IMDb Movie Rating Predictions

Xue (Jessica) Jin

Problem Introduction

How can we determine the quality of a movie before it arrives in the cinema? Have you ever wondered how movie ratings would be predicted once released to the public? It takes time to obtain a fair amount of critics review after a movie is released and it is difficult to provide an accurate rating based on limited ratings by users.

We have different features of movies such as Movie Gross, Budget, Genres, Content Rating, Director, Actors and their social media enjoys predicting movie ratings. Thus we can utilize different machine learning models that take in open source data and automatically predicts the score to rate a movie.

1. Dataset

Import Data

```
import pandas as pd
```

```
# Import Data
mov = pd.read_csv("https://raw.githubusercontent.com/sundeepblue/movie_rating_prediction/master/movie_metadata.csv")
print(mov.columns.values)
```

```
['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' 'movie_title'
 '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']
```

This dataset includes 5043 movies with 28 features.

```
mov.shape
```

```
(5043, 28)
```

2. Data Preprocessing

Data Cleaning

- Remove duplicate values

```
# Data Preprocessing
# Remove duplicate values
mov = mov.drop_duplicates()
```

```
mov.shape
```

```
(4998, 28)
```

- Drop unrelated columns

  Throughout the data cleaning, I decided to drop aspect_ratio and the movie_imdb_link since I cannot find how these features could affect the rating of the movie. Speaking about language, most of the movies in our dataset are English. For color, most of the movies in our dataset interoperates the same logic. Therefore language and color can easily be dropped from our dataset as they do not capture much difference.

```
mov = mov.drop(columns = ["aspect_ratio","movie_imdb_link","language","color"])
```

```
mov.shape
```

```
(4998, 24)
```

- Fill gaps

  After removing duplicate values and dropping unrelated columns, there are 4998 movies with 24 features left in the dataset. In fact many features have missing values like gross, content rating and budget. However, if I drop all the missing values, the final analysis will be affected. Thus, I choose to use average to fill in the missing numerical values. Missing Categorical features are dropped from the dataset.

```
# Fill Gaps
# Fill missing values with mean
mean = mov[['num_critic_for_reviews','duration','director_facebook_likes','actor_3_facebook_likes','actor_1_facebook_likes','gro
mov = mov.fillna(mean)
```

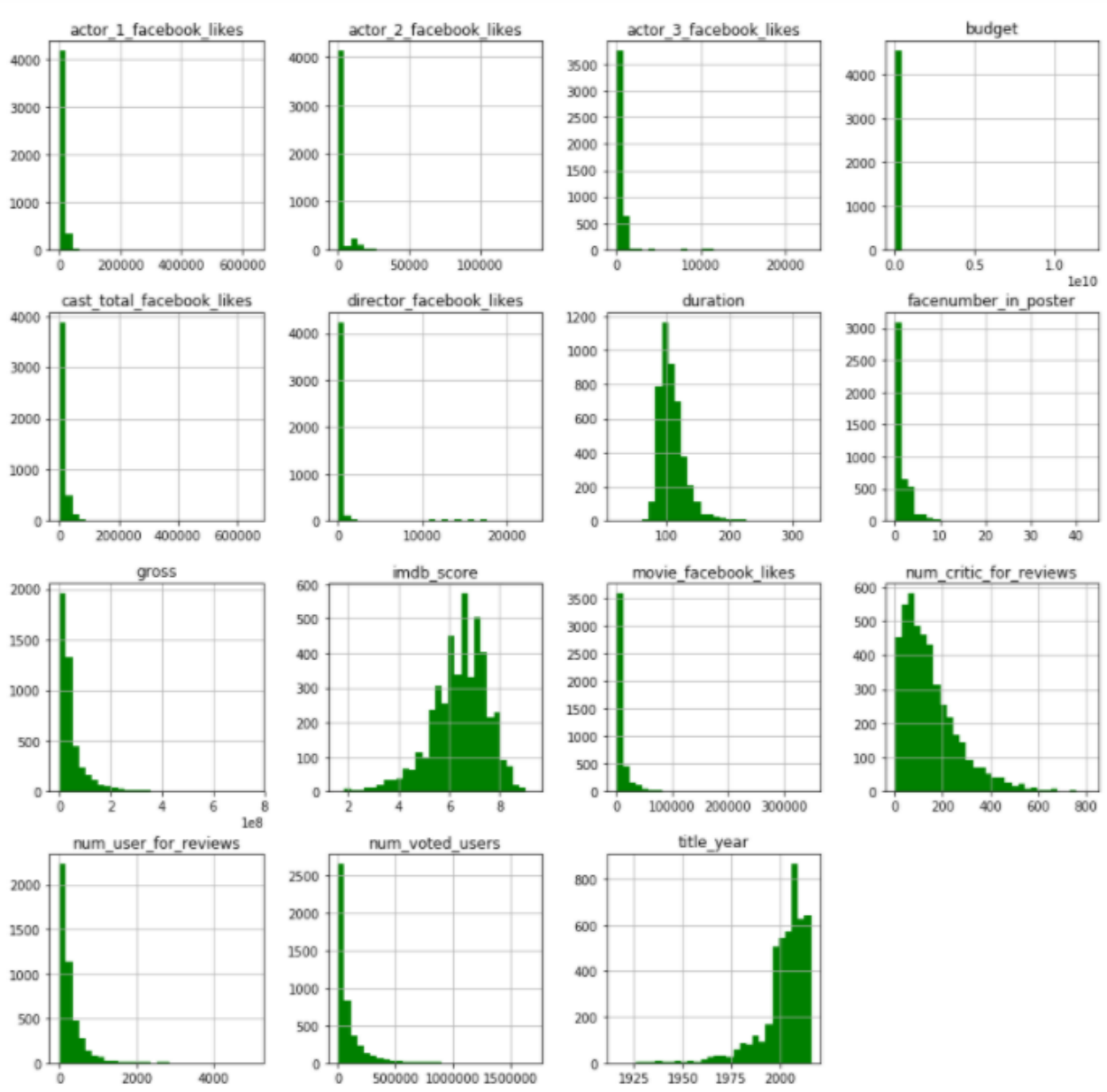
```
mov = mov.dropna()
mov.shape
```

```
(4554, 24)
```

