Stephen_Project#1_Proposal

October 28, 2018

0.1 Analysis of Major Revenue Driving Factors in American Movies

by Stephen Gou

Student Number: 1000382908

0.1.1 Questions

1. Is there a statistically significant difference between the mean revenues of NY critic picks and non-picks movies?
2. Do reviews of NY critic picks display a different set of sentiments than that of non-picks?
3. What are the major characteristics of a modern American movie that affect its deomestic lifetime revenue? Modern is defined here as after 1990.

0.1.2 Data Collection

- OMDb API provies a good baseline meta data about movies including release date, runtime, genre, director, writer, actors, production companies, and opening box office. However, several data of interests are missing, e.g. life-time revenue (it only provides opening box office) and budget.
- To supplement OMDb, I found The Movie Dataset on Kaggle: https://www.kaggle.com/rounakbanik/the-movies-dataset#movies_metadata.csv This data was collected from TMDB. It contains revenue, budget, multiple genre tags, and keywords data.
- NY movie reviews can be accessed through its API.
- Text Blob for sentiment analysis https://textblob.readthedocs.io/en/dev/quickstart.html

0.1.3 EDA

1. Plot the distribution of revenues of critic-picks vs non-picks.
2. Generate frequencies of words with different sentiments.
3. Feature selection and transformation for analyzing movie revenues factors

- Pick out relevant features in the data for predicting the revenues through intuition, e.g. genre, director, keywords and actors.

- Examine the validity of important data like revenue. For example, are these revenue figures inflation adjusted? Are they domestic and lifetime revenues? Validate some revenues with other data source like Box Office Mojo.
- Think about how to represent and transform certain features to be ready for modelling: For instance, actors. One way to use it meaningfully is to pull external references and assign a "popularity score" to each actor.
- Keywords is another example that we have to explore its range and values and figure out how to incoporate it into our model. How many unique ones are in total and how many keywords are associated with each movie? "The Dark Knight Rises" has 21 keywords associated with it, including "dc comics", "terrorist", "gotham city", "catwoman"and etc. Some words like "dc comics" might offer very good predictive value since it's associated with many movies, while others like "gotham city" and "catman" might be too specific. Here we might need to plot a histogram of frequencies of all popular keywords.

0.1.4 Analysis

1. Is there a statistically significant difference between the mean revenues of NY critic picks and non-picks movies?

```
Conduct a t-test on the mean revenue of critic-picks and determine if there's statistically sig
```

2. Do reviews of NY critic picks display a different set of sentiments than that of non-picks?

```
Compare the top most-frequent key words to picks vs non-picks.
```

3. What are the major characteristics of a modern American movie that affect its deomestic lifetime revenue? - Construct a linear regression model that fits a movie's features to its revenue. Categorical features like genre tags, and key words will be one hot encoded. Reason about possible interaction terms and include them in the model. - Examine coefficients and their corresponding p-values to identify the most influential features that drive revenue. - Finally, test for likely confounders. For instance, genre might affect a movie's revenue and the type of directors at the same time. - Try random forest of regression trees and compare performance

0.2 Introduction

0.3 Methods

0.4 Cleaning

0.5 EDA

0.6 Feature Selection and Mapping

0.7 Results

0.8 Conclusion

Code for importing, basic trimming and observation of data from TMDB and OMDB.

