PROJECT REPORT

"IMDb Movie Rating Prediction"

SUBMITTED BY

Abhishek Kailasam	**(100856936)**
Sanath Davis	**(100884693)**
Bindia Biju	**(100886575)**
Riya Xavier	**(100847513)**

UNDER THE GUIDANCE OF:

Sk. Md. Mizanur Rahman, PhD

Associate Professor in
Artificial Intelligence Analysis, Design and Implementation
Durham College

Table of Content

INTRODUCTION

What characteristics distinguish a great film? The first things that come to mind are an engaging story, standout acting from the cast, a fitting and memorable soundtrack, and excellent graphics.
A more informed spectator will also be interested in the film's direction and photography, and some films must meet requirements set forth by the genre they belong to (for example, horror films must be tense and terrifying). Movies are frequently evaluated based on the message they hope to deliver or the applicability of their themes. Even though this is a simplified perspective, it is clear that many of the factors that influence whether or not a movie is good are arbitrary or difficult to measure.

A key source of international trade and marketing nowadays, movies are no longer the only means of entertainment. In particular, among young people, movies start a new enthusiasm. The general public is likewise interested in the success of movies, in addition to movie directors and box office officials. These topics used to be discussed on social media. Consequently, social media data analysis about movies has recently gained popularity among data analysts.

In addition to this, there are still some other opportunities, such as researching an actor's or director's past successes. Once more, the analysis may vary depending on the country. Naturally, people from different parts of the world have different reactions.

Today, you may watch movies online. People can publish their movie reviews on websites like IMDb (Internet Movie Database), Rotten Tomatoes, Metacritics, etc. These sites are growing in popularity day by day since they provide individuals with frank reviews. Therefore, there is a wealth of information on movie reviews and ratings online. This article analyzes movie rating data in order to forecast movie ratings.

As people are very enthusiastic about movies, there are many studies that attempt to forecast the success rate of movies. Very few studies use a movie's attributes to forecast its success rate, including the director, screenplay, actor or actress, genre, etc. Therefore, the goal of this article is to forecast the success rate using the movie's own features.

Dataset for this project has been obtained from Kaggle https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows (Shankhdhar, --)

EXPLORATORY DATA ANALYSIS (EDA)

Exploratory data analysis is the crucial process of doing preliminary analyses on data in order to find patterns, identify anomalies, test hypotheses, and double-check assumptions with the aid of summary statistics and graphical representations.

The IMDb dataset contains 1000 rows and 16 columns

	Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1
0	https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	Drama	9.3	Two imprisoned men bond over a number of years...	80.0	Frank Darabont	Tim Robbins
1	https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	Crime, Drama	9.2	An organized crime dynasty's aging patriarch t...	100.0	Francis Ford Coppola	Marlon Brando
2	https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action, Crime, Drama	9.0	When the menace known as the Joker wreaks havo...	84.0	Christopher Nolan	Christian Bale

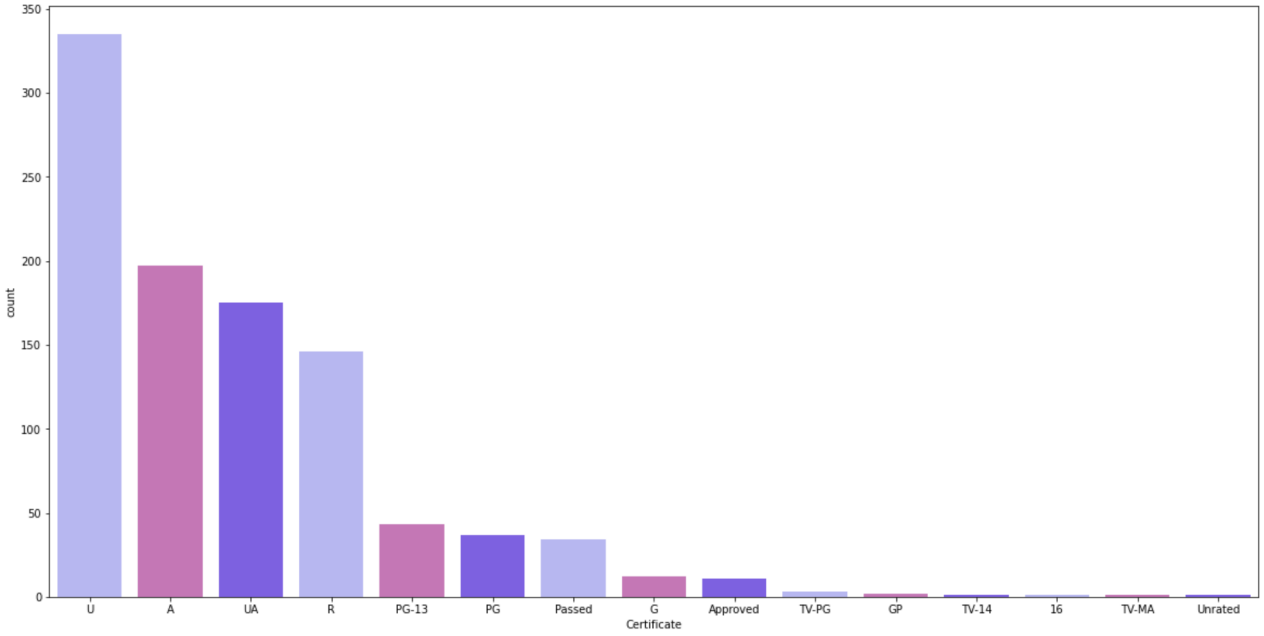
Let’s understand each column of the dataset.

