TEAM-11(entertainment)

**Predicting Movie Success**

**Submitted by:**

AJMAL N

NIMYA SUNIL

SINI JACOB

MISHAL M NAINA

Table of Contents

[ABSTRACT 3](#_Toc59373036)

[INTRODUCTION 4](#_Toc59373037)

[Data exploration 5](#_Toc59373038)

[2.1 Describing Data 5](#_Toc59373039)

[2.2 Reading and exploring data 5](#_Toc59373040)

[2.3 Categorizing the target varible 6](#_Toc59373041)

[2.4 Handling the Missing values 8](#_Toc59373042)

[Visualization 9](#_Toc59373043)

[3.1 Correlation 9](#_Toc59373044)

[Model building 12](#_Toc59373045)

[4.1 Splitting and training the data 12](#_Toc59373046)

[4.2 Scaling the data 12](#_Toc59373047)

[Feature scaling 14](#_Toc59373048)

[5.1 Feature Selection using RFECV 14](#_Toc59373049)

[5.2 Random Forest 17](#_Toc59373050)

[Classification Report 18](#_Toc59373051)

[Conclusion 19](#_Toc59373052)

[Future Work 19](#_Toc59373053)

# ABSTRACT

Success of a movie depends upon a lot of factors like good directors or excellent actors or story plotline. However, famous directors and actors can always bring an expected box-office income but cannot guarantee a highly rated imdb score.

In the movie industry, the most common and high-priority question to be answered is “whether a particular movie with the specified cast in the selected genre will be a box office success or not?”. Now a days massive data generated from the search engines has widened the perspective of the market research and analysis in the imdb’s data

With the help of other parameters we will predict whether a movie is Hit, Average or Flop .Its success is based on the available massive IMDB data. The dataset contains 28 variables for 5043 movies, spanning across 100 years in 66 countries. There are 2399 unique director names, and thousands of actors/actresses. “imdb\_score” is the response variable while the other 27 variables are possible predictors.

**CHAPTER 1**

# INTRODUCTION

**IMDb**(Internet Movie Database) is one of the most recognized names for its comprehensive online database collection of movies, films, TV series and so on. As of today (July 2020), you’ll see through the following data pull that IMDb database has approximately 7 million titles.

IMDb, in full Internet Movie Database, Web site that provides information about millions of films and television programs as well as their cast and crew. The name is an acronym for Internet Movie Database. As a wholly owned subsidiary of Amazon.com, IMDb is based in Seattle, but the office of Col Needham, the founder and CEO, remains in Bristol, England, where the Web site was founded.

Subsets of IMDb data are available for access to customers for personal and non-commercial use. You can hold local copies of this data, and it is subject to our terms and conditions. Please refer to the [Non-Commercial Licensing](https://help.imdb.com/article/imdb/general-information/can-i-use-imdb-data-in-my-software/G5JTRESSHJBBHTGX?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=3aefe545-f8d3-4562-976a-e5eb47d1bb18&pf_rd_r=GMPX40VNPC4KBDY9WZ8P&pf_rd_s=center-1&pf_rd_t=60601&pf_rd_i=interfaces&ref_=fea_mn_lk1) and [copyright/license](http://www.imdb.com/Copyright?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=3aefe545-f8d3-4562-976a-e5eb47d1bb18&pf_rd_r=GMPX40VNPC4KBDY9WZ8P&pf_rd_s=center-1&pf_rd_t=60601&pf_rd_i=interfaces&ref_=fea_mn_lk2) and verify compliance. Success of a movie depends upon alot of factors like good directors or excellent actors or story plotline.

However, famous directors and actors can always bring an expected box-office income but cannot guarantee a highly rated imdb score. So we will try to predict an output using the dataset whether a movie will be success or not.

**CHAPTER 2**

# Data exploration

## 2.1 Describing Data

We as a team (nimya and ajmal) selected data from kaggle where the name of dataset is named as “imdb 5000 movie dataset” and the data is coherently large ; Itcontains 28 variables for 5043 movies, spanning across 100 years in 66 countries. There are 2399 unique director names, and thousands of actors/actresses. “imdb\_score” is the response variable while the other 27 variables are possible predictors.