Finally, there are 4554 movies with 24 features in the dataset.

3. Exploratory Data Analysis

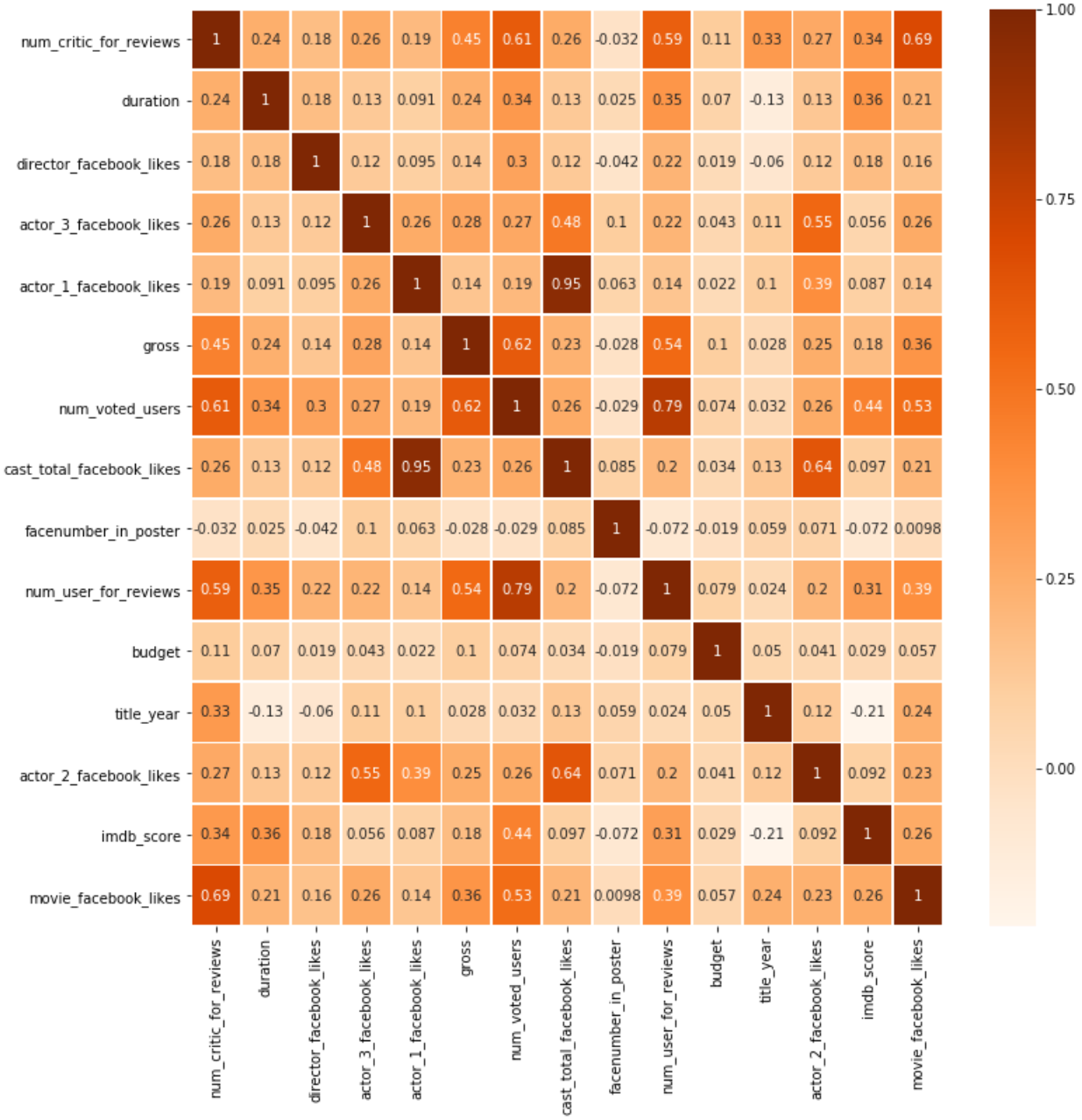
1) Create a histogram of all the features