Top 1000 Movies by IMDB Rating data

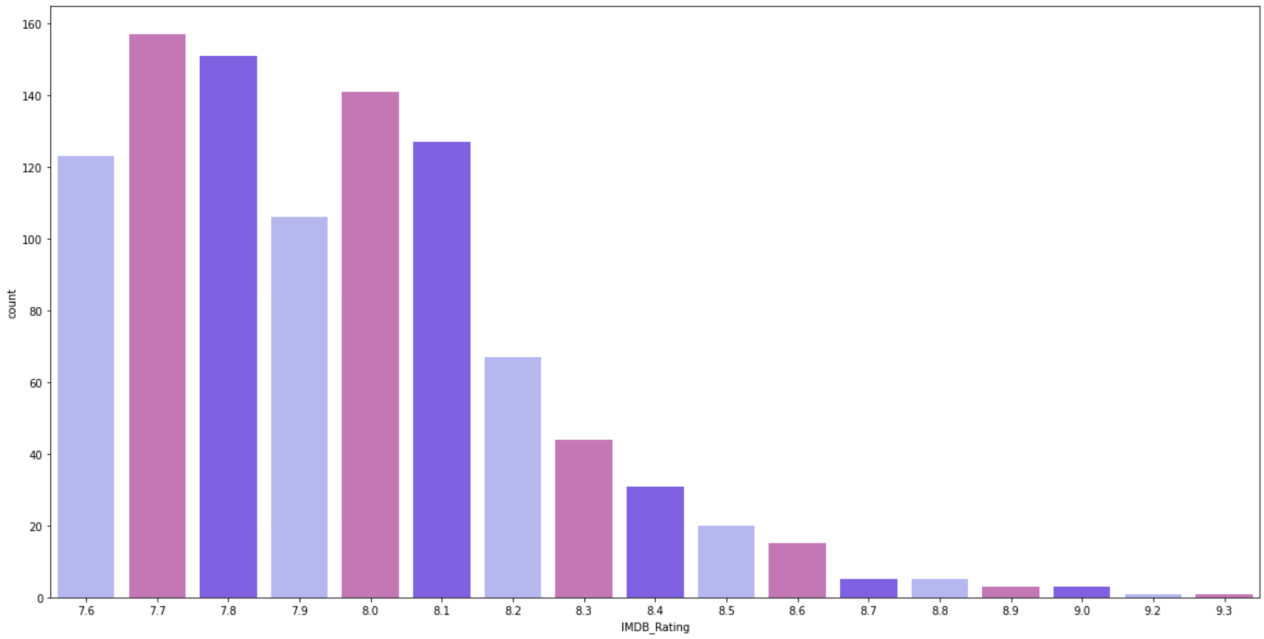
```
PosterLink: Link of the poster that imdb using
SeriesTitle: Name of the movie
ReleasedYear: Year at which that movie released
Certificate: Certificate earned by that movie
Runtime: Total runtime of the movie
Genre: Genre of the movie
IMDB Rating: Rating of the movie at IMDB site
Overview: mini story/ summary
Meta_score: Score earned by the movie
Director: Name of the Director
Star1,Star2,Star3,Star4: Name of the Stars
No of votes: Total number of votes
Gross: Money earned by that movie
```

Let’s check the spread of data in our dataset.