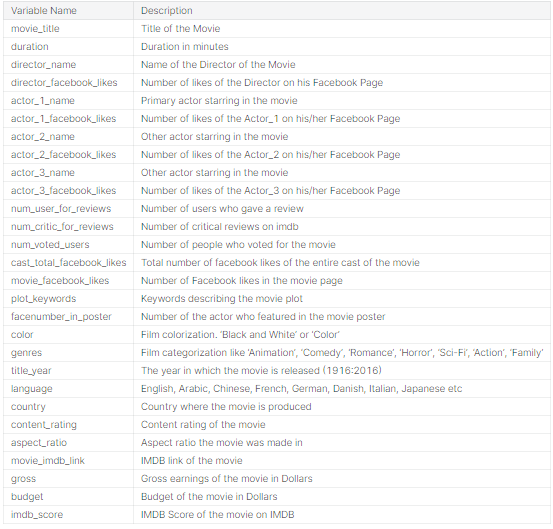
https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset

The data we got Here we have dataset named movie\_metadata in which the target variable is IMDB score and other variables that decide the IMDB score. Instead of just IMDB score,With the help of other parameters we want to predict whether a movie is Hit,Avg or Flop.

## 2.2 Reading and exploring data

First we load data using pandas.read into variable data, then we explore data by function data.head(), data.info()

Here we have dataset named movie\_metadata in which the target variable is IMDB score and other variables that decide the IMDB score. Instead of just IMDB score,With the help of other parameters we want to predict whether a movie is Hit,Avg or Flop.



## 2.3 Categorizing the target varible

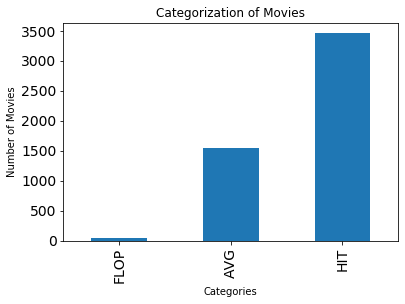
Here we are categorizing the target variable in such a way that IMDB score between 1 and 3 is FLOP , between 3 and 6 is AVG, between 6 and 10 is HIT.And we are using binning in pandas to acheive this by creating a target variable for imdb\_score

*#Categorising the target varible*

bins = [ 1, 3, 6, 10]

labels = ['FLOP', 'AVG', 'HIT']

data['imdb\_binned'] = pd.cut(data['imdb\_score'], bins=bins, labels=labels)



then we got Barplot of imbd\_binned column and the categorical variable

We can see a new column named imdb\_binned but data.head(),which is correctly categorising the imdb score. So will be looking more to this new variable which have a more precise predecessor and the target variable focus changed to the newly created variable imdb\_score.

Our dataset contains 5043 samples(rows) and 28 variables(columns)



## 2.4 Handling the Missing values

Every datset have some sort of missing values, finding out in which cloumns they are and droping the samples that have missing values by drop method and All the categorical columns and the columns with text data are being Label Encodeded in the next step.

Dropping all the samples that having missing values byt data.dropna(inplace=True)

Next we have to find out how the string variables are behaving

*#Describing the categorical data*

data.describe(include**=**'object')

'movie\_title','movie\_imdb\_link' columns are almost unique,so they doesn't contribute in predicting target variable

*#Dropping 2 columns*

data.drop(columns**=**['movie\_title','movie\_imdb\_link'],inplace**=True**)

**CHAPTER 3**

# Visualization

So what exactly the data want to say?. So here our first visualization, Where we infer insights using the dataset and our observation as follows:

Individually inspecting 'cast\_total\_facebook\_likes','num\_critic\_for\_reviews', both are slightly skewed to right

