Predicting IMDb Scores Using Machine Learning

Phase 4 Submission Document

Team Members

ABIN BINU JACOB – 961721104002

SELVAK S - 961721104011

ARUN D - 961721104304

VASANTH P.S - 961721104317

SANTHOSH KUMAR R – 961721104314

Project: Predicting IMDb Scores

Phase 4 : Development part 2

Topic : Continue building IMDb Score prediction model by Feature engineering , Model training and Evaluation

Introduction

- Predicting IMDb scores is valuable for movie studios, distributors, and viewers. It assists in understanding a movie's potential success, guiding marketing strategies, and aiding investment decisions.
- This report details the process of building an IMDb score prediction model. To compare accuracy among various types of regression analysis, with and without clustering techniques.
- Specifically, for regression analysis, we used three different types of regression to create prediction models, which are linear regression, polynomial regression, and support vector regression.
- As clustering into smaller groups of similar items may improve accuracy of the prediction models, we cluster the movie data and apply regression analysis on each cluster.
- Three clustering techniques are explored, namely clustering by movie genre, K-mean clustering and Expectation Maximization clustering. In order to compare performance across models, k-fold cross validation technique was employed to divide the data in two groups: training and testing data.
- The training data was used to create the prediction model, while the testing data was used to test accuracy of the regression models. R-square and root mean square error (RMSE) were used as performance indicators of the models.

Content for Project

Data Overview

Data Pre-processing

Feature Engineering

Model Selection

Model Training

Model Evaluation

Case Study

The dataset here gives the massive information about the movies and their IMDB scores respectively. We are going to analyze each and every factor which can influence the IMDb ratings so that we can predict better results. The movie with the higher IMDb score is more successful as compared to the movies with low IMDb score.

Data Source

A good data source for house price prediction using machine learning should beAccurate, Complete, Covering the geographic area of interest, Accessible.

Data Overview

We collected a comprehensive dataset of movies, including various features such as genres, cast, director, budget, and release year. The dataset also includes IMDb scores, which serve as our target variable

Feature Engineering

Feature engineering involved extracting valuable features from the dataset, including

- Genre-based features
- Actor and director-related features
- Budget normalization
- Release year transformation

- It describes how the authors extracted additional features from the movie data to use in their prediction models

- The features include one hot encoding of categorical data, mean rating of directors, writers, cast and production companies, and release month of the movie

- This also shows how the movie ratings correlate with the extracted features using plots and a correlation matrix.

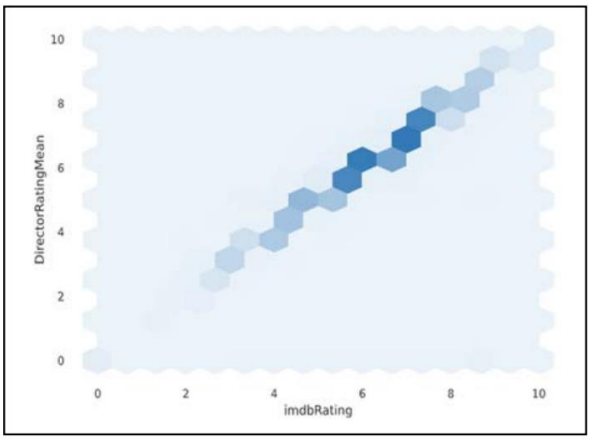


Fig. 1. Interaction of Director Mean Rating with imdbRating.