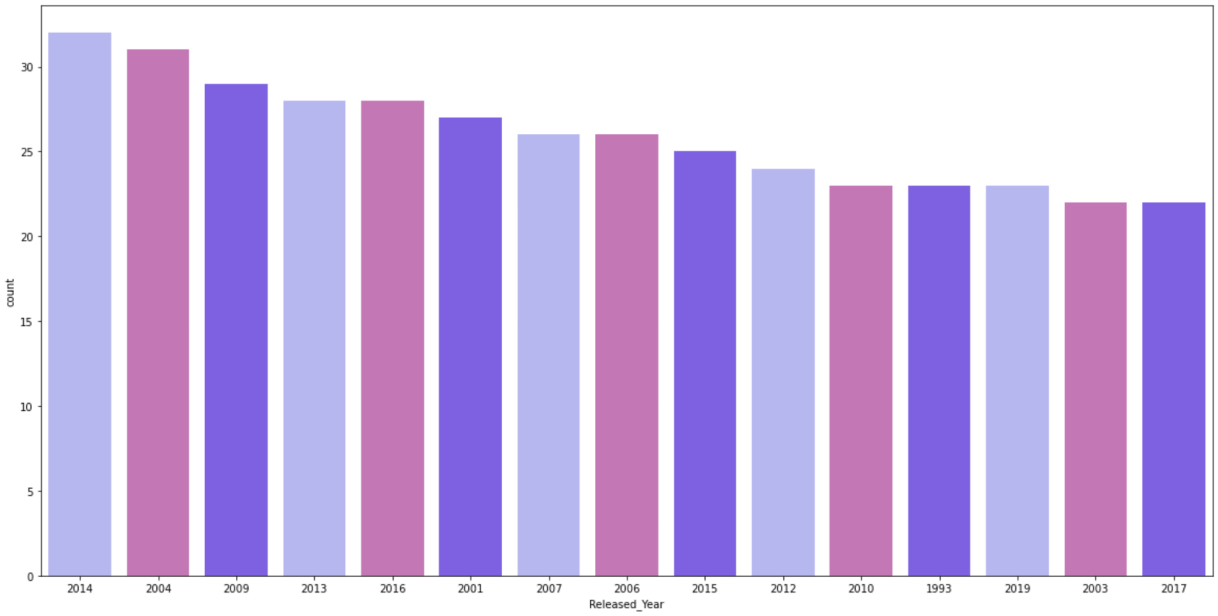
Number of movies certificate-wise



Number of movies IMDb rating wise

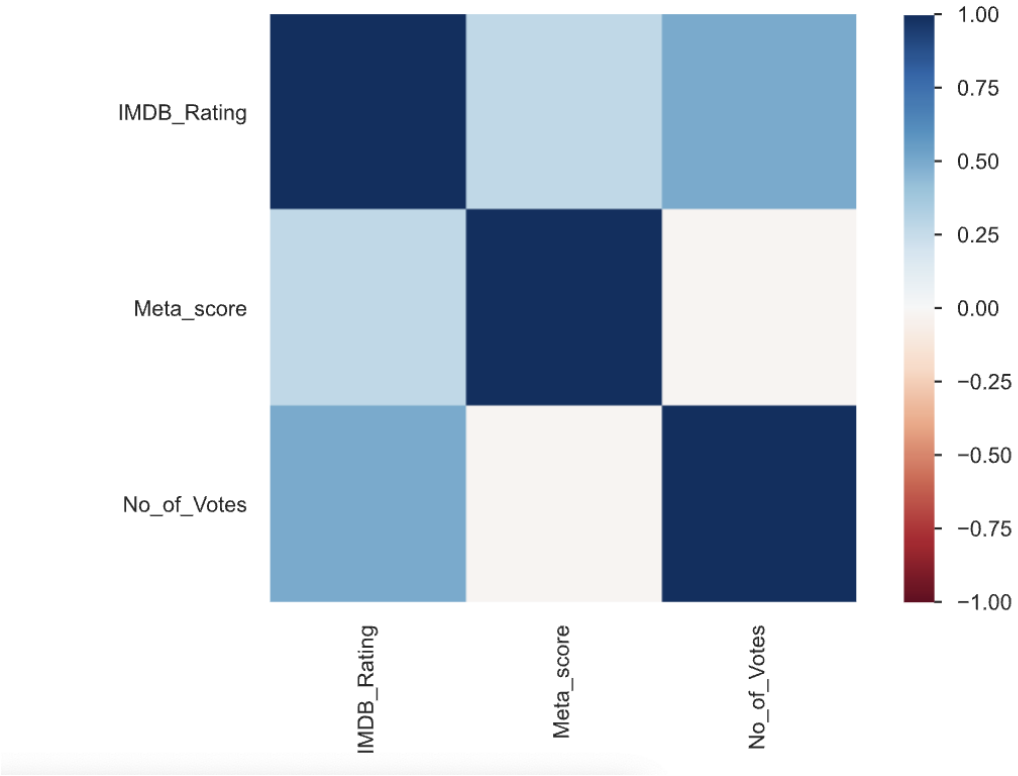


Number of movies year wise

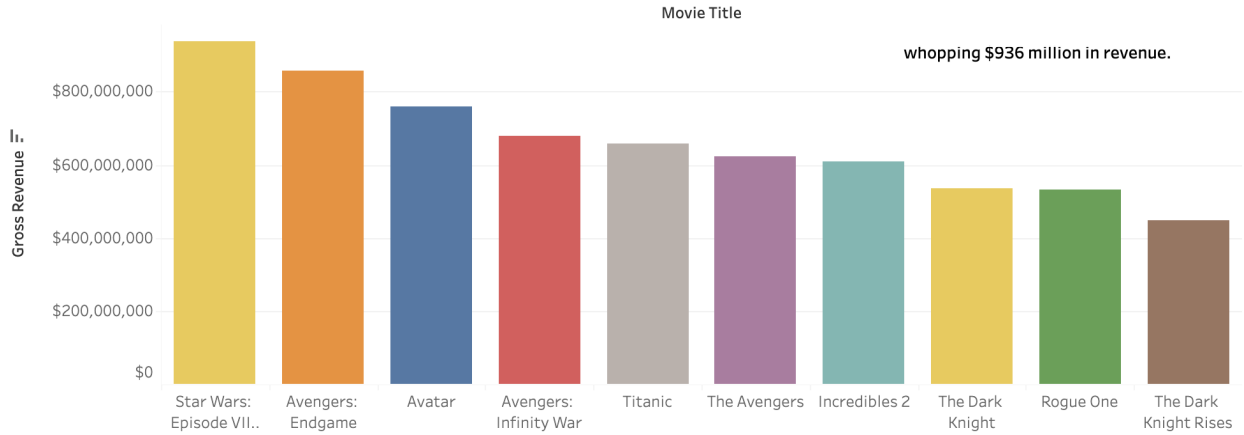