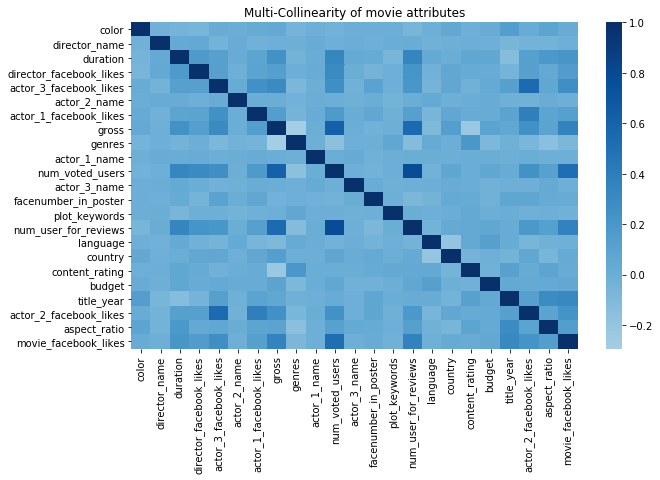
## 3.1 Correlation

To find out whether there is any relation between variables, in other terms multicollineariaty.

fig, ax **=** plt.subplots(figsize**=**(10,6))

sns.heatmap(data.corr(), center**=**0, cmap**=**'Blues')

ax.set\_title('Multi-Collinearity of movie attributes')



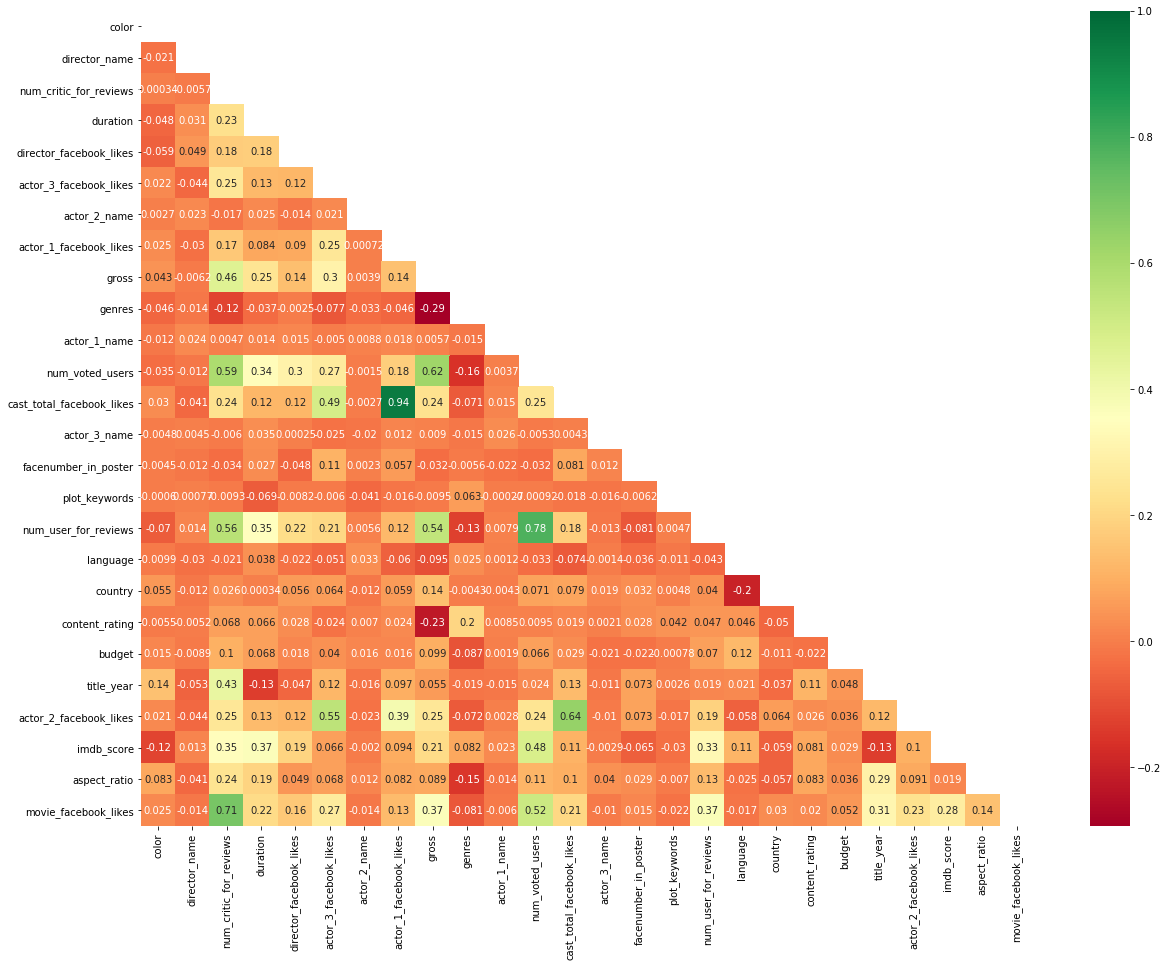
We can see a slightly positive trend between ‘imdb\_score’ and ‘imdb\_binned’ i.e, a score increased, binned terms also varies

