

Movie Financial Success Predictor

Milind Kulkarni

March 1, 2017

Table of Contents

1. Background and problem statement.....	2
2. Exploratory Data Analysis.....	2
3. Data clean up	7
4. Setting the Prediction target.....	12
5. Machine learning	13
6. Summary of Results.....	16
7. Conclusion	17

1. Background and problem statement

The basic idea of this project is to use datapoints like cast, director, date of release, budget, genre, run time etc. to predict whether or not a movie will be financially successful.

Thousands of movies are produced in America every year. These movies require a huge amount of investment upfront. So, a tool which will be able to predict the financial success of a movie would be very useful.

The final success of a movie does depend on the content, story and quality of production of the individual movie. But there are macro trends in what factors determine the success of a movie. For instance, the quality of production can be estimated by factors like the number of Facebook likes for the director and actors. There is a possibility that a director who was good in his previous movies may not be able to recreate his success but we are assuming that there is a good degree of correlation between past and future success.

Prospective Clients:

This project will help movie production houses on deciding whether or not to produce a movie. It might help cast members to make a decision on whether to pick up a particular role or not.

2. Exploratory Data Analysis

The dataset being used in this project is the IMDB 5K dataset. There are other datasets like the movie lens data set which has 27K movies. However, that might be too heavy to run on a local machine. So, I decided to use the smaller dataset for prototyping, once the concept is proven it can easily be expanded to a larger data set.

This dataset has 27 features. Some of the features are imdb score, budget, gross, color/black and white, content rating, genre etc.

```
In [29]: print(titles.columns)
```

```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',  
      'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',  
      'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',  
      'num_voted_users', 'cast_total_facebook_likes', 'actor_3_name',  
      'facenumber_in_poster', 'plot_keywords', 'movie_imdb_link',  
      'num_user_for_reviews', 'language', 'country', 'content_rating',  
      'budget', 'title_year', 'actor_2_facebook_likes', 'imdb_score',  
      'aspect_ratio', 'movie_facebook_likes'],  
      dtype='object')
```

I have analyzed the dataset for lots of characteristics to check for distributions of imdb rating, genres, content rating etc. To make sure that the dataset contains a wide variety of data points which is necessary for efficient prediction.

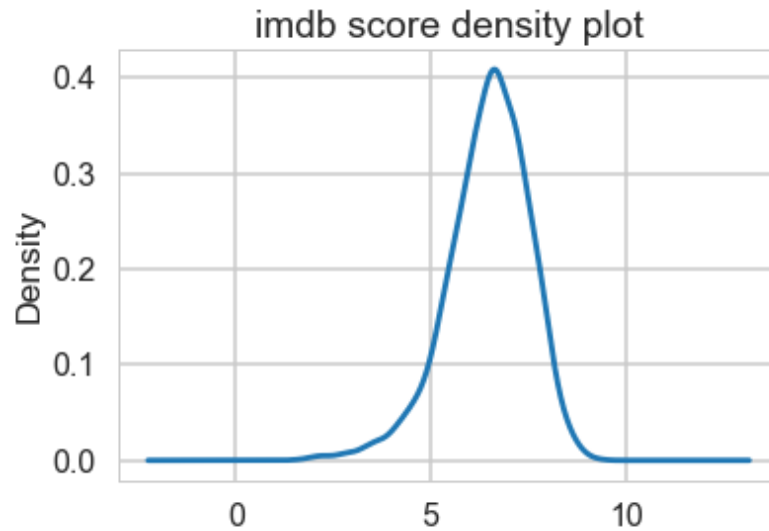


Figure 1

Fig [2] and Fig [3] show that there is a good distribution of Genre's and ratings of movies in the dataset. This is done just as a sanity check measure for the dataset.

As you can see the highest number of movies have an imdb rating between 6 to 7. Majority of movies are in the range of 5.0 to 7.9.

Analyzing imdb score distribution

```
In [33]: ((titles.imdb_score//1)).value_counts()
```

```
Out[33]: 6.0    1459
         7.0    1091
         5.0     769
         4.0     217
         8.0     211
         3.0      64
         2.0      22
         9.0       4
         1.0       3
         Name: imdb_score, dtype: int64
```

```
In [36]: titles.genres.value_counts().plot(kind = 'bar', title='Movie Genre frequency')
```

