IEOR 142 Final Project
Group Members: William Lu, Brandon Hong,
Alan Edmonds, Wesley Hopkins, Jae Chung

Blockbuster or Box Office Bomb?

Machine Learning Predictions of Film Performance

Introduction

For our final project, we decided to look at movies. We liked this domain for a few reasons: first, they're a cultural staple that a wide variety of people pay some attention to. Most people see at least