Between 0–100 mln budget, we can see a good number of movies providing profit ranging from 0–300 mln dollars (excluding negative profit movies)

The dataset contains the 100 best performing movies from the year 2010 to 2016. However, a heat map tells a different story.

Removing the column "imdb\_score" since we have "imdb\_binned

we will train the model with imdb\_binned not with imdb\_score so dropping the column.



*#Removing few columns due to multicollinearity*

data.drop(columns**=**['cast\_total\_facebook\_likes','num\_critic\_for\_reviews'],inplace**=True**)

*#Removing the column "imdb\_score" since we have "imdb\_binned"*

data.drop(columns**=**['imdb\_score'],inplace**=True**)

data.shape

(3756, 24)

**CHAPTER 4**

# Model building

It demonstrates that the value of y is dependent on the value So, y is referred to as dependent feature or variable x are independent features or variables. Any predictive mathematical model tends to divide the observations (data) into dependent/ independent features in order to determine the causal effect.

## 4.1 Splitting and training the data

we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Splitting the data into X and y where X contains Indepentent variables and y contain Target/Dependent variable.

We need data not only to train our model but also to test our model. So splitting the dataset into 70:30 (Train:Test) ratio.We have a predefined a function in Sklearn library called test\_train\_split, lso using that we can get our train model

**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size **=** 0.3, random\_state **=** 0,stratify **=** y)

print(X\_train.shape)

print(y\_train.shape)

(2629, 23)

(2629,)

## 4.2 Scaling the data

Few variables will be in the range of Millions and some in Tens, lets bring all of them into same scale

*#Scaling the dependent variables*

**from** sklearn.preprocessing **import** StandardScaler

sc **=** StandardScaler()

X\_train **=** sc.fit\_transform(X\_train)

X\_test **=** sc.transform(X\_test)

**CHAPTER 5**

# Feature scaling

## 5.1 Feature Selection using RFECV

Finding optimal features to use for Machine learning model training can sometimes be a difficult task to accomplish.There are just so many methods to choose from and here I am going with RFECV.

Recursive Feature Elimination with Cross Validation

Recursive — involving doing or saying the same thing several times in order to produce a particular result or effect

Feature — individual measurable property or characteristic of a phenomenon being observed — an attribute in your dataset

Cross-Validation — a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting, ie, failing to generalize a pattern.

You will need to declare two variables — X and y where first represents all the features, and the second represents the target variable. Then you’ll make an instance of the Machine learning algorithm (In this case RandomForests). In it, you can optionally pass a random state seed for reproducibility. Now you can create an instance of RFECV.

*#Performing Recursive Feauture Elimation with Cross Validation*

*#Using Random forest for RFE-CV and logloss as scoring*

**from** sklearn.feature\_selection **import** RFECV

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.metrics **import** log\_loss

clf\_rf**=**RandomForestClassifier(random\_state**=**0)

rfecv**=**RFECV(estimator**=**clf\_rf, step**=**1,cv**=**5,scoring**=**'neg\_log\_loss')

rfecv**=**rfecv.fit(X\_train,y\_train)

In [24]:

*#Optimal number of features*

X\_train **=** pd.DataFrame(X\_train)

X\_test **=** pd.DataFrame(X\_test)

print('Optimal number of features :', rfecv.n\_features\_)

print('Best features :', X\_train.columns[rfecv.support\_])

Optimal number of features : 14

Best features : Int64Index([2, 3, 4, 6, 7, 8, 10, 11, 13, 14, 18, 19, 20, 22], dtype='int64')

| **Features Selected** | **Features Dropped** |
| --- | --- |
| duration | color |
| director\_facebook\_likes | director name |
| actor\_3\_facebook\_likes | actor\_2\_name |
| actor\_1\_facebook\_likes | actor\_1\_name |
| gross | facenumber\_in\_poster |
| genres | language |
| num\_voted\_users | country |
| actor\_3\_name | content\_rating |
| actor\_3\_name | aspect\_ratio |