Pearson's Correlation Map

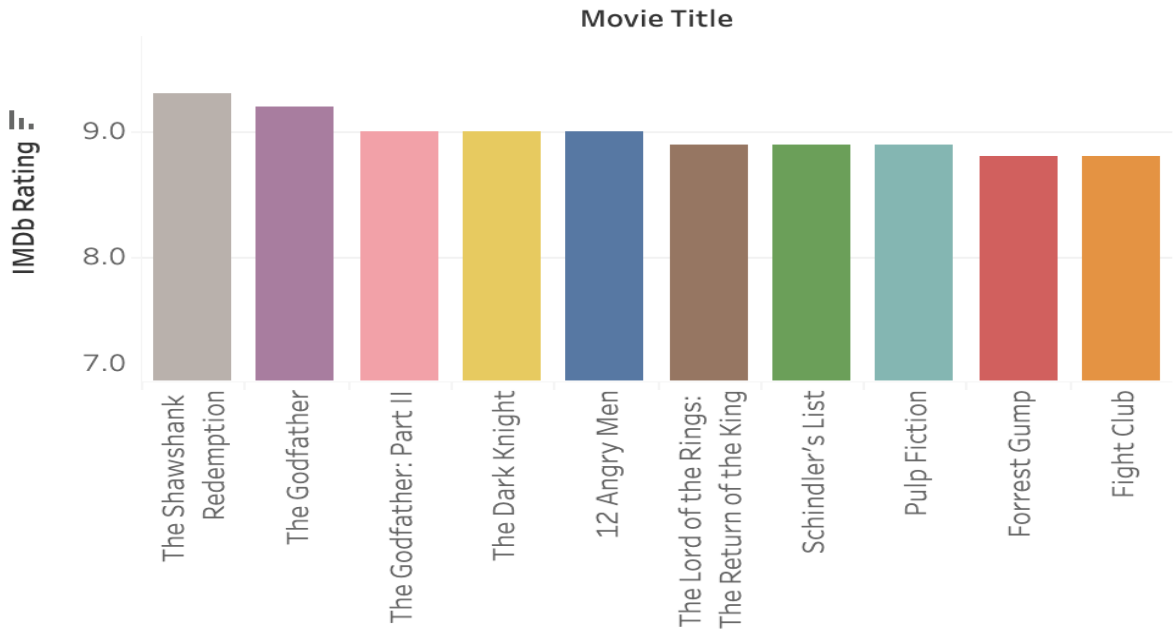
Pearson's correlation coefficient (*r*) is a measure of linear correlation between two variables. Its value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation.