According to the histogram indicated above, we can figure out the film industry thrived after 2000. Most of the imdb_score are concentrated between 6 and 8.

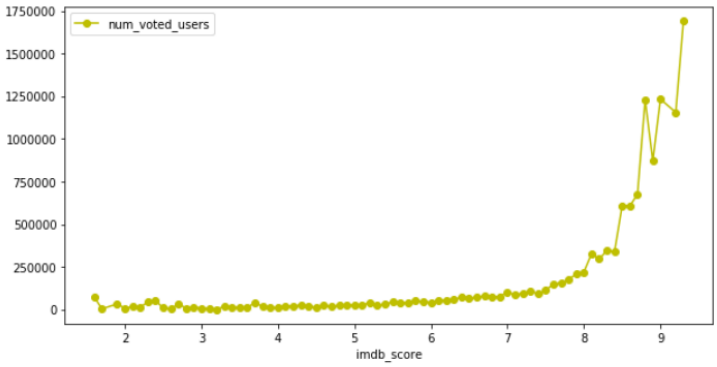
2) Correlations of numerical features against the IMDb Score

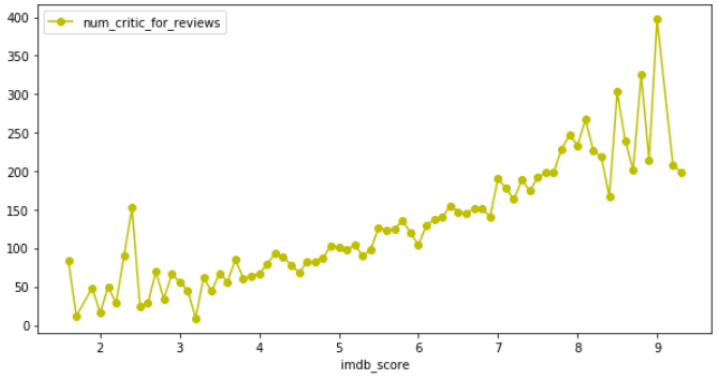
I try to find which numerical features have the most influence on IMDb Score.

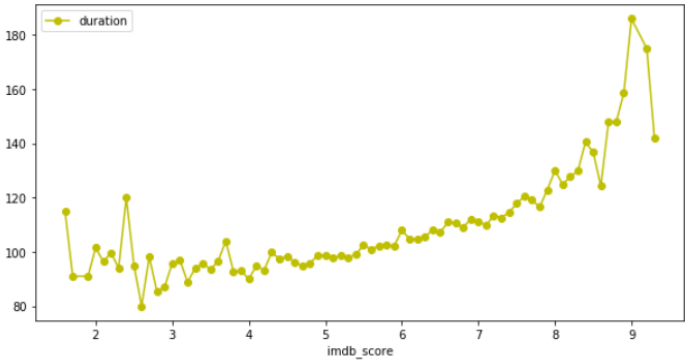


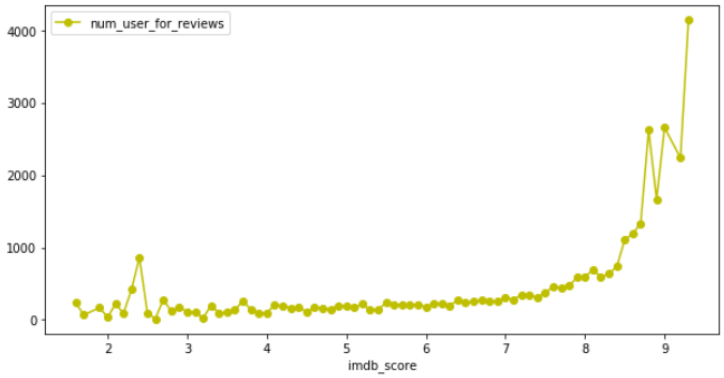
The top five highest correlation of numerical features with “imdb_score” are “num_voted_users”, “duration”, “num_critic_for_reviews”, “num_user_for_reviews”, “movie_facebook_likes”. This five features maybe have the greatest influence on IMDb score.