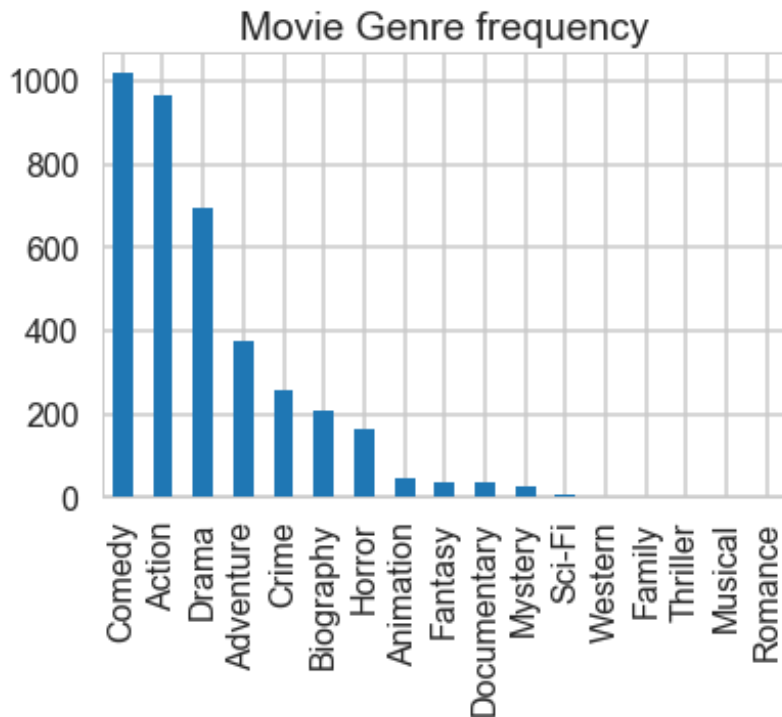


Figure 2

```
In [37]: titles.content_rating.value_counts().plot(kind = 'bar', title='Movie Rating frequency')
```

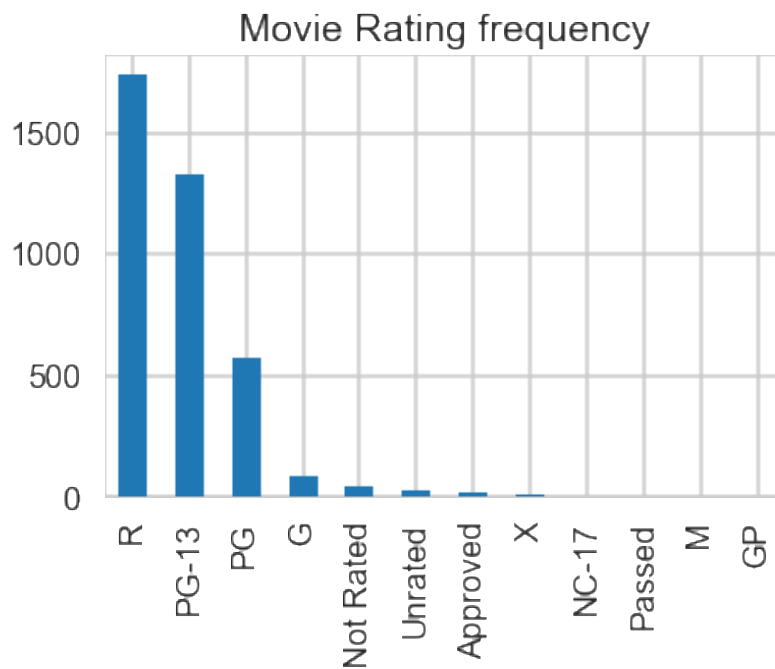


Figure 3

As we can see in the above figures, the top three primary genres are Comedy, Action and Drama. Also the top three ratings are R, PG-13 and PG. These trends make sense.

Next, we will try to get a feel for the amount of profit made by movies. Fig [4] plots the profit distribution of movies. Profits are depicted as ratio between gross and budget.

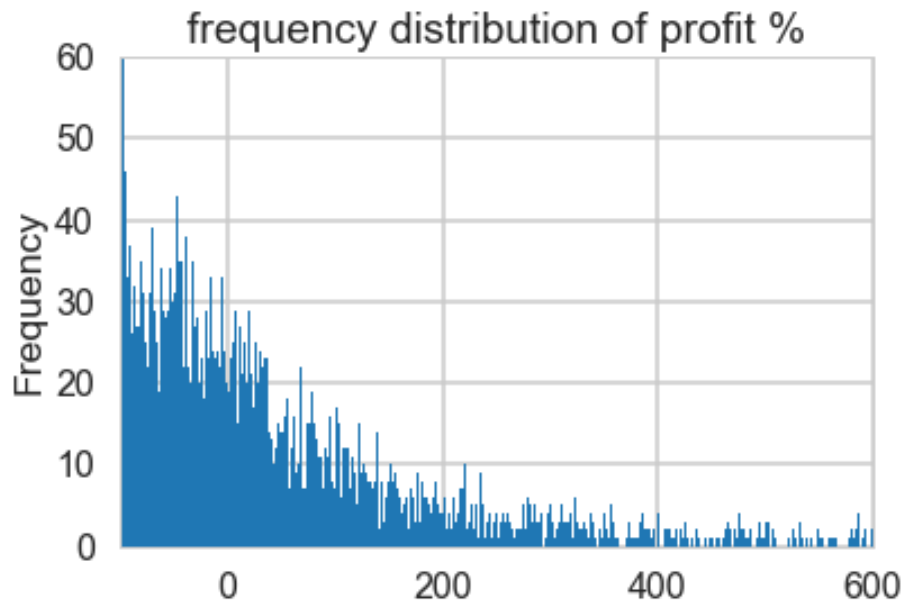


Figure 4

Some vital statistics of the dataset. The mean profit ratio is 2.7x (170%), however the median profit is just 8%.

```
In [43]: titles.ratio.mean() , titles.ratio.std() , titles.ratio.median()
```

```
Out [43]: (2.7023612667579786, 8.60812467633687, 1.081657044736842)
```