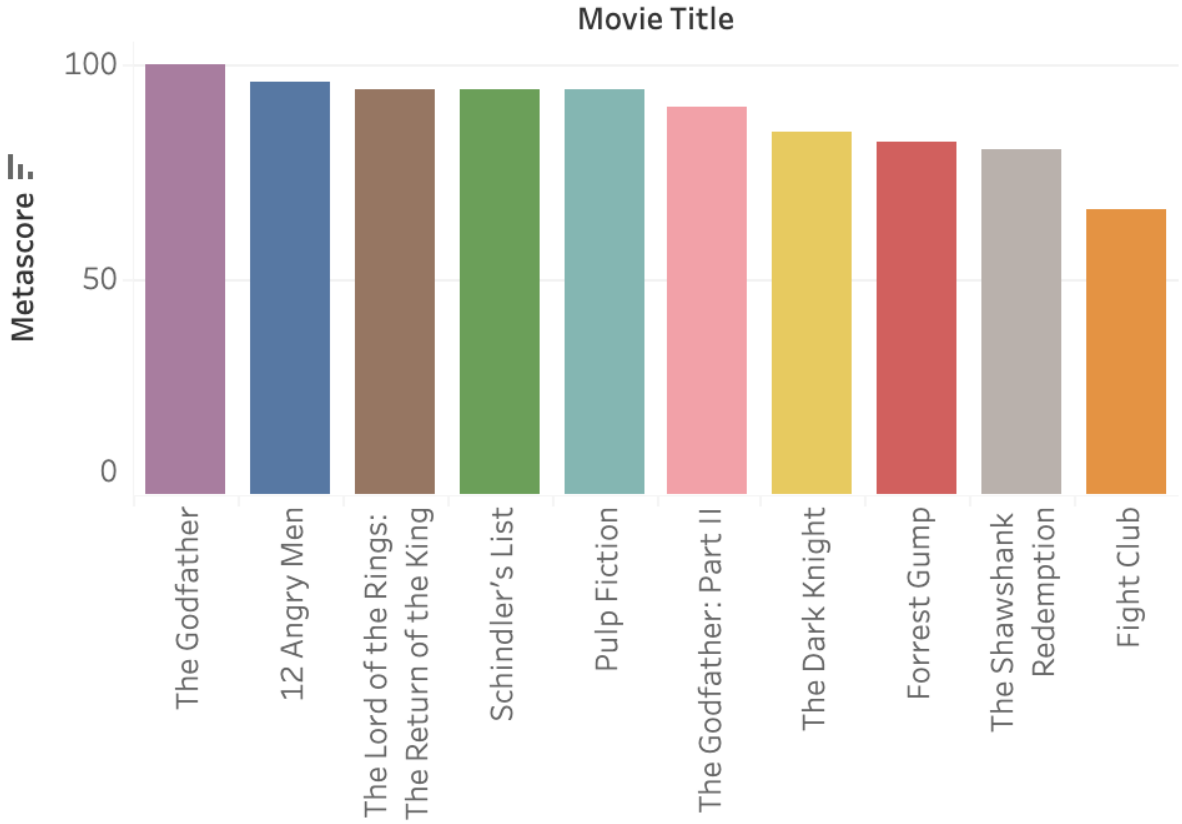
Top 10 Highest-Grossing Movies



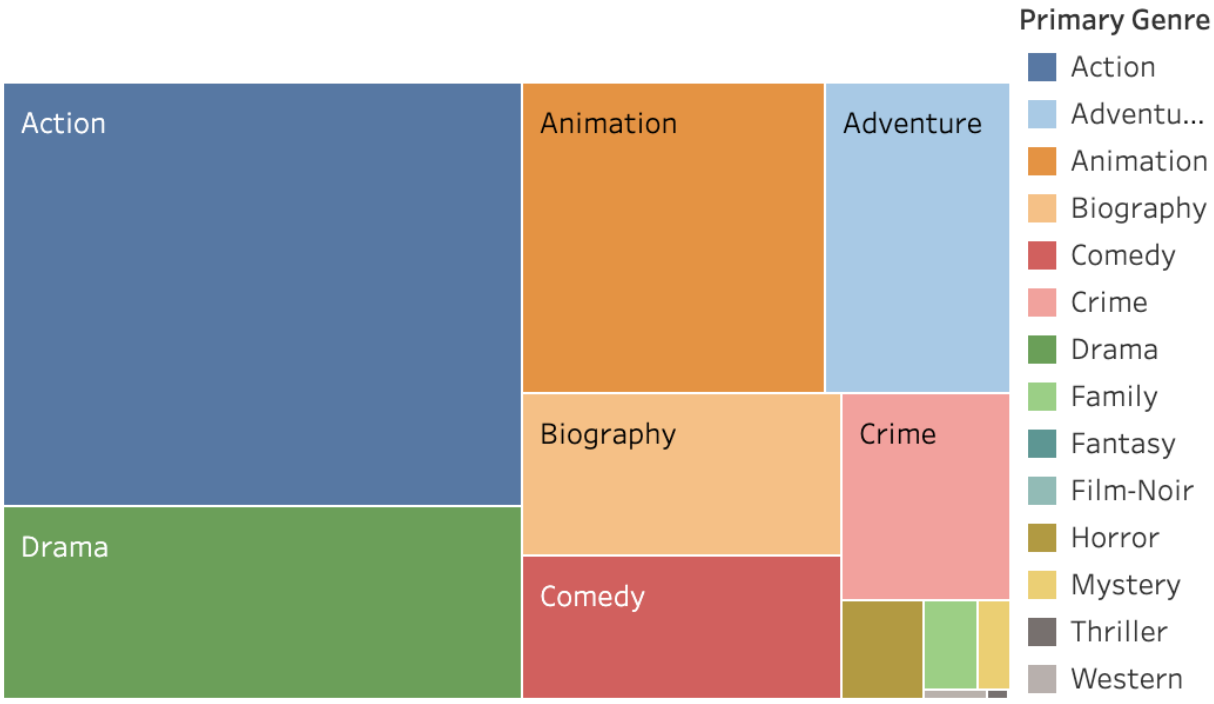
Top 10 Movies by IMDb Ratings



Top 10 Movies by MetaScore



Movie Revenue by Genre



MODELLING APPROACH

We have decided to build below-mentioned models for our project:

1. Linear Regression
2. Random Forest Regressor
3. XGBoost Regressor

Out of all these, our main focus is on Random Forest with XGBoost Ensemble technique. We will compare the error rates with all of the above models so that we can go with the best performing model.

Why Linear Regression?

Linear regression is a statistical modelling process that compares the relationship between two variables, which are usually independent or explanatory variables and dependent variables. For each data point, this linear regression indicator depicts the value of the trendline.

Working of Linear Regression:

There are two types of variable in Linear Regression, one variable is called an independent variable, and the other is a dependent variable. Linear regression is commonly used for predictive analysis. The main idea of regression is to examine two things. First, does a set of predictor variables do a good job in predicting an outcome (dependent) variable? The second thing is which variables are significant predictors of the outcome variable?

Why Random Forest?

- No overfitting - The use of multiple trees reduces the risk of overfitting
- Training time is less.
- High accuracy - For large databases produces highly accurate predictions

Random Forest falls under the category of the supervised algorithm. It is also an ensemble technique.