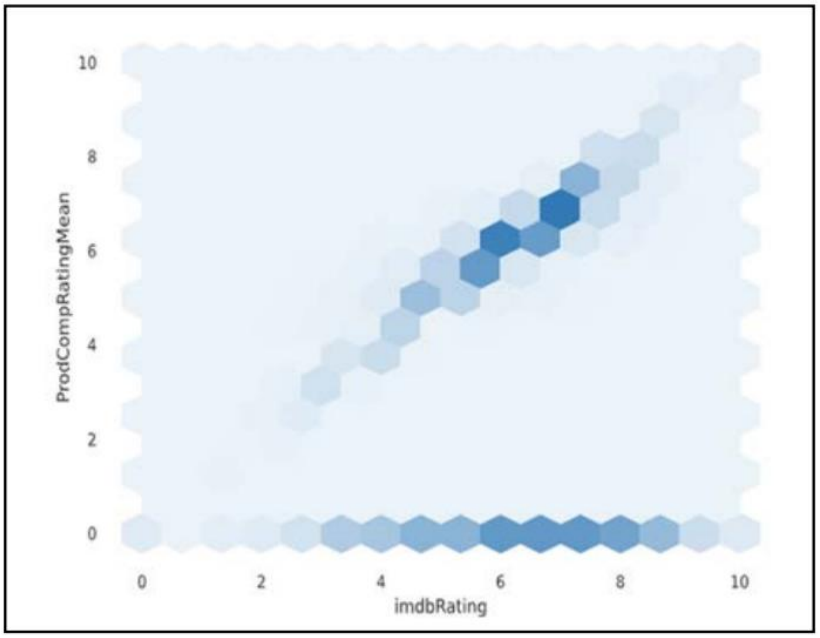


Fig. 2. Interaction of Production Company Ratings with imdb Ratings.

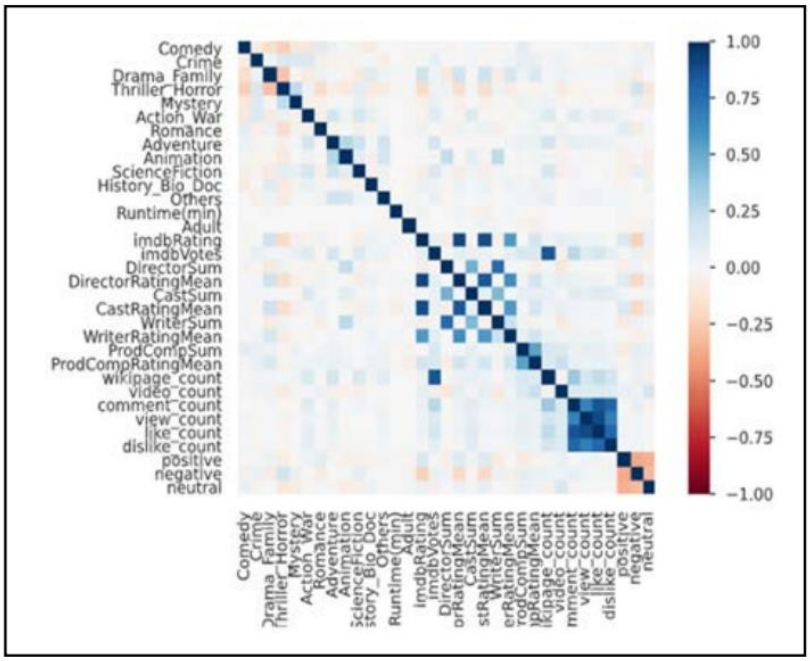
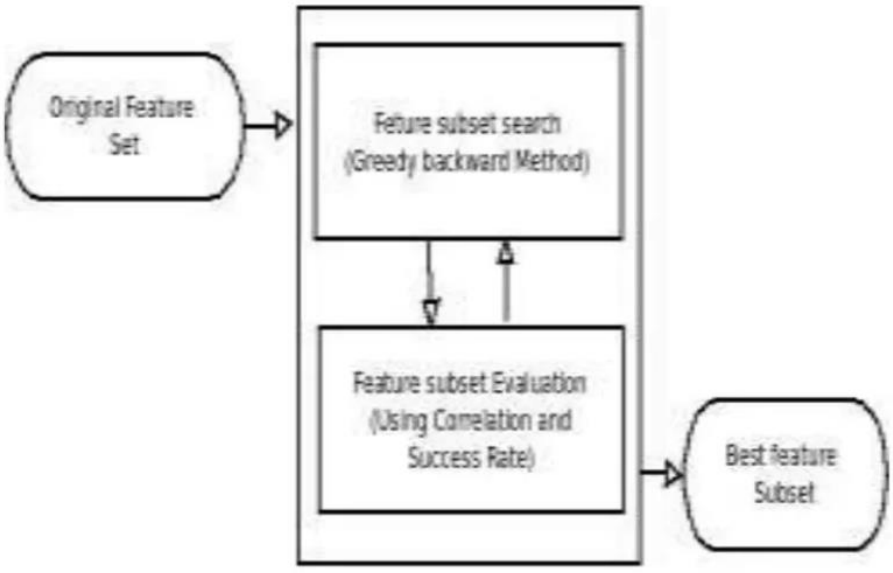