Fig [5] shows the distribution between the imdb score and profit ratio, as expected the profit ratio does show a good deal of correlation with the imdb score.

```
In [70]: titles[(titles.ratio<10)].plot.scatter(x='imdb_score',y='ratio',s=10)
```

```
Out [70]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2354f5c0>
```

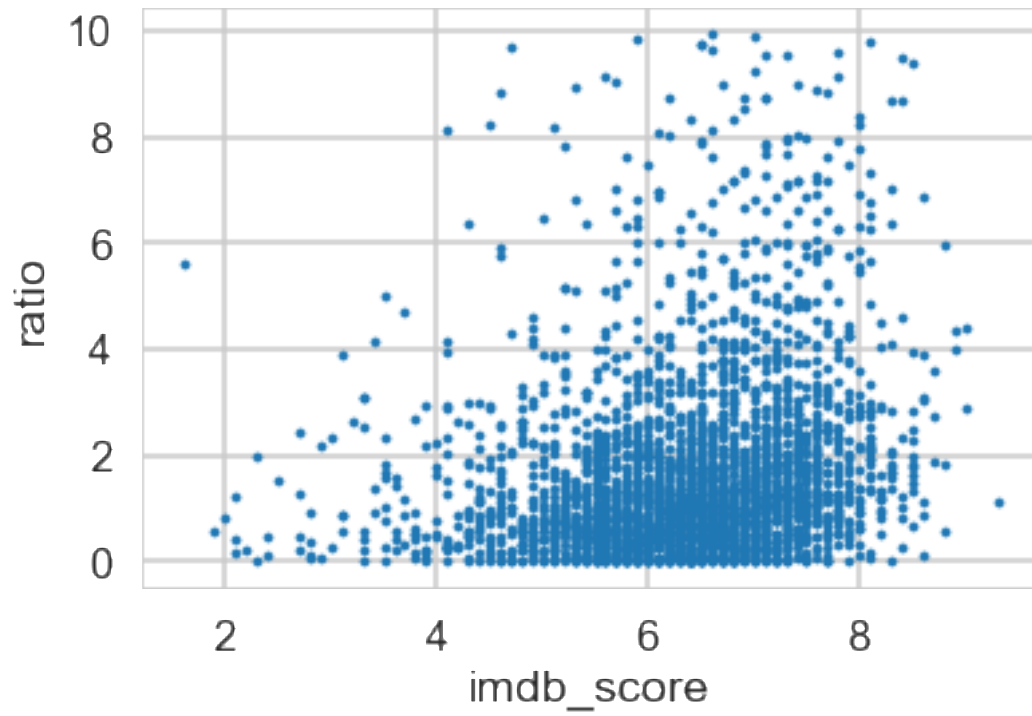


Figure 5

3. Data clean up

The main purpose of doing this is to help the Machine Learning predictors to make better predictions. So we are looking for outliers and faulty data points.

- As we can see below there are 8 movies which make more than 100x profit. These are certainly outliers and have to be filtered out in order to improve the predictions.

**** looking for outliers in profit ratio. To improve efficiency of the model ****

```
In [39]: (titles.ratio / 100).value_counts()
```

```
Out [39]: 0.0      3823
          1.0        9
          2.0         2
          3.0         2
          71.0        1
          4.0         1
          27.0        1
          23.0        1
          Name: ratio, dtype: int64
```

removing movies with a very high profit ratio (>200x) for which are mostly low budget movies

```
In [40]: titles=titles.drop(titles[(titles.ratio>200)].index)
```

```
In [41]: titles.ratio.mean(), titles.ratio.std()
```

```
Out [41]: (2.7023612667579786, 8.60812467633687)
```

- I. Next, let us look at the budgets of the movies. And try to find if there are any outliers in this feature and try to filter any stray data points.

Below are the distribution of profits in 100 Million dollars. As we can see there are a few stray points above 500 Million dollars. These points seem to be incorrect as there aren't any movies with that high of a budget as of 2018. So we will need to investigate into those data points. And try to remove them from the dataset

```
In [32]: temp=(titles.budget//.5e8)*.5).value_counts().sort_index()

In [37]: temp.plot.bar(title='Profit distribution in 100 Million dollars')

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1a265fd588>
```