In [20]:
```
import json
import requests
import numpy as np
import pandas as pd
from pandas.io.json import json_normalize

github_raw_root = 'https://raw.githubusercontent.com/gouzhen1/Moives-Data-Analysis/mas
#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']]
```

```python
#TMDB Main Dataset
tmdb_df = pd.read_csv(github_raw_root + 'tmdb_5000_movies.csv')
tmdb_df['release_date'] = pd.to_datetime(tmdb_df['release_date'])
tmdb_df.drop(tmdb_df[tmdb_df['release_date'].dt.year < 1990].index, inplace=True)
tmdb_df = tmdb_df[tmdb_df['revenue'] > 0]
tmdb_df = tmdb_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
tmdb_df['production_countries'] = tmdb_df.apply(process_country, axis = 1)
tmdb_df = tmdb_df[tmdb_df['production_countries'] =='United States of America']
tmdb_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']
        result.append(genre_name)
        if genre_name not in genre_dict:
            genre_dict[genre_name] = 1
        else:
            genre_dict[genre_name] += 1

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

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

#inflation adjust
```

```
        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(df):
            year = df['release_date'].year
            revenue = df['revenue']
            return cpi_dict['2017-01-01']/cpi_dict['{}-01-01'.format(year)] * revenue

        tmdb_df['revenue_a'] = tmdb_df.apply(get_cpi_adjusted,axis=1)
        tmdb_df = tmdb_df.drop(columns = ['release_date','original_language','popularity','hom
```

```
In [21]: print(tmdb_df.shape)
        tmdb_df = tmdb_df.merge(tmdb_credits_df,how='left')
        tmdb_df.columns = map(str.lower, tmdb_df.columns)
        tmdb_df.to_csv('wrangled_dataset.csv')
        tmdb_df.describe()
        tmdb_df.head()
```

```
(2033, 29)
```

```
Out[21]:       budget  runtime                                    title  critics_pick  \
        0  237000000    162.0                                   Avatar           1.0
        1  300000000    169.0  Pirates of the Caribbean: At World's End           0.0
        2  250000000    165.0                    The Dark Knight Rises           1.0
        3  260000000    132.0                              John Carter           0.0
        4  258000000    139.0                             Spider-Man 3           0.0

           is_action  is_adventure  is_fantasy  is_science fiction  is_crime  \
        0          1             1           1                   1         0
        1          1             1           1                   0         0
        2          1             0           0                   0         1
        3          1             1           0                   1         0
        4          1             1           1                   0         0

           is_drama       ...        is_documentary  rating_g  rating_nc-17  \
        0         0       ...                     0         0             0
        1         0       ...                     0         0             0
        2         1       ...                     0         0             0
        3         0       ...                     0         0             0
        4         0       ...                     0         0             0

           rating_not rated  rating_pg  rating_pg-13  rating_r     revenue_a  \
        0                 0          0             1         0  3.185239e+09
        1                 0          0             1         0  1.136173e+09
        2                 0          0             1         0  1.158438e+09
        3                 0          0             1         0  3.033879e+08
        4                 0          0             1         0  1.053261e+09
```

```
   cast_score  director_score
0    1.141077        0.836364
1    1.943331        0.400000
2    3.147587        0.709091
3    1.339893        0.000000
4    1.381308        0.490909

[5 rows x 31 columns]
```

0.8.1 Analysis and Modelling

```
In [57]: import statsmodels.api as sm
         import statsmodels.formula.api as smf

         results = smf.ols('revenue_a ~ is_documentary', data=tmdb_df).fit()
         print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              revenue_a   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     8.932
Date:                Wed, 24 Oct 2018   Prob (F-statistic):            0.00284
Time:                        23:25:05   Log-Likelihood:                -42139.
No. Observations:                2035   AIC:                         8.428e+04
Df Residuals:                    2033   BIC:                         8.429e+04
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept       1.647e+08   5.32e+06     30.945      0.000    1.54e+08    1.75e+08
is_documentary  -1.31e+08   4.38e+07     -2.989      0.003   -2.17e+08    -4.5e+07
==============================================================================
Omnibus:                     1725.246   Durbin-Watson:                   0.740
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            63319.468
Skew:                           3.816   Prob(JB):                         0.00
Kurtosis:                      29.239   Cond. No.                         8.30
==============================================================================

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

```
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=1990-0
    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 [ ]: df = pd.DataFrame(np.random.randn(3,2), columns=['A', 'B'])
        print(df.head())
        print(type(df.iloc[0]))
        print(type(df['A']))
```