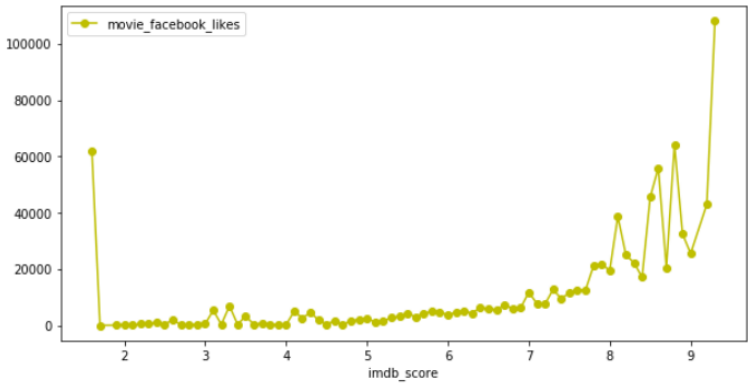
3) Plotting graphs of the top five highest correlation of numerical features against the imdb score











For all the features, it is obvious to see that the more the features, the higher the IMDb score, especially for number of voted users, number of users for reviews and movie facebook likes. For higher IMDb score between 8 and 9, number of critic for reviews and movie facebooks likes don't make so much difference on ratings which means good film is recognized by everyone.

4. Machine Learning Models

I created 3 different models involving **Linear Regression Model**, **Random Forest Model** and **Xgboost Model** to predict IMDB rating of a movie.

4.1 Splitting the Dataset

```
# Create Machine Learning Models to predict IMDB rating of a movie
# Splitting the data for training and testing
from sklearn.model_selection import train_test_split
X = mov_for_ML
y = mov['imdb_score']
X.shape,y.shape
```

```
((4554, 5), (4554,))
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((3643, 5), (3643,), (911, 5), (911,))
```

4.2 Scaling to avoid Euclidean Distance Problem

```
# Scaling to avoid Euclidean Distance Problem
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train.values), columns=X_train.columns, index=X_train.index)
```

```
X_test = pd.DataFrame(scaler.transform(X_test.values), columns = X_train.columns, index = X_test.index)
```

4.3 Machine Learning Models

1) Linear Regression model

```
# Linear regression model
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
lm = LinearRegression()
lm.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
from sklearn.metrics import mean_squared_error
```

```
prec_lm=lm.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
print('The mean squared error using Linear regression is: ',mean_squared_error(y_test,prec_lm))
print('The mean absolute error using Linear regression is: ',mean_absolute_error(y_test,prec_lm))
```

```
The mean squared error using Linear regression is:  0.888146083332038
The mean absolute error using Linear regression is:  0.7212462006471558
```

```
mean_squared_error(prec_lm, y_test)
```

```
0.888146083332038
```

2) Random Forest Model

```
# Random Forest Model
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
```

```
C:\Users\Golden Snow\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:246: FutureWarning: The default value of n_estimato
rs will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

```
prec_rf=rf.predict(X_test)
```

```
print('The mean squared error using Random Forest model is: ',mean_squared_error(y_test,prec_rf))
print('The mean absolute error using Random Forest model is: ',mean_absolute_error(y_test,prec_rf))
```

```
The mean squared error using Random Forest model is:  0.7931176728869374
The mean absolute error using Random Forest model is:  0.6675521405049396
```

```
mean_squared_error(prec_rf, y_test)
```

```
0.7931176728869374
```

3) XGboost Model

```
# XGboost Model
from xgboost import XGBClassifier
Xgb = XGBClassifier()
Xgb.fit(X_train,y_train)
```

```
prec_Xgb=Xgb.predict(X_test)
```

```
print('The mean squared error using the Xgboost model is: ',mean_squared_error(y_test,prec_Xgb))
print('The mean absolute error using the Xgboost model is: ',mean_absolute_error(y_test,prec_Xgb))
```

5. Conclusion

- Budget is important, although there is no strong correlation between budget and movie rating.
- The Facebook popularity of the top 3 actors/actresses is important.
- Number of users that vote for a movie influence the IMDB Score the most, it has the highest correlation with IMDB Score.
- The Random Forest Model has performed the best with MSE of '0.793' and MAE of '0.668' and it would be the model of choice for given data set among various models I tried.
- Try more GBM and nonlinear models such as SVM with kernels and higher order regression fits.