Working of Random Forest

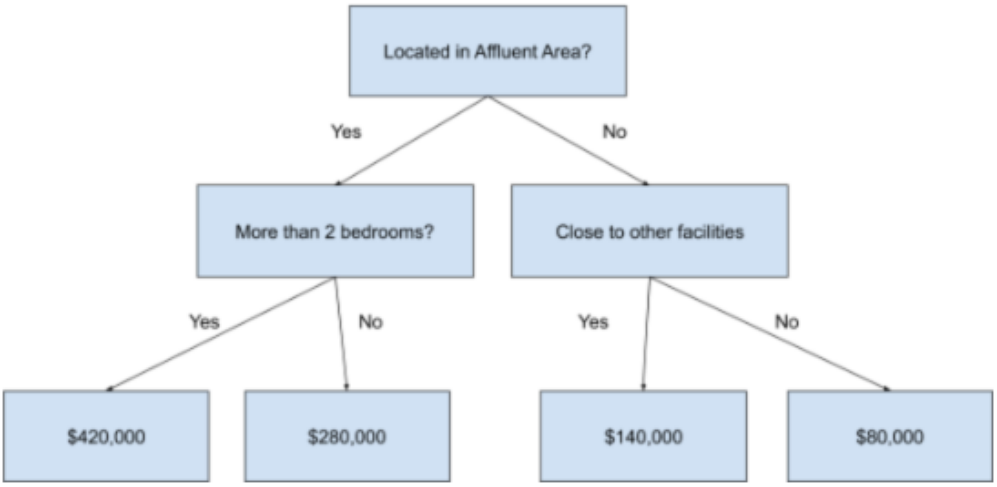
Firstly, ensemble simply means combining multiple models. Ensemble uses two types of methods:

1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting or averaging. Example, Random Forest.
2. **Boosting**– It combines weak learners into strong learners by creating sequential models. For a classification task, the final model would have the highest accuracy. For example, ADA BOOST, XGBOOST

Bagging, also known as Bootstrap Aggregation, is the ensemble technique used by random forests. Bagging chooses a random sample from the data set. Each model is generated from the samples provided by the original data with a replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting or averaging. This step which involves combining all the results and generating is known as aggregation.

Let's understand the working with an example.Consider that you want to purchase real estate and want to determine what constitutes a fair deal in order to avoid being taken advantage of.

The natural course of action would be to review previous sales prices for nearby homes, then develop some sort of decision criteria to compile the typical selling prices given the real estate specification. The decision chart may be used to determine whether or not the apartment's quoted price is a good deal. It could seem as follows:



With a sequence of yes/no questions leading you from the real estate description ("3 bedrooms") to its historical average price, the chart simulates a decision-making process. The decision tree may be used to estimate a real estate's predicted price based on its features.However, you could come up with a distinctly different decision tree structure:

The "wisdom of the crowds" is used by the random forest regression technique. It uses a number of distinct regression decision trees and gives each a "vote." Based on the choice criteria it selected, each tree must forecast the anticipated real estate price. The average of all the forecasts is then calculated via random forest regression to get an excellent estimation of what the anticipated price for real estate should be.
(*The Ultimate Guide to Random Forest Regression*, 2020)

Boosting

In machine learning, boosting is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning

Why XG Boost Regressor?

XGBoost is a powerful approach for building supervised regression models.The two main reasons to use XGBoost are execution speed and model performance. XGBoost has been integrated with a wide variety of other tools and packages such as scikit-learn for Python .

Working of XG Boost Regressor:

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg:linear. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the

predictions are combined, bad predictions cancels out and better one sums up to form final good predictions.

Data Preparation

Firstly, we checked to see if there are any unwanted columns in our dataset. We found some columns that are not important for our analysis, hence dropped them.

```
#axis = 1 means we are working with columns
dataset = dataset.drop([
    'Star1',
    'Star2',
    'Star3',
    'Star4',
    'Poster_Link',
    'Overview',
    'Series_Title',
    'Series_Title',
    'Director'
    ],
axis = 1)
```

Next we found that there were 169 records with NULL values. These were the values for revenue of movies. We went ahead and dropped those rows of data.

Then we found an unrelated entry in Year column.

```
dataset['Released_Year'].unique()
```

```
array(['1994', '1972', '2008', '1974', '1957', '2003', '1993', '2010',
       '1999', '2001', '1966', '2002', '1990', '1980', '1975', '2019',
       '2014', '1998', '1997', '1995', '1991', '1977', '1954', '2011',
       '2006', '2000', '1988', '1985', '1968', '1960', '1942', '1936',
       '1931', '2018', '2016', '2017', '2012', '2009', '1981', '1979',
       '1964', '2004', '1992', '1987', '1986', '1984', '1983', '1976',
       '1973', '1971', '1965', '1962', '1959', '1958', '1952', '1944',
       '1941', '2013', '2007', '2005', '1989', '1963', '1950', '1948',
       '2015', '1996', '1982', '1978', '1967', '1951', '1949', '1940',
       '1939', '1934', '1970', '1969', '1961', '1946', '1930', '1938',
       '1933', 'PG', '1953'], dtype=object)
```

Remove the phrase 'PG' which is not an year

Hence dropped those specific records.