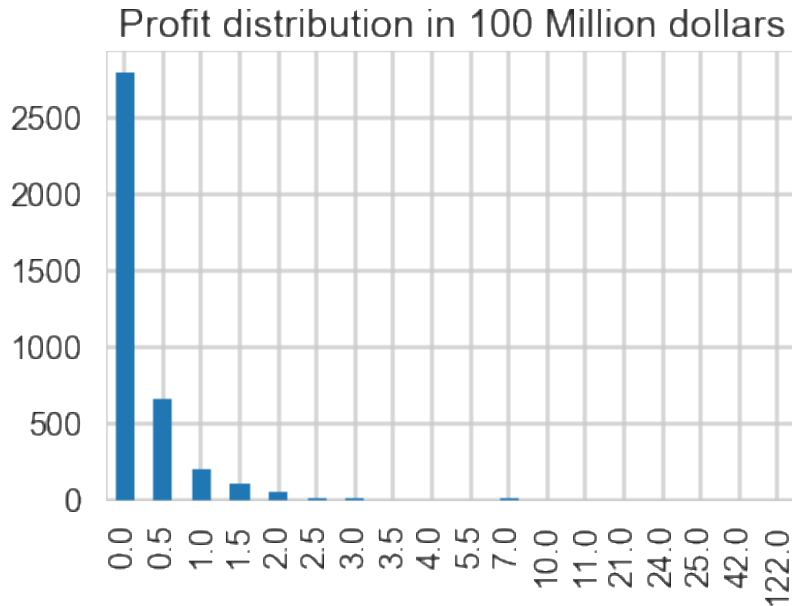


Figure 6

Showing the same data in a different format below.

```
In [26]: ((titles.budget//1e8)).value_counts().sort_index()

Out[26]: 0.0      3464
         1.0      300
         2.0       54
         3.0        3
         4.0         1
         5.0         1
         7.0         2
        10.0         1
        11.0         1
        21.0         1
        24.0         1
        25.0         1
        42.0         1
        122.0        1
         Name: budget, dtype: int64
```

Again we can see movies with budget of over 500 Million dollars which are outliers. So we will look at the country of origin of these highly expensive movies. As we can see below all of the movies are non-US movies. And their budgets and gross might be reported in the currency of their country of origin.

Looks like there are a few outliers which will have to be filtered out later On further investigation these movies are not made in the USA and do not have their budget statistics in USD.

```
In [25]: titles[titles.budget>.33e9].country.value_counts()
```

```
Out[25]: Japan          4
         South Korea    2
         India          1
         China          1
         Hungary        1
         Thailand        1
         Spain          1
         France         1
         Name: country, dtype: int64
```

So we will go ahead and remove all foreign movies hopefully this should not decrease the size of the dataset too much. As we can see there are 788 foreign movie which is about 15% of the movies. This is something that we can tolerate. In the future we can convert the amounts to USD but for now we will go ahead with US movies only.

```
In [26]: len(titles[titles.country != "USA"])
```

```
Out[26]: 788
```

**** Removing foreign movies from the Dataset. Maybe add currency conversion in the future****

```
In [27]: titles=titles.drop(titles[titles.country != "USA"].index)
```

II. Now let's look at the correlation matrix for this dataset.

The column of interest in this case is the ratio vs the other features. As that's what determines the financial success of a movie. As we can see in this correlation chart the ratio correlates very strongly with the IMDB rating, sadly that is a feature that we cannot use. That is because IMDB rating for a movie is not known in advance when investing in a movie.

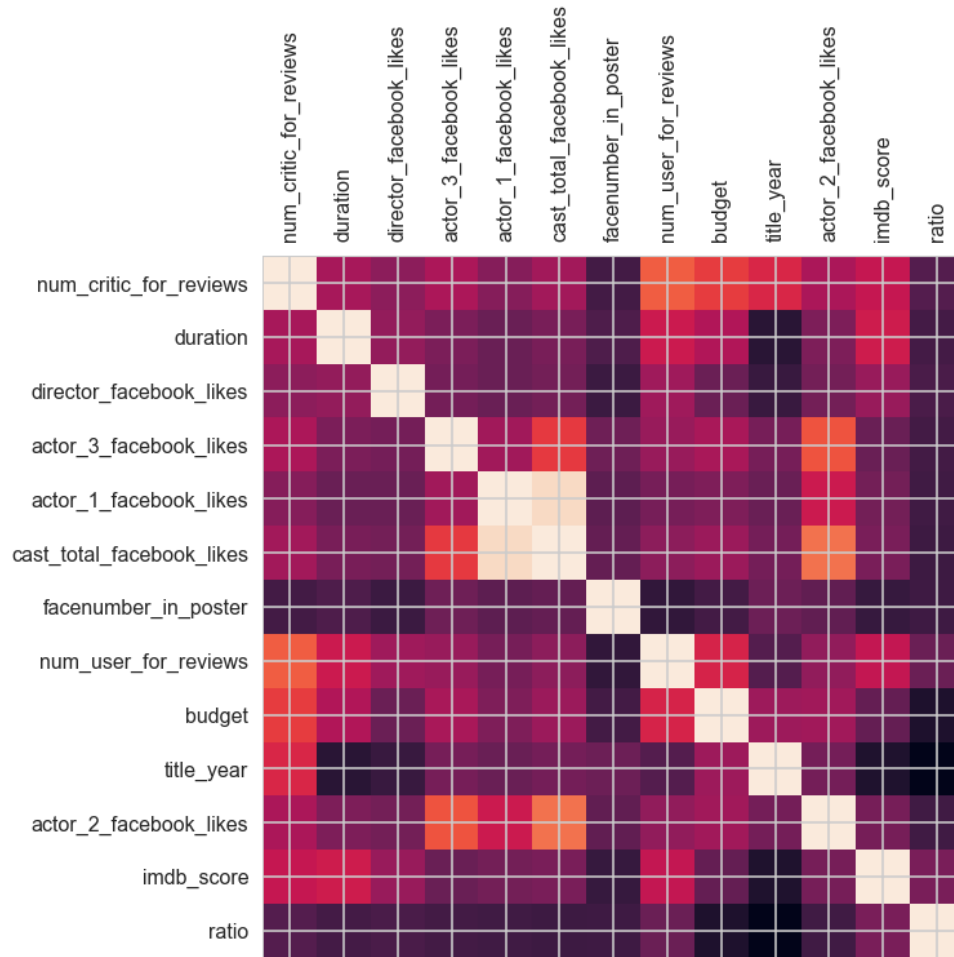


Figure 7

The profit ratio is negatively correlated to the release year of the movie. Looking at the average profit ratio through the years, we can see that profit ratio did on average go down since the 1980's. That might be because only the successful movies from the 1980's might have been included in the dataset.