Fig. 3. Correlation Matrix.



Program

Removing Columns with names

In[41]:

```
movie_df.drop('director_name', axis=1, inplace=True)
```

In[42]:

```
movie_df.drop('actor_1_name',axis=1,inplace=True)
```

In[43]:

```
movie_df.drop('actor_2_name',axis=1,inplace=True)
```

In[44]:

```
movie_df.drop('actor_3_name',axis=1,inplace=True)
```

In[45]:

```
movie_df.drop('movie_title',axis=1,inplace=True)
```

In[46]:

```
movie_df.drop('plot_keywords',axis=1,inplace=True)
```

In[47]:

```
movie_df['genres'].value_counts()
```

Out[47]:

```
Drama                                209
Comedy                               186
Comedy|Drama|Romance                 182
Comedy|Drama                         180
Comedy|Romance                       149
                                    ...
Comedy|Drama|Horror                    1
Mystery|Western                        1
Animation|Comedy|Drama|Romance         1
Adventure|Drama|Romance|Western        1
Drama|War|Western                      1
Name: genres, Length: 875, dtype: int64
```

In[48]:

```
movie_df.drop('genres',axis=1,inplace =True)
```

Remove the Linear dependent variables

In[49]:

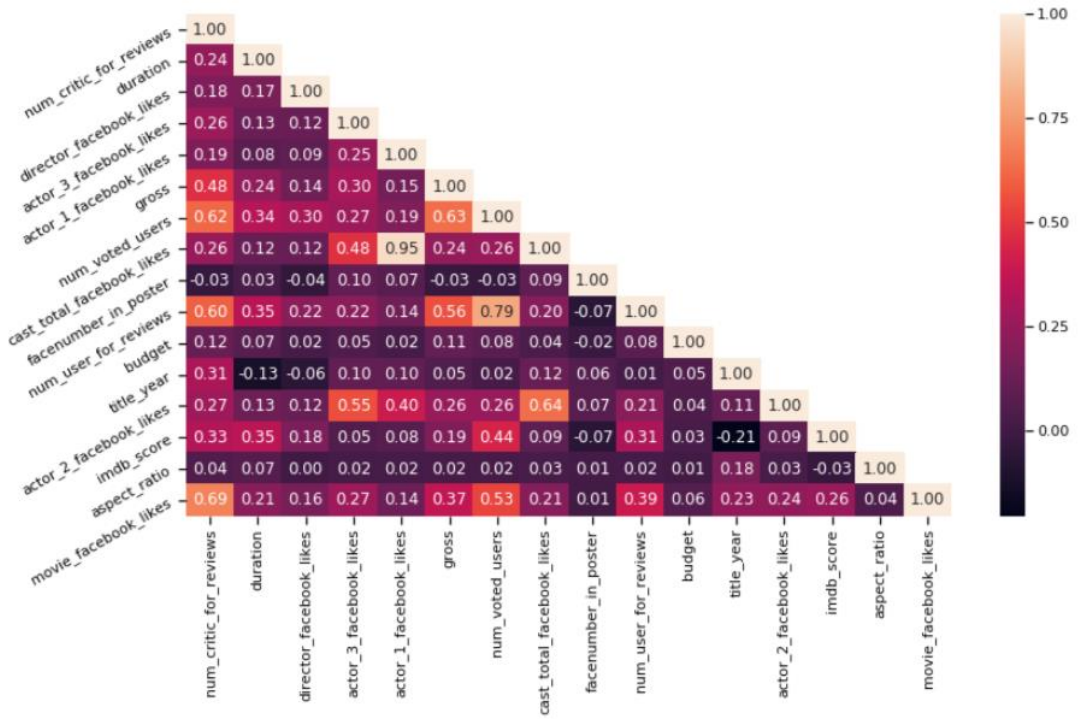
```
movie_df.drop('Profit',axis=1,inplace=True)
```