Encoding

While checking Genre and Certificate columns for data consistency we found that a single movie can have multiple genres, but this will be difficult to work on. Hence, to convert these categorical data into numerical values we use encoding techniques on the Genre and Certificate column. For this we used One Hot Encoding technique.

In [20]:
```
# let us see what the encoding of the category Drama looks like
enc.transform([['Drama']]).toarray()
```

```
/Users/riya/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X does not have valid feature names, but OneHotEncoder was fitted with feature names
  warnings.warn(
```

Out[20]:
```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

One Hot Encoding was not successful due to the presence of multiple values for Genre. Hence we encoded manually.

First Genre. Find out all unique Genres - SInce many movies have multiple genres - and make them boolean columns

In [26]:
```
genres = dataset.Genre.unique()
required_genre_columns = []
for genre in genres:
    multiple_genres = genre.split(", ")
    for single_genre in multiple_genres:
        required_genre_columns.append(single_genre)


#removing duplicates
required_genre_columns = list(dict.fromkeys(required_genre_columns))
print(required_genre_columns)
```

```
['Drama', 'Crime', 'Action', 'Adventure', 'Biography', 'History', 'Sci-Fi', 'Romance', 'Western', 'Fantasy', 'Comed
y', 'Thriller', 'Animation', 'Family', 'War', 'Mystery', 'Music', 'Horror', 'Sport', 'Musical', 'Film-Noir']
```

Add each unique Genre as column to the dataset

In [27]:
```
def transform_row(r):
    for new_genre in required_genre_columns:
        r[new_genre] = 1 if new_genre in r.Genre else 0
    return r

dataset = dataset.apply(transform_row, axis=1)
dataset = dataset.drop(['Genre'], axis = 1)
dataset
```

Out[27]:

	Released_Year	Certificate	Runtime	IMDB_Rating	Meta_score	No_of_Votes	Gross	Drama	Crime	Action	...	Thriller	Animation	Family	War	Myste
0	1994	A	142	9.3	80.0	2343110	28341469	1	0	0	...	0	0	0	0	

Out[27]:

	Released_Year	Certificate	Runtime	IMDB_Rating	Meta_score	No_of_Votes	Gross	Drama	Crime	Action	...	Thriller	Animation	Family	War	Myst
0	1994	A	142	9.3	80.0	2343110	28341469	1	0	0	...	0	0	0	0	
1	1972	A	175	9.2	100.0	1620367	134966411	1	1	0	...	0	0	0	0	
2	2008	UA	152	9.0	84.0	2303232	534858444	1	1	1	...	0	0	0	0	
3	1974	A	202	9.0	90.0	1129952	57300000	1	1	0	...	0	0	0	0	
4	1957	U	96	9.0	96.0	689845	4360000	1	1	0	...	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
990	1971	PG	157	7.6	77.0	30144	696690	1	0	0	...	0	0	0	1	
991	1970	GP	144	7.6	50.0	45338	1378435	0	0	0	...	0	0	0	1	
992	1967	U	78	7.6	65.0	166409	141843612	0	0	0	...	0	1	1	0	
994	1964	U	87	7.6	96.0	40351	13780024	0	0	0	...	0	0	0	0	
997	1953	Passed	118	7.6	85.0	43374	30500000	1	0	0	...	0	0	0	1	

713 rows × 28 columns

In [22]:
```
#Encoding all the categories
encoded_category = enc.transform(dataset[['Genre']]).toarray()
encoded_category = pd.DataFrame(encoded_category, columns=enc.categories_)
print(encoded_category.shape)
encoded_category
```

```
(713, 172)
```

Out[22]:

	Action, Adventure	Action, Adventure, Comedy	Action, Adventure, Drama	Action, Adventure, Family	Action, Adventure, Fantasy	Action, Adventure, History	Action, Adventure, Horror	Action, Adventure, Mystery	Action, Adventure, Romance	Action, Adventure, Sci-Fi	...	Film-Noir, Mystery, Thriller	Horror	Horror, Mystery, Sci-Fi	Horr Myst Thrill
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
708	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
709	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
710	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
711	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
712	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	

713 rows × 172 columns

Model1 - Linear Regression

Now, for Linear Regression, we split the dataset as 60:40 for train and test and apply the model.

Check Error

In [32]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error
linear_reg_mean_square = mean_squared_error(y_test, y_pred)
linear_reg_mean_absolute_percentage = mean_absolute_percentage_error(y_test, y_pred)

print("mean Square: ", linear_reg_mean_square)
print("mean absolute percentage error: ", linear_reg_mean_absolute_percentage)

mse = mean_squared_error(y_test, y_pred)
print("RMSE: %.2f" % (mse**(1/2.0)))
```

```
mean Square:  0.0425164107552973
mean absolute percentage error:  0.020226780645811317
RMSE: 0.21
```

Model2 - Random Forest Regressor

2) Random Forest Regressor

In [33]:
```
from sklearn.ensemble import RandomForestRegressor
regr = RandomForestRegressor(max_depth=2, random_state=0)
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

random_forest_reg_mean_square = mean_squared_error(y_test, y_pred, squared=False)
random_forest_reg_mean_absolute_percentage = mean_absolute_percentage_error(y_test, y_pred)

print("mean Square: ", random_forest_reg_mean_square)
print("mean absolute percentage error: ", random_forest_reg_mean_absolute_percentage)

mse = mean_squared_error(y_test, y_pred)
print("RMSE: %.2f" % (mse**(1/2.0)))
```

```
mean Square:  0.22038027617212402
mean absolute percentage error:  0.0220395
RMSE: 0.22
```

click to scroll output; double click to hide

Later we included XGBoost ensemble technique to make our prediction better.