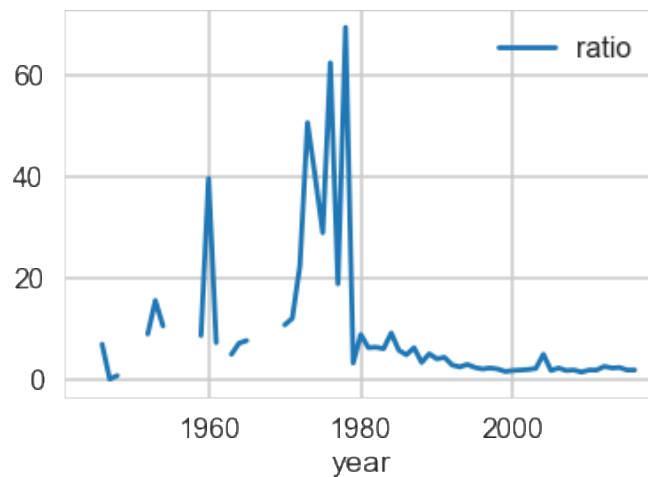


Figure 8

- III. Next let's take a look at the profit distribution of low budget and high budget movies. Usually the factors that influence profits vary between low and high budget movies.

Fig [9] plots the profit distribution frequency. As we can see the high budget movies (in red) a large number of movies just make enough to cover their expenses. A good amount of movies make a decent amount of profit which is up to $\sim 4x$. However after that the profit ratio completely goes down.

For low budget movies. A high number of movies make less than 50% of their budgets. However, when it comes to profits over $2x$ they put their high budget counterparts behind them.

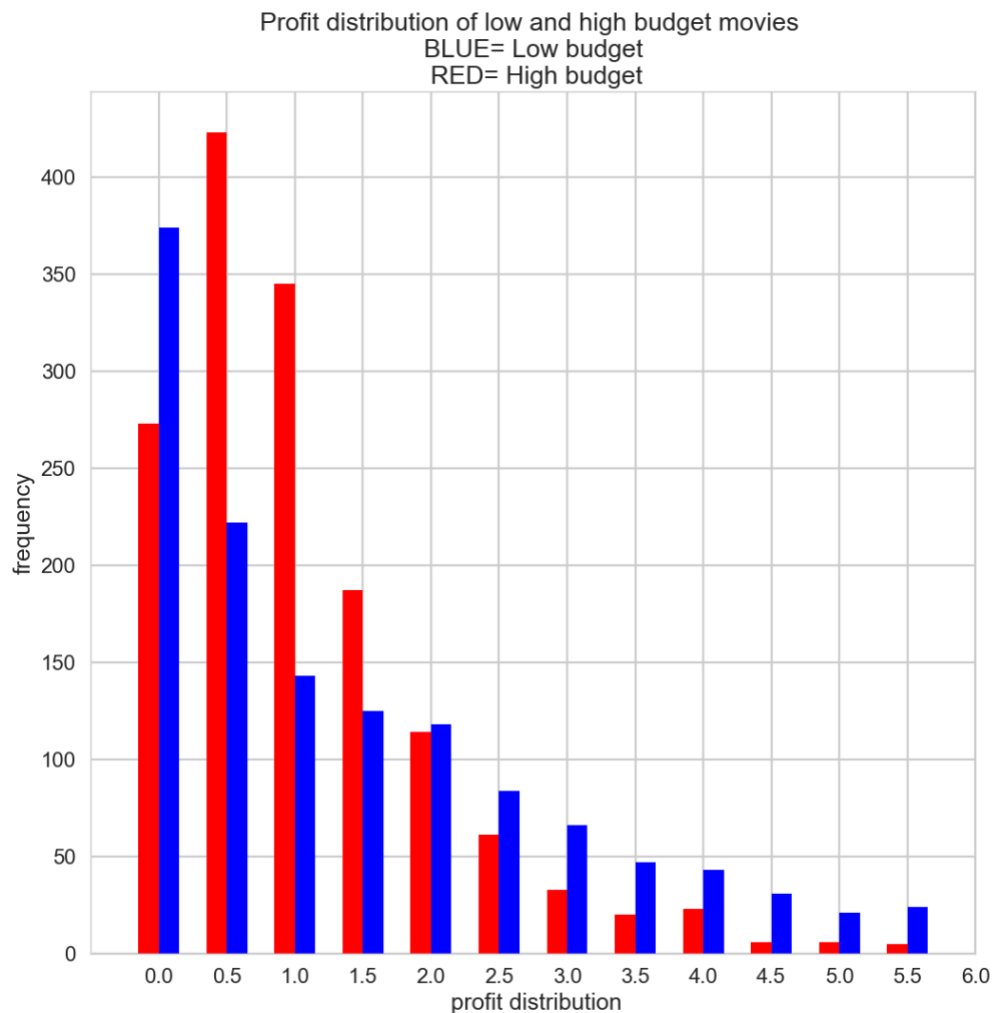


Figure 9

Looking at such stark variation in profit distribution I decided to split the training data set into high and low budget movies. This should improve the prediction accuracy.

