
budget:critics_pick	1.1313	0.305	3.703	0.000	0.532
is_action:budget	0.3774	0.237	1.593	0.111	-0.087
is_science_fiction:budget	0.8013	0.231	3.464	0.001	0.348
is_crime:budget	-0.2863	0.305	-0.939	0.348	-0.884
is_drama:budget	-0.3219	0.217	-1.481	0.139	-0.748

is_thriller:budget	0.0315	0.221	0.142	0.887	-0.402
is_western:budget	-2.1261	0.496	-4.291	0.000	-3.098
is_comedy:budget	0.2390	0.191	1.250	0.212	-0.136
is_romance:budget	0.8532	0.284	3.009	0.003	0.297
is_horror:budget	0.4877	0.448	1.089	0.276	-0.391
is_mystery:budget	-1.1558	0.368	-3.141	0.002	-1.878
is_history:budget	-1.3964	0.638	-2.189	0.029	-2.648
is_war:budget	0.0040	0.650	0.006	0.995	-1.270
is_music:budget	-0.6909	0.700	-0.987	0.324	-2.064
is_documentary:budget	1.8970	4.371	0.434	0.664	-6.676
is_adventure:budget	-0.0337	0.225	-0.150	0.881	-0.475
is_fantasy:budget	-0.0558	0.232	-0.240	0.810	-0.511
is_animation:budget	1.4158	0.335	4.225	0.000	0.759
director_score:cast_score	-21.8845	28.898	-0.757	0.449	-78.560
director_score:runtime	-0.8575	1.147	-0.748	0.455	-3.107
director_score:critics_pick	72.8374	49.261	1.479	0.139	-23.772
is_action:director_score	-50.3250	60.495	-0.832	0.406	-168.967
is_science_fiction:director_score	0.9002	73.489	0.012	0.990	-143.226
is_crime:director_score	-68.6221	52.481	-1.308	0.191	-171.547
is_drama:director_score	-139.2089	54.035	-2.576	0.010	-245.181
is_thriller:director_score	-10.3933	50.815	-0.205	0.838	-110.052
is_western:director_score	-9.0702	166.514	-0.054	0.957	-335.636
is_comedy:director_score	-73.0845	54.208	-1.348	0.178	-179.397
is_romance:director_score	88.5034	61.002	1.451	0.147	-31.133
is_horror:director_score	-137.2064	107.712	-1.274	0.203	-348.450
is_mystery:director_score	35.4801	60.641	0.585	0.559	-83.449
is_history:director_score	-141.2301	93.069	-1.517	0.129	-323.756
is_war:director_score	91.9619	93.462	0.984	0.325	-91.336
is_music:director_score	-23.2919	130.199	-0.179	0.858	-278.637
is_documentary:director_score	1.976e-13	1.57e-13	1.261	0.207	-1.1e-13
is_adventure:director_score	11.6648	64.095	0.182	0.856	-114.037
is_fantasy:director_score	-51.3585	69.780	-0.736	0.462	-188.211
is_animation:director_score	-418.6969	119.324	-3.509	0.000	-652.714
critics_pick:cast_score	13.0960	17.077	0.767	0.443	-20.395
critics_pick:runtime	0.8376	0.736	1.138	0.255	-0.606
is_action:critics_pick	-19.3163	40.896	-0.472	0.637	-99.522
is_science_fiction:critics_pick	138.9234	45.713	3.039	0.002	49.271
is_crime:critics_pick	29.5516	35.389	0.835	0.404	-39.854
is_drama:critics_pick	21.0700	29.846	0.706	0.480	-37.464
is_thriller:critics_pick	-33.4994	33.317	-1.005	0.315	-98.840
is_western:critics_pick	67.6626	134.181	0.504	0.614	-195.492
is_comedy:critics_pick	32.0952	27.523	1.166	0.244	-21.882
is_romance:critics_pick	15.4418	28.320	0.545	0.586	-40.098
is_horror:critics_pick	136.4412	65.400	2.086	0.037	8.180
is_mystery:critics_pick	-135.1908	45.644	-2.962	0.003	-224.708
is_history:critics_pick	34.3902	63.104	0.545	0.586	-89.368
is_war:critics_pick	22.5842	81.096	0.278	0.781	-136.460
is_music:critics_pick	0.7064	61.261	0.012	0.991	-119.439

```
is_documentary:critics_pick          60.6578     80.051      0.758      0.449    -96.338      2
is_adventure:critics_pick           -41.6079     35.691     -1.166      0.244   -111.605
is_fantasy:critics_pick              31.4661     42.812      0.735      0.462    -52.497      1
is_animation:critics_pick            24.2466     51.597      0.470      0.638    -76.945      1
np.power(budget, 2)                  -0.0023      0.001     -1.776      0.076     -0.005
np.exp(cast_score)                   -2.1782      2.700     -0.807      0.420     -7.474
np.exp(director_score)              154.0041    109.469      1.407      0.160    -60.685      3
np.exp(critics_pick)                -75.8470     36.991     -2.050      0.040   -148.393
==============================================================================
Omnibus:                     1097.267   Durbin-Watson:                   1.533
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            18149.355
Skew:                           2.159   Prob(JB):                         0.00
Kurtosis:                      16.978   Cond. No.                     1.23e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 3.3e-21. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```