In[50]:

```
movie_df.drop('Profit_Percentage',axis=1,inplace=True)
```

In[51]:

```
import matplotlib.pyplot as plt
import seaborn as sns
corr = movie_df.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth": 2.5})
plt.figure(figsize=(13,7))
# create a mask so we only see the correlation values once
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
a = sns.heatmap(corr,mask=mask, annot=True, fmt='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=30)
```



In[52]:

```
movie_df['Other_actor_facebbok_likes']=movie_df["actor_2_facebook_likes"] +
movie_df['actor_3_facebook_likes']
```

In[53]:

```
movie_df.drop('actor_2_facebook_likes',axis=1,inplace=True)
```

In[54]:

```
movie_df.drop('actor_3_facebook_likes',axis=1,inplace=True)
```

In[55]:

```
movie_df.drop('cast_total_facebook_likes',axis=1,inplace=True)
```

In[56]:

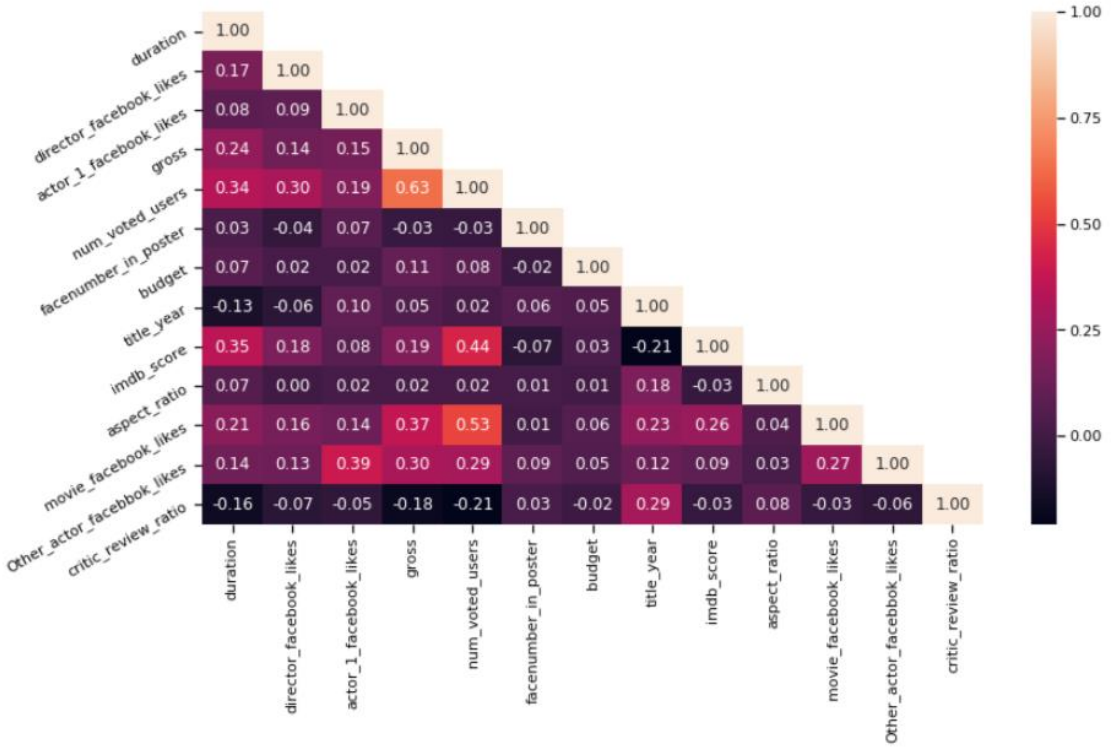
```
movie_df['critic_review_ratio']=movie_df['num_critic_for_reviews']/movie_df['num_user_for_reviews']
```