*#Feauture Ranking*

clf\_rf **=** clf\_rf.fit(X\_train,y\_train)

importances **=** clf\_rf.feature\_importances\_

std **=** np.std([tree.feature\_importances\_ **for** tree **in** clf\_rf.estimators\_],

axis**=**0)

indices **=** np.argsort(importances)[::**-**1]

*#Logloss vs Number of features*

**import** matplotlib.pyplot **as** plt

plt.figure()

plt.xlabel("Number of features selected")

plt.ylabel("Cross validation score of number of selected features")

plt.title("Log loss vs Number of fetures")

plt.plot(range(1, len(rfecv.grid\_scores\_) **+** 1), rfecv.grid\_scores\_)

plt.show()

*#Selecting the Important Features*

X\_opt **=** X\_train.iloc[:,X\_train.columns[rfecv.support\_]]

X\_test **=** X\_test.iloc[:,X\_test.columns[rfecv.support\_]]

*#Creating anew dataframe with column names and feature importance*

dset **=** pd.DataFrame()

data1 **=** data

data1.drop(columns**=**['imdb\_binned'],inplace**=True**)

dset['attr'] **=** data1.columns

dset['importance'] **=** clf\_rf.feature\_importances\_

*#Sorting with importance column*

dset **=** dset.sort\_values(by**=**'importance', ascending**=True**)

*#Barplot indicating Feature Importance*

plt.figure(figsize**=**(16, 14))

plt.barh(y**=**dset['attr'], width**=**dset['importance'], color**=**'#1976D2')

plt.title('RFECV - Feature Importances', fontsize**=**20, fontweight**=**'bold', pad**=**20)

plt.xlabel('Importance', fontsize**=**14, labelpad**=**20)

plt.show()

## 5.2 Random Forest

Random forests is an ensemble learning method for classification that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) of the individual trees

n\_estimators is a parameter that specify number of trees in the forest.

criterion is to specify what function to measure the quality of a split. “entropy” is for the information gain.

*#Training the Random Forest Classifer on Train data*

**from** sklearn.ensemble **import** RandomForestClassifier

classifier **=** RandomForestClassifier(n\_estimators **=** 100, criterion **=** 'entropy', random\_state **=** 0)

classifier.fit(X\_opt, y\_train)

​

*#Predicting the target variable*

y\_pred **=** classifier.predict(X\_test)

3.5 Confusion Matrix

Confusion matrix gives a clear view of ground truth and prediction.

*#Confusion Matrix*

**from** sklearn.metrics **import** confusion\_matrix

cm **=** confusion\_matrix(y\_test,y\_pred)

cm

array([[188, 0, 147],

[ 6, 0, 2],

[ 68, 0, 716]], dtype=int64)

**CHAPTER 5**

# Classification Report

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.

The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives. Positive and negative in this case are generic names for the predicted classes

*#Classification Report*

**from** sklearn.metrics **import** classification\_report

cr **=** classification\_report(y\_test,y\_pred)

print(cr)

precision recall f1-score support

AVG 0.72 0.56 0.63 335

FLOP 0.00 0.00 0.00 8

HIT 0.83 0.91 0.87 784

accuracy 0.80 1127

macro avg 0.52 0.49 0.50 1127

weighted avg 0.79 0.80 0.79 1127

**CHAPTER 6**

# Conclusion

As stated in data modeling, the data we selected was a well labeled one so we choose supervised learning model like random forest and feature scaling. Upon

Testing the hypothesis values, we got an accuracy about 0.8 for our predicting model. Further classification we got an absolute model which give an acute values for predicting movie success, which is hit, avg, flop based on the imdb dataset.

## Future Work

we are planning to predict the success of movies which is planning to be released in major OTTs like Netflix, amazon prime, hulu; the main focus is to get the feedback of various aspects of the movie like quality, story line, sound, graphics, acting and many more from the subscribed users or a team of officials who will rate the movies based on their taste and perscpective. If the success rate is above 80 we can release the film or if it below the preferred value we will reject the propose of the film.