```
In [15]: logitmod = smf.logit(formula = '''critics_pick ~ is_action''', data = main_df).fit()
         print(logitmod.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.388965
```

```
        Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:           critics_pick   No. Observations:                 2035
Model:                          Logit   Df Residuals:                     2033
Method:                           MLE   Df Model:                            1
Date:                Mon, 29 Oct 2018   Pseudo R-squ.:                 0.03115
Time:                        13:01:08   Log-Likelihood:                -791.54
converged:                       True   LL-Null:                       -816.99
                                        LLR p-value:                 9.766e-13
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -1.6015      0.069    -23.239      0.000      -1.737      -1.466
is_action     -1.2789      0.206     -6.206      0.000      -1.683      -0.875
==============================================================================
```

1.9 Predictive Modelling

In [79]:
```
def evaluate_model(model,X,Y):
    if hasattr(model,'coef_'):
        print('Intercept: ' + str(model.intercept_))
        print('\nCoefficients: ' + str([l + ': ' + str(c) for l,c in zip(x_train.colu
        print('\nR-Squared: ' + str(model.score(x_test,y_test)))
    print('\nResidual Plot')
    plt.scatter(Y, model.predict(X) - Y)
    plt.xlabel('revenue')
    plt.ylabel('residual')
    plt.show()

x_df = main_df.drop(columns=['revenue','title'])
x_df.drop(columns=['rating_g' ,'rating_nc_seventeen', 'rating_pg', 'rating_pg_thirtee
y_df = main_df['revenue']
x_train,x_test,y_train,y_test = train_test_split(x_df,y_df,test_size = 0.3,random_sta
x_train.head()
```

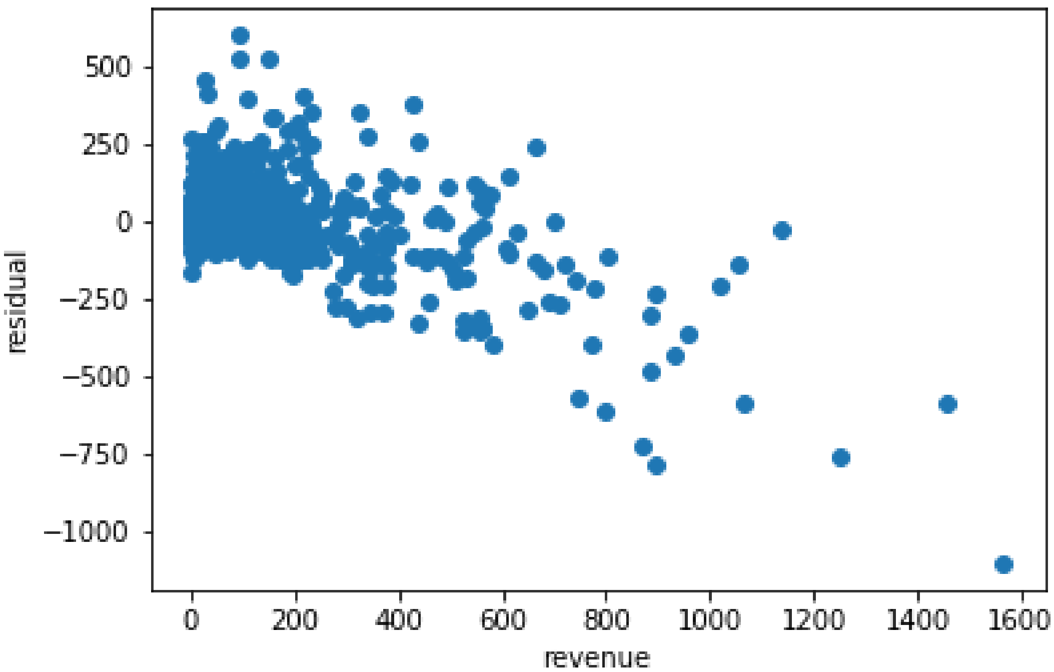
Out[79]:
```
          budget  runtime  critics_pick  is_action  is_adventure  is_fantasy  \
1481   23.442100    101.0           0.0          0             0           1
829    66.220708     82.0           0.0          1             0           0
866    40.325588    113.0           0.0          0             0           0
1942    2.255767    110.0           0.0          0             0           0
210   136.289427    167.0           1.0          0             0           0