In[57]:

```
movie_df.drop('num_critic_for_reviews',axis=1,inplace=True)
movie_df.drop('num_user_for_reviews',axis=1,inplace=True)
```

In[58]:

```
import matplotlib.pyplot as plt
import seaborn as sns
corr = movie_df.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth": 2.5})
plt.figure(figsize=(13,7))
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
a = sns.heatmap(corr,mask=mask, annot=True, fmt='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=30)
```



In[59]:

```
movie_df["imdb_binned_score"]=pd.cut(movie_df['imdb_score'], bins=[0,4,6,8,10],
right=True, labels=False)+1
```

In[60]:

```
movie_df.drop('imdb_score',axis=1,inplace=True)
```

In[61]:

```
movie_df.head(5)
```

Out[61]:

	duration	director_facebook_likes	actor_1_facebook_likes	gross	num_voted_users	facenumber_in_poster	country	content_rating	budget	title_year
1937	142.0	0.0	11000.0	28341469.0	1689764	0.0	USA	R	25000000.0	1994.0
3466	175.0	0.0	14000.0	134821952.0	1155770	1.0	USA	R	6000000.0	1972.0
66	152.0	22000.0	23000.0	533316061.0	1676169	0.0	USA	PG-13	185000000.0	2008.0
2837	220.0	0.0	22000.0	57300000.0	790926	1.0	USA	R	13000000.0	1974.0
3355	178.0	16000.0	13000.0	107930000.0	1324680	1.0	USA	R	8000000.0	1994.0

Model Selection

We experimented with several machine learning models suitable for regression tasks, including

- Linear Regression
- Random Forest
- Gradient Boosting
- Neural Networks
- RandomForestRegressor – n_estimators, max_depth, min_samples_split, min_samples_leaf, random_state.
- XGBRegressor – n_estimators, learning rate, objective, reg_lambda, reg_alpha.
- The linear regression as a supervised machine learning algorithm that predicts a dependent variable based on one or more independent features.
- It distinguishes between univariate and multivariate linear regression based on the number of independent features.
- Goal of linear regression is to find the best linear equation that represents the relationship between the dependent and independent variables.
- The slope of the line as an indicator of how much the dependent variable changes for a unit change in the independent variable.

Program

Classification Model Selection

Logistic Regression

In[66]:

```
from sklearn.linear_model import LogisticRegression
logit =LogisticRegression()
logit.fit(X_train,np.ravel(y_train,order='C'))
y_pred=logit.predict(X_test)
```

In[67]:

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
print(cnf_matrix)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Out[67]:

```
[[  0  24  22   0]
 [  0  93 285   0]
 [  0  67 851   6]
 [  0   1  28  32]]
Accuracy: 0.6926898509581263
```

KNN

In[68]:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=22)
knn.fit(X_train, np.ravel(y_train,order='C'))
knnpred = knn.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, knnpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, knnpred))
```

Out[68]:

```
[[  0  24  22   0]
 [  0 155 223   0]
 [  0 153 771   0]
 [  0   2  47  12]]
Accuracy: 0.6657203690560681
```

SVC

In[69]:

```
from sklearn.svm import SVC
svc= SVC(kernel = 'sigmoid')
svc.fit(X_train, np.ravel(y_train,order='C'))
svcpred = svc.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, svcpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, svcpred))
```

Out[69]:

```
[[  1  36   9   0]
 [  2 157 219   0]
 [  4 175 730  15]
 [  0   8  33  20]]
Accuracy: 0.6444286728176012
```

Naive Bayes

In[70]:

```
from sklearn.naive_bayes import GaussianNB
```

```
gaussiannb= GaussianNB()
gaussiannb.fit(X_train, np.ravel(y_train,order='C'))
gaussiannbpred = gaussiannb.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, gaussiannbpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, gaussiannbpred))
```

Out[70]:

```
[[ 42   0   1   3]
 [326   1   1  50]
 [530   0   5 389]
 [  8   1   0  52]]
Accuracy: 0.07097232079488999
```

Decision Tree

In[71]:

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(criterion='gini')
dtree.fit(X_train, np.ravel(y_train,order='C'))
dtreepred = dtree.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, dtreepred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, dtreepred))
```

Out[71]:

```
[[  5  24  17   0]
 [ 24 194 160   0]
 [ 15 192 706  11]
 [  0   2  23  36]]
Accuracy: 0.6678495386799148
```

ADA Boosting

In[72]:

```
from sklearn.ensemble import AdaBoostClassifier
abcl = AdaBoostClassifier(base_estimator=dtree, n_estimators=60)
abcl=abcl.fit(X_train,np.ravel(y_train,order='C'))
abcl_pred=abcl.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, abcl_pred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, abcl_pred))
```

Out[72]:

```
[[  7  25  14   0]
 [ 20 194 164   0]
 [ 13 205 689  17]
 [  0   1  29  31]]
Accuracy: 0.6536550745209369
```

Random Forest

In[73]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 200)#criterion = entopy,gini
rfc.fit(X_train, np.ravel(y_train,order='C'))
rfcpred = rfc.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, rfcpred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, rfcpred))
```

Out[73]:

```
[[  2  31  13   0]
 [  0 177 201   0]
 [  0  77 846   1]
 [  0   0  34  27]]
Accuracy: 0.7466288147622427
```

Bagging Classifier

In[74]:

```
new_movie_df=movie_df.pop("imdb_binned_score")
```

In[75]:

```
from sklearn.ensemble import BaggingClassifier
bgcl = BaggingClassifier(n_estimators=60, max_samples=.7 , oob_score=True)

bgcl = bgcl.fit(movie_df, new_movie_df)
print(bgcl.oob_score_)
```

Out[75]:

```
0.7429179978700745
```

Gradient Boosting

In[76]:

```
from sklearn.ensemble import GradientBoostingClassifier
gbcl = GradientBoostingClassifier(n_estimators = 50, learning_rate = 0.09, max_depth=5)
gbcl = gbcl.fit(X_train,np.ravel(y_train,order='C'))
test_pred = gbcl.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, test_pred)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, test_pred))
```

Out[76]:

```
[[  1  38   7   0]
 [  0 211 167   0]
 [  2 100 817   5]
 [  1   0  30  30]]
Accuracy: 0.751596877217885
```

XG Boosting

In[77]:

```
from xgboost import XGBClassifier
xgb = XGBClassifier()
xgb.fit(X_train, np.ravel(y_train,order='C'))
xgbprd = xgb.predict(X_test)
cnf_matrix = metrics.confusion_matrix(y_test, xgbprd)
print(cnf_matrix)
print("Accuracy:",metrics.accuracy_score(y_test, xgbprd))
```

Out[77]:

```
[[  1  36   9   0]
 [  2 198 178   0]
 [  1 101 818   4]
 [  0   2  26  33]]
Accuracy: 0.7452093683463449
```

Model Training

The selected model was trained on a portion of the dataset, and hyperparameters were tuned for optimal performance

- The data collection procedure and the model implementation for predicting movie ratings using ensemble learning algorithms.
- It explains how the data was obtained from different sources such as IMDb, Wikipedia, and YouTube, and how it was pre-processed and transformed to extract relevant features.
- The use of two ensemble learning algorithms: Random Forest and XGBoost, and how they were implemented using Scikit-learn library in Python.
- The performance metrics used to evaluate the models, such as cross-validation score, mean squared error, root mean squared error, and mean absolute error.

Program

Splitting the Data into Training and test Data

In[64]:

```
X=pd.DataFrame(columns=['duration','director_facebook_likes','actor_1_facebook_likes','gr
oss','num_voted_users','facenumber_in_poster','budget','title_year','aspect_ratio','movie_fa
cebook_likes','Other_actor_facebbok_likes','critic_review_ratio','country_USA','country_oth
er','content_rating_G','content_rating_GP','content_rating_M','content_rating_NC-
17','content_rating_Not Rated','content_rating_PG','content_rating_PG-
```

```
13','content_rating_Passed','content_rating_R','content_rating_TV-14','content_rating_TV-G','content_rating_TV-PG','content_rating_Unrated','content_rating_X'],data=movie_df)

y=pd.DataFrame(columns=['imdb_binned_score'],data=movie_df)

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=100)
```

In[65]:

```
from sklearn.preprocessing import StandardScaler

sc_X = StandardScaler()

X_train = sc_X.fit_transform(X_train)

X_test = sc_X.transform(X_test)
```

Model Evaluation

The model's performance was assessed using various evaluation metrics, including

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-squared (R2)
- Mean square error is a way of measuring how accurate a statistical model or prediction is. It is calculated by taking the average of the squared differences between the actual and predicted values.
- Mean square error can be used for different types of models, such as regression, classification, or neural networks
- Root mean square error (RMSE) is a measure of how well a statistical model or prediction fits the observed data.
- RMSE also has some limitations and assumptions that need to be checked before using it. For example, RMSE may not be

appropriate if the data has outliers, non-linear relationships, or non-normal errors.

TABLE IV. PERFORMANCE METRICS (MOVIE METADATA + EXTRACTED FEATURES)

Performance Metric	Random Forest	XGBoost
Cross Validation Score	0.894	0.931
MSE	0.28	0.18
RMSE	0.53	0.42
MAE	0.37	0.22

TABLE V. PERFORMANCE METRICS (MOVIE METADATA + EXTRACTED FEATURES + SOCIAL MEDIA DATA)

Performance Metric	Random Forest	XGBoost
Cross Validation Score	0.907	0.953
MSE	0.16	0.09
RMSE	0.40	0.30
MAE	0.25	0.13

These metrics provided insights into the model's predictive accuracy and its ability to estimate IMDb scores effectively

Model Comparison

In[78]:

```
from sklearn.metrics import classification_report

print('Logistic  Reports\n',classification_report(y_test, y_pred))
```

```
print('KNN Reports\n',classification_report(y_test, knnpred))
print('SVC Reports\n',classification_report(y_test, svcpred))
print('Naive BayesReports\n',classification_report(y_test, gaussiannbpred))
print('Decision Tree Reports\n',classification_report(y_test, dtreepred))
print('Ada Boosting\n',classification_report(y_test, abcl_pred))
print('Random Forests Reports\n',classification_report(y_test, rfcpred))
print('Bagging Clasifier',bgcl.oob_score_)
print('Gradient Boosting',classification_report(y_test, test_pred))
print('XGBoosting\n',classification_report(y_test, xgbprd))
```

Out[78]:

```
Logistic  Reports
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        46
           2       0.50      0.25      0.33       378
           3       0.72      0.92      0.81       924
           4       0.84      0.52      0.65        61

    accuracy                           0.69      1409
   macro avg       0.52      0.42      0.45      1409
weighted avg       0.64      0.69      0.65      1409