Model3 - XGBoost Regressor

3) XG Boost Regressor

In [34]:

```
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

model = XGBRegressor()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

score = model.score(X_train, y_train)
print('Training Score:', score)

score = model.score(X_test, y_test)
print('Testing Score:', score)

output = pd.DataFrame({'Predicted':y_pred})

xg_reg_mean_square = mean_squared_error(y_test, y_pred, squared=False)
xg_reg_mean_absolute_percentage = mean_absolute_percentage_error(y_test, y_pred)

print("mean Square: ", xg_reg_mean_square)
print("mean absolute percentage error: ", xg_reg_mean_absolute_percentage)

mse = mean_squared_error(y_test, y_pred)
print("RMSE: %.2f" % (mse**(1/2.0)))
```

```
Training Score: 0.9999118275837339
Testing Score: 0.5844648960746007
mean Square:  0.1988609735299707
mean absolute percentage error:  0.019651313486915643
RMSE: 0.20
```

Fine Tuning Parameters

Hyperparameter Tuning with Grid Search and Random Search

Hyperparameter Grid Search with XGBoost

In [35]:

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb

params = { 'max_depth': [3,6,10],
           'learning_rate': [0.01, 0.05, 0.1],
           'n_estimators': [100, 500, 1000],
           'colsample_bytree': [0.3, 0.7]}
xgbr = xgb.XGBRegressor(seed = 20)
clf = GridSearchCV(estimator=xgbr,
                   param_grid=params,
                   scoring='neg_mean_squared_error',
                   verbose=1)
clf.fit(X, y)
print("Best parameters:", clf.best_params_)
print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
Best parameters: {'colsample_bytree': 0.7, 'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 500}
Lowest RMSE:  0.2684548496780129
```

Hyperparameter RANDOM Search with XGBoost

In [37]:
```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
params = { 'max_depth': [3, 5, 6, 10, 15, 20],
           'learning_rate': [0.01, 0.1, 0.2, 0.3],
           'subsample': np.arange(0.5, 1.0, 0.1),
           'colsample_bytree': np.arange(0.4, 1.0, 0.1),
           'colsample_bylevel': np.arange(0.4, 1.0, 0.1),
           'n_estimators': [100, 500, 1000]}
xgbr = xgb.XGBRegressor(seed = 20)
clf = RandomizedSearchCV(estimator=xgbr,
                         param_distributions=params,
                         scoring='neg_mean_squared_error',
                         n_iter=25,
                         verbose=1)
clf.fit(X, y)
print("Best parameters:", clf.best_params_)
print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
Best parameters: {'subsample': 0.8999999999999999, 'n_estimators': 500, 'max_depth': 3, 'learning_rate': 0.1, 'colsam
ple_bytree': 0.7999999999999999, 'colsample_bylevel': 0.5}
Lowest RMSE:  0.2627337288861362
```

Comparing Grid Search and Random Search

Comparing GRID SEARCH and Randomised Search on xgboost

Search Type	Time Taken	Lowest RMSE
Grid Search	1 minute 3 seconds	0.27
Randomised Search	24 seconds	0.27

CONCLUSION

There are three error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

- Mean Squared Error (MSE).
- Root Mean Squared Error (RMSE).
- Mean Absolute Error (MAE)

After running our models, we checked the error rates for all of our models.

Comparing Models

Regressor	Time Taken	Mean Square Error	Mean Absolute Percentage Error	RMSE
Linear	.4 seconds	0.0425	0.02	0.21
Random Forest	.2 seconds	0.220	0.0220	0.22
XG Boost	1.2 seconds	0.198	0.0196	0.20
XG Boost after Tuning	.6 seconds	0.191	0.018	0.19

Here we see that error rates are least for XGBoost with Hyperparameter Tuning. Hence, we can use this model for our prediction system.

REFERENCES

Shankhdhar, H. *IMDB Movies Dataset*. Kaggle.
Retrieved December 7,2022, from
https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movie-
and-tv-shows

The Ultimate Guide to Random Forest Regression. (2020, September 17).
Keboola.
Retrieved December 7, 2022, from
https://www.keboola.com/blog/random-forest-regression

Design of a Movie Review Rating Prediction (MR2P) Algorithm. (2020, August).
Oluwatofunmi Adetunji, Babcock University, Ilishan-Remo
https://www.researchgate.net/publication/343686586_Design_of_a_Movie_Review_R
ating_Prediction_MR2P_Algorithm

Datacamp. (2019, Dec 05). *Understanding Logistic Regression in Python Tutorial*.
Understanding Logistic Regression in Python Tutorial.
https://www.datacamp.com/tutorial/understanding-logistic-regression-python