      is_science_fiction  is_crime  is_drama  is_thriller      ...        \
1481                   0         0         1            1      ...
829                    1         0         0            0      ...
866                    0         0         1            0      ...
```

```
1942                     0         1         1            0       ...
210                      0         1         1            0       ...

      is_comedy  is_romance  is_horror  is_mystery  is_history  is_war  \
1481          0           0          1           0           0       0
829           0           0          0           0           0       0
866           0           0          0           0           0       0
1942          1           1          0           0           0       0
210           0           0          0           0           1       0

      is_music  is_documentary  cast_score  director_score
1481         0               0    1.056336        0.000000
829          0               0    0.342000        0.000000
866          0               0    1.057400        0.000000
1942         0               0    1.541084        0.000000
210          0               0    3.072537        0.963636

[5 rows x 23 columns]
```

Baseline: OLS with all features, only linear terms

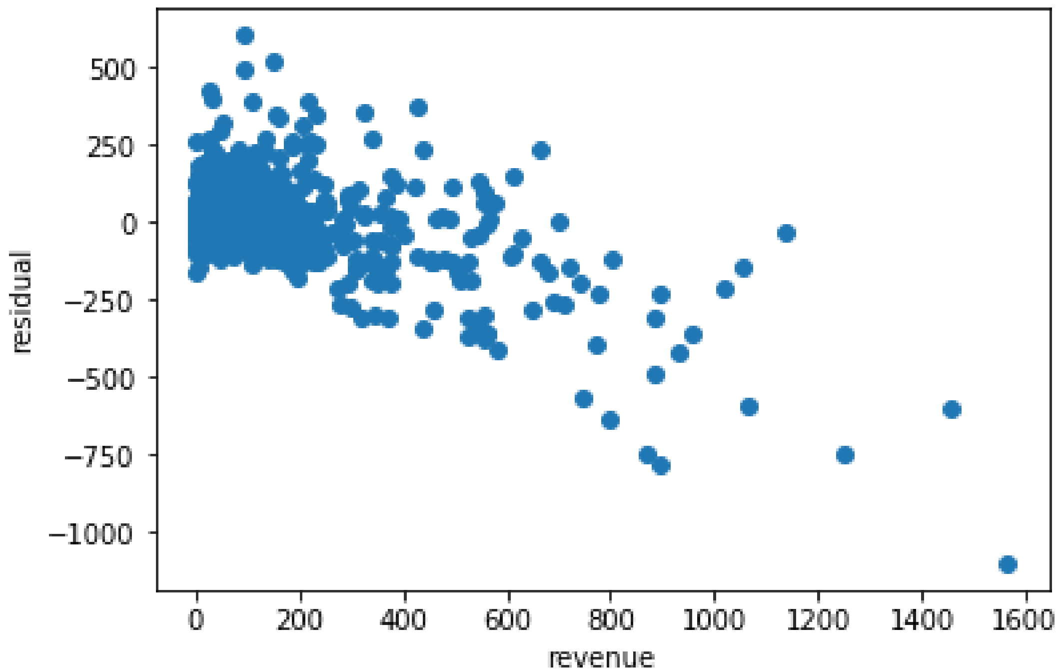
In [80]:
```
ols_model = linear_model.LinearRegression()
ols_model.fit(x_train,y_train)
evaluate_model(ols_model, x_test,y_test)
```

```
Intercept: -234.77809927194082

Coefficients: ['budget: 2.5129658768468626', 'runtime: 2.2433300035453247', 'critics_pick: 30.3