KNN Reports
              precision    recall  f1-score   support

           1       0.00      0.00      0.00        46
           2       0.46      0.41      0.44       378
           3       0.73      0.83      0.78       924
           4       1.00      0.20      0.33        61

    accuracy                           0.67      1409
   macro avg       0.55      0.36      0.39      1409
weighted avg       0.64      0.67      0.64      1409
```

SVC Reports

	precision	recall	f1-score	support
1	0.14	0.02	0.04	46
2	0.42	0.42	0.42	378
3	0.74	0.79	0.76	924
4	0.57	0.33	0.42	61
accuracy			0.64	1409
macro avg	0.47	0.39	0.41	1409
weighted avg	0.62	0.64	0.63	1409

Naive BayesReports

	precision	recall	f1-score	support
1	0.05	0.91	0.09	46
2	0.50	0.00	0.01	378
3	0.71	0.01	0.01	924
4	0.11	0.85	0.19	61
accuracy			0.07	1409
macro avg	0.34	0.44	0.07	1409
weighted avg	0.61	0.07	0.02	1409

Decision Tree Reports

	precision	recall	f1-score	support
1	0.11	0.11	0.11	46
2	0.47	0.51	0.49	378
3	0.78	0.76	0.77	924
4	0.77	0.59	0.67	61
accuracy			0.67	1409
macro avg	0.53	0.49	0.51	1409
weighted avg	0.67	0.67	0.67	1409

Ada Boosting

	precision	recall	f1-score	support
1	0.17	0.15	0.16	46
2	0.46	0.51	0.48	378
3	0.77	0.75	0.76	924
4	0.65	0.51	0.57	61
accuracy			0.65	1409
macro avg	0.51	0.48	0.49	1409
weighted avg	0.66	0.65	0.66	1409

Random Forests Reports

	precision	recall	f1-score	support
1	1.00	0.04	0.08	46
2	0.62	0.47	0.53	378
3	0.77	0.92	0.84	924
4	0.96	0.44	0.61	61
accuracy			0.75	1409
macro avg	0.84	0.47	0.52	1409
weighted avg	0.75	0.75	0.72	1409

Bagging Clasifier 0.7429179978700745

Gradient Boosting precision recall f1-score support

1	0.25	0.02	0.04	46
2	0.60	0.56	0.58	378
3	0.80	0.88	0.84	924
4	0.86	0.49	0.62	61
accuracy			0.75	1409
macro avg	0.63	0.49	0.52	1409
weighted avg	0.73	0.75	0.74	1409

XGBoosting

	precision	recall	f1-score	support
1	0.25	0.02	0.04	46
2	0.59	0.52	0.55	378
3	0.79	0.89	0.84	924
4	0.89	0.54	0.67	61
accuracy			0.75	1409
macro avg	0.63	0.49	0.53	1409
weighted avg	0.72	0.75	0.73	1409

Conclusion

- The conclusion is that Random Forest Algorithm along with the gradient boosting have the accuracy of 74.5 and 75.5 respectively
- This analysis compares certain attributes regarding Facebook and IMDB site against the gross revenue of a film.
- The higher number of Facebook likes from the primary actor and supporting actors plays a significant role in generating revenue from a film.
- Through both models and the sensitivity analysis, someone can easily see the support in this conclusion. Directors and producers can take this dataset and implement it into their thought process when planning their movie.
- Movie-goers can use this dataset to make the same predictions once the movie is announced with primary and supporting actors/actresses. It could possibly save movie-goers money when debating on whether to go see a movie or not.
- This project has helped me substantially in practicing with running analysis on certain topics and generating a result.
- It has developed my skill in Excel and XL Miner by using the Missing Data handle, the Reduce Categories handle, the Data Partition handle, the Multiple Linear Regression handle, and a sensitivity analysis.
- Overall, the effectiveness of this project was very useful for me in the preparation for my career. I can take this project as proof of knowledge in these areas as well as knowing associated terms.