Profit Distribution between low and high budget movies was found to be quite different. So splitting them will be beneficial from a point of view of accurately training the predictors

Taking the median as roughly the point that splits the high budget and low budget movies

```
In [48]: titles_highb = titles[(titles.budget > titles.budget.median())]
```

```
In [49]: titles_lowb = titles[(titles.budget <= titles.budget.median())]
```

```
In [50]: len(titles_highb)
```

```
Out [50]: 1508
```

```
In [51]: len(titles_lowb)
```

```
Out [51]: 1527
```

As we can see in the code snippet above. I divided the dataset with the median as the dividing point.

4. Setting the Prediction target.

After cleaning up the data set we need to set a prediction goal. The main aim of this project is predicting the financial success of a movie. So we need to set a target above which a movies is considered “profitable”.

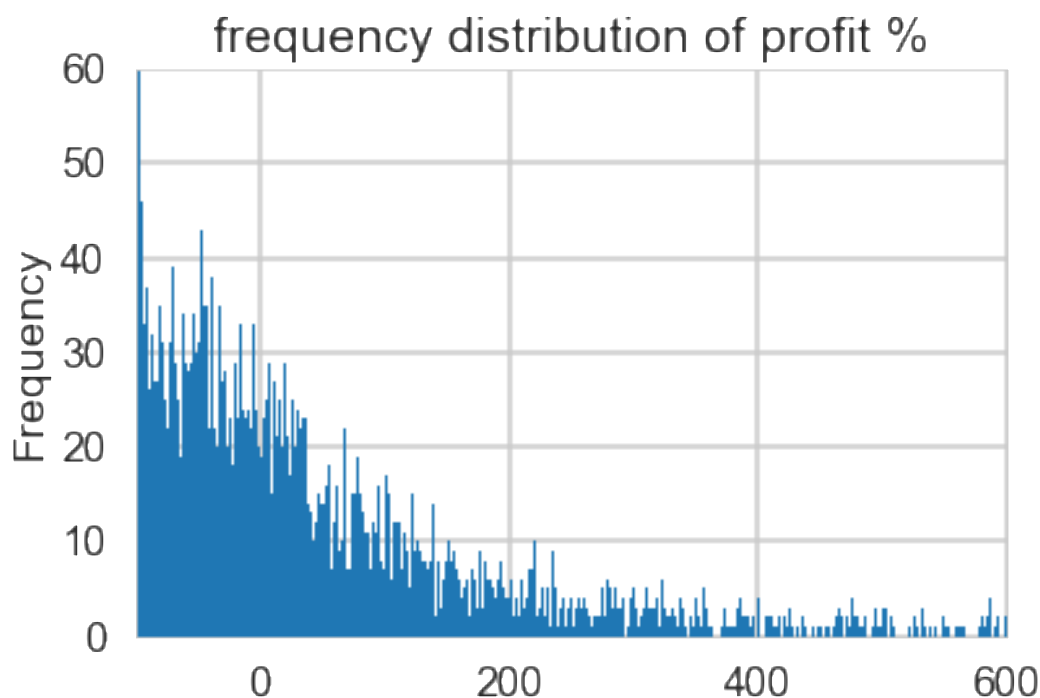


Figure 10

```
In [43]: titles_ratio.mean() , titles_ratio.std() , titles_ratio.median()
```

```
Out [43]: (2.7023612667579786, 8.60812467633687, 1.081657044736842)
```

Looking at the figure above, a 20% profit is a good determinant of whether or not a movie is successful. So, movies with a profit above 20% are profitable and below are considered to be loss making in this predictor.

```
In [44]: titles['profit_making'] = (titles_ratio > 1.2)
```

```
In [45]: titles.profit_making.value_counts()
```

```
Out [45]: False      2059
          True       1773
          Name: profit_making, dtype: int64
```

As we can see above the 20% profitability target roughly splits the dataset into 2 halves.

5. Machine learning

In this section I will walk through all the machine learning techniques that I tried on this dataset. I will also compare the performance of all these prediction techniques. The prediction techniques that I try out are 1. Logistic Regression 2. Support Vector Machines 3. Random Forrest and 4. Voting classifier.

The first step towards this is splitting the dataset into a training and test. I do a 70:30 split between the training and test dataset.

The second part is training the predictor. There are two parameters to keep in mind with this,.