R-Squared: 0.4915047891259611

Residual Plot
```



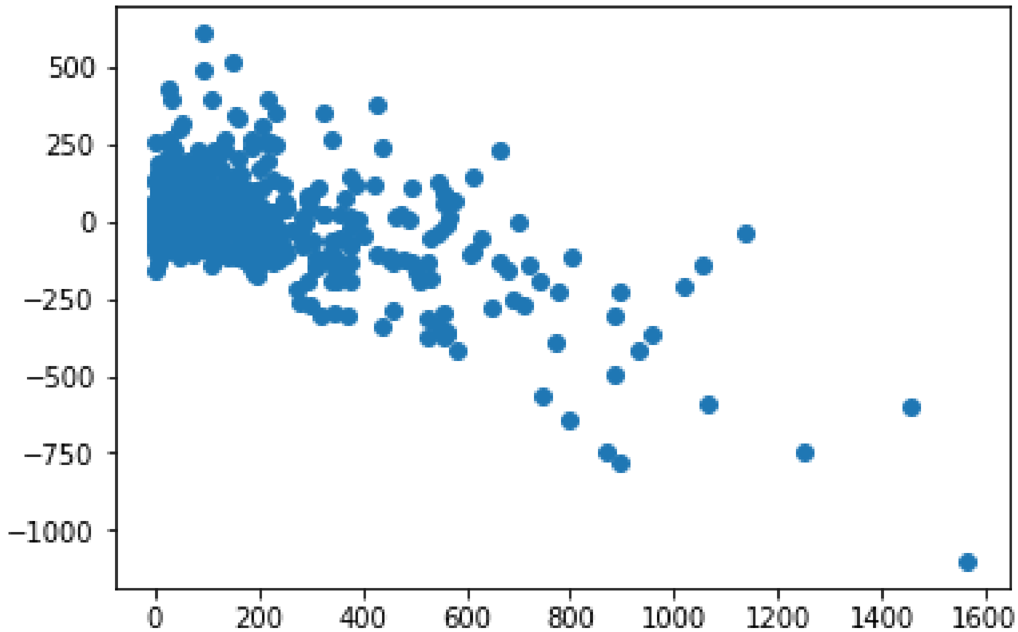

```
In [74]: ridge_model = linear_model.Ridge(alpha=0.5)
         ridge_model.fit(x_train,y_train)
         evaluate_model(ridge_model, x_test,y_test)
```

```
Intercept: -235.0027046494504

Coefficients: ['budget: 2.460874899625821', 'runtime: 2.2629817984809084', 'critics_pick: 33.79

R-Squared: 0.4922255893150719

Residual Plot
```



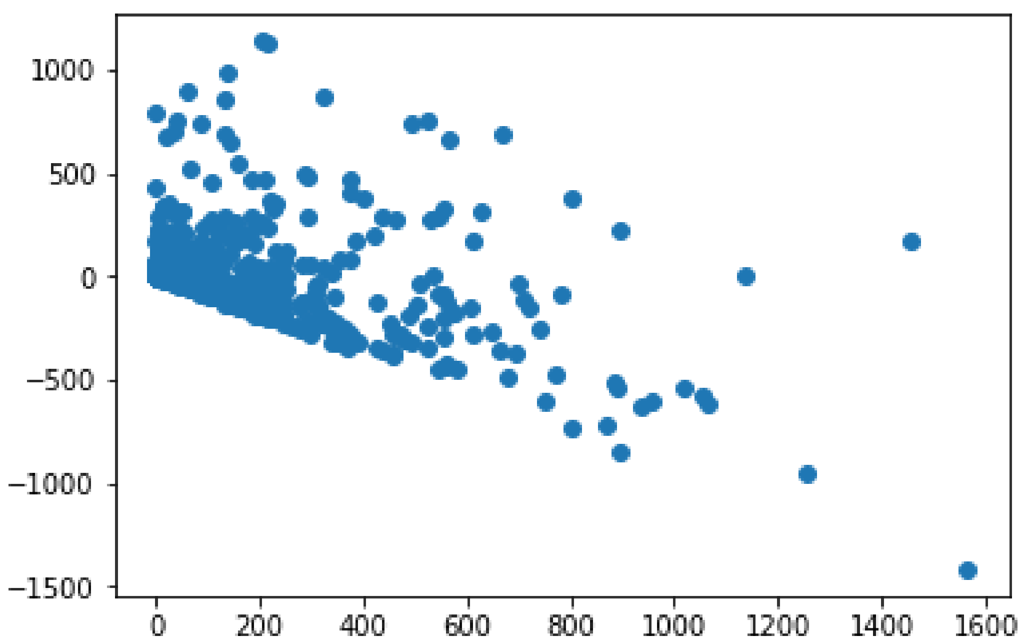

```
In [63]: lasso_model = linear_model.Lasso(alpha=0.1,random_state=0)
         lasso_model.fit(x_train,y_train)
         evaluate_model(lasso_model, x_test,y_test)
```

```
Intercept: -230.73698486192998

Coefficients: ['budget: 2.4686951494712104', 'runtime: 2.2465400373714215', 'critics_pick: 33.5

R-Squared: 0.4928593669381266

Residual Plot
```



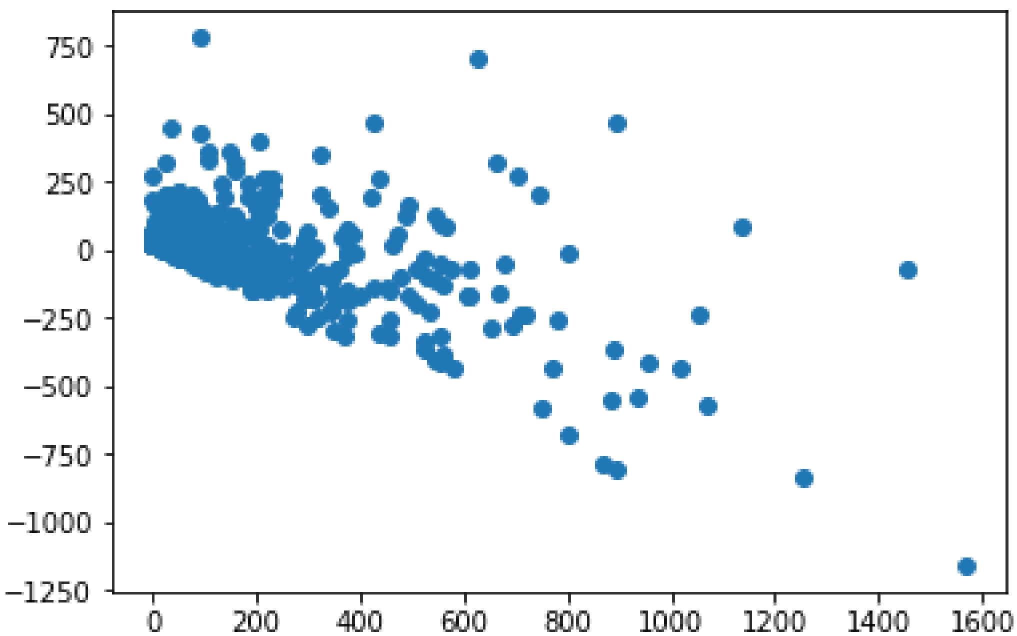

```
In [70]: reg_tree_model = DecisionTreeRegressor(random_state=0)
         reg_tree_model.fit(x_train,y_train)
         evaluate_model(reg_tree_model, x_test,y_test)
```

```
Residual Plot
```



```
In [71]: forest_model = RandomForestRegressor(max_depth=5, random_state=0, n_estimators=100)
         forest_model.fit(x_train,y_train)
         print(forest_model.score(x_test,y_test))
         evaluate_model(forest_model, x_test,y_test)
```

```
0.4746996424538614

Residual Plot
```



1.10 Helper functions

Functions for retrieving data from NYT movies API and OMDB API. Results are saved in csv files.

```
In [ ]: #Codes for scraping, dont run. saved to csv file.
        NYT_API_KEY = '53223e11b006467490bde835d45b0c74'

        all_ny_df = []
        for offset in range(0,8000,20):
            url = 'http://api.nytimes.com/svc/movies/v2/reviews/search.json?opening-date=1990-
            ny_json = pd.read_json(url, orient = 'records')
            ny_df = json_normalize(ny_json['results'])
            if ny_df.empty:
```

```
            break
        all_ny_df.append(ny_df)

    ny_df = pd.concat(all_ny_df)
    print(ny_df.tail())
    ny_df.to_csv('NY Movie Reviews.csv')
```

title = 't=' + nytdata['display_title'][1].replace(' ', '+') req = 'http://www.omdbapi.com/?apikey='+ OMDB_API_KEY + '&'+ title print(pd.read_json(req))

```
In [ ]: OMDB_API_KEY = 'd42886f4'

        def fetch_omdb(title):
            title = 't=' + title.replace(' ', '+')
            print (title)
            req = 'http://www.omdbapi.com/?apikey='+ OMDB_API_KEY + '&'+ title
            omdb_df = pd.read_json(req)
            return omdb_df

        count = 0
        omdb_df_list = []
        for title in tmdb_df['title'].tolist():
            count += 1
            omdb_df_list.append(fetch_omdb(title))
            if count > 5:
                break

        complete = pd.concat(omdb_df_list,axis=0)
        complete.to_csv('omdb_data.csv')

In [ ]:
```