1. Usage of the best possible hyper-parameters for optimal accuracy.
2. Making sure that predictor takes the bias-variation tradeoff into account. Higher bias is when the predictor is over fitted. The predictor works perfectly for the training set however there is a big drop off in accuracy when it is applied on a new data set.

```
def cv_score(clf, x, y, score_func=accuracy_score):
    result = 0
    nfold = 5
    for train, test in KFold(nfold).split(x): # split data into train/t
        clf.fit(x.loc[train], y.loc[train]) # fit
        result += accuracy_score(clf.predict(x.loc[test]), y.loc[test])
    return result / nfold # average
```

I use K-fold cross validation to reduce the bias of a model. The `cv_score` function is used for cross-validation. As we can see in the function above it is a 5-fold cross validation. And the function returns back the average accuracy of the five predictions.

Cross validation reduces the bias. The next step is to get the best possible predictor parameters. I use two techniques to that end.

- 1) The first method is running a loop around all the possible parameter values. As you can see in the code snippet underneath. For the random forest classifier, we run loops around the number of estimators and the number of leaf nodes. And the code finally returns the best hyper parameters.

```
score=0
m=[10,100,1000]
n=[100,1000,10000]
max_nodes_final,n_estim_fianl=0,0
for max_nodes in m:
    for n_estim in n:
        clf=RandomForestClassifier(n_estimators=n_estim,max_leaf_nodes=max_nodes,
n_jobs=-1,random_state=42)
        if score<cv_score(clf,X,Y):
            score=cv_score(clf,X,Y)
            score_val=score
            max_nodes_final=max_nodes
            n_estim_fianl=n_estim
            print(score_val,max_nodes,n_estim)
print(score_val,max_nodes_final,n_estim_fianl)
```

- 2) The second method is using Scikit-learn's inbuilt gridsearch function. This function looks at

```
clf_2=Pipeline([
    ("scaler",StandardScaler()),
    ("svm_clf",SVC(kernel='poly',C=1,degree=2))
])
param_grid=[{'svm_clf__degree':[1,2,3],'svm_clf__C':[.1,1,10,100]},]
grid_search_2=GridSearchCV(clf_2,param_grid,cv=5)
grid_search_2.fit(titles_cat_logistic_highb_train_x,titles_cat_logistic_highb_train_y.profit_making)
```

1. Logistic Regression

The first algorithm used is the Logistic Regression.

```
In [70]: score=0
Cs = [0.001, .01, 0.1, 1, 10, 100, 1000]
C_val=0
for i in Cs:
    clf = LogisticRegression(C=i)
    if (score < cv_score(clf, x, y)):
        score=cv_score(clf, x, y)
        C_val=i
        score_val=score

print(score_val,C_val)

/Users/mrkulkarnil/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)

0.649514811032 10
```

The average prediction accuracy score was 66%. The parameter “C” controls the regularization of the predictor (it’s the inverse of the regularization parameter). Higher values of C can cause overfitting and lower values cause under fitting. So, k-fold cross validation was used for the optimum C value generation. In this case the optimum C value happened to be 10.

2. Support Vector machine.

The second predictor I used was SVM. The hyper parameters in this case were the kernel type (linear, rbf or polynomial), C parameter (inverse of regularization) and degree in case of polynomial kernel.

I used both Kfold cross validation and GridCVsearch for the parameter determination. The best combination was polynomial kernel, degree=2 and C=1.

The average prediction accuracy was 69%. The confusion matrix generated was as follows:

```
[215, 43]
[ 92, 101]
```

3. Random Forest.

Random Forest was also tried out with number of estimators and maximum number of nodes as the main hyper parameters.

Again, using GridCvSearch the optimal parameters was maximum_nodes=100 and n_estimators=1000. With an average prediction accuracy of 72%.

Confusion Matrix:

```
[117, 60]
[ 51, 214]
```

4. Voting Classifier.

Voting classifier as the name suggests runs all the predictors and makes the decides on the basis of the majority of the predictors.

The average prediction accuracy using the three aforementioned predictors was 73%.

Confusion Matrix:

[101, 76]

[38, 227]

6. Summary of Results

As you can infer from the prediction accuracies and the confusion matrices, the best predictor is Voting Classifier Forest (~73% accuracy) followed by Random Forest, SVM and Logistic Regression.

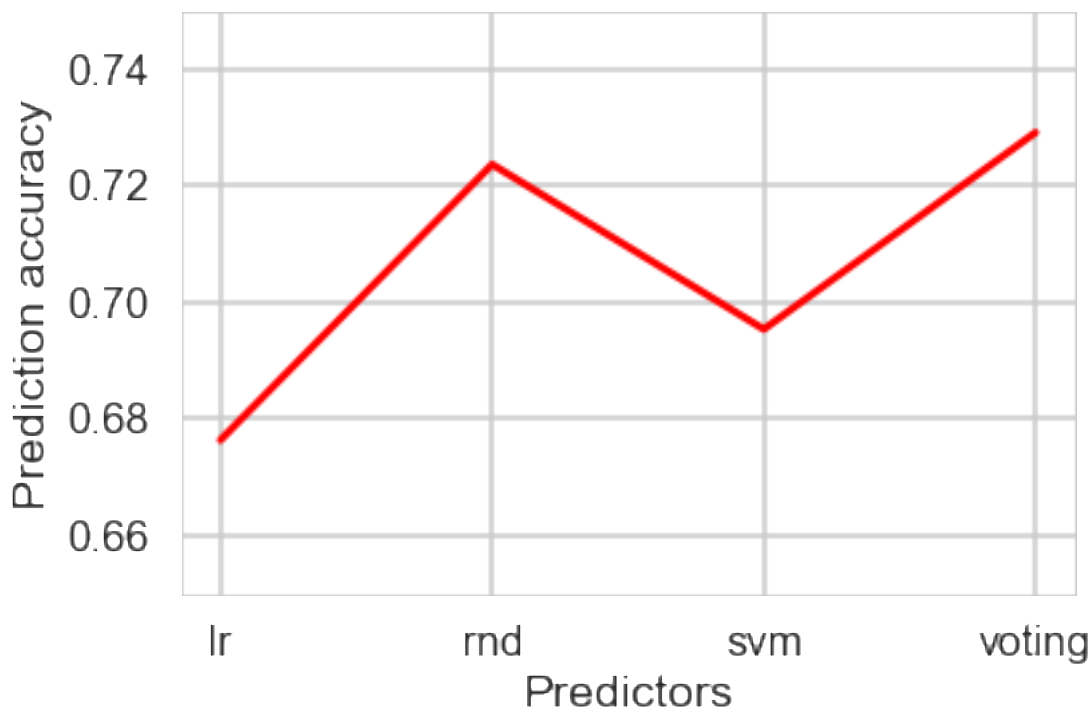


Figure 11

Looking at the F1 scores (which are the harmonic means of precision and recall) in Fig[12]. We can see that the voting classifier has the highest F1 score of .705

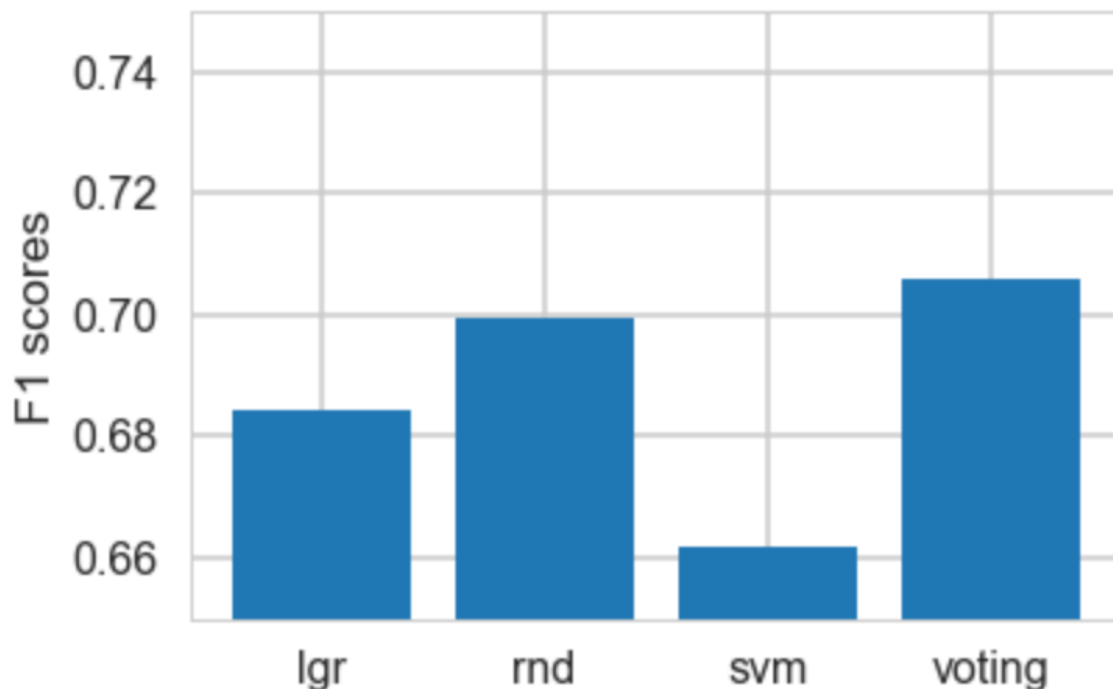


Figure 12

```
Out [303] :      f1_Score  Accuracy
          lgr      0.684225  0.676328
          svm      0.661876  0.695355
          rnd      0.699357  0.723658
          voting    0.705708  0.729134
```

7. Conclusion


So, to summarize we can predict with a good deal of accuracy whether or not a movie will be successful. The most significant features were the year of the movie, the number of Facebook likes for the actors and directors and the number of reviewers for the movie. The voting classifier was able to predict with a 73% accuracy whether or not a movie will be successful.

I think bringing in more data related to month of release and maybe breakdown of budget into production, marketing etc. Might be useful.

Also, another finding of great importance is that adding the IMDB score of the movie in the dataset, does not increase the accuracy by a great deal it bumps the accuracy from 73% to 74%.

Fig[13] . The IMDB score is supposed to track the quality of the movie. So, does this mean that the movie quality is of little consequence in the financial success of Hollywood movies? This question seems to be a area ripe for further analysis.

Predictor scores using IMDB score as a
feature in the dataset



	f1_Score	Accuracy
lgr	0.696655	0.716420
svm	0.685265	0.697437
rnd	0.725501	0.739202
voting	0.725000	0.728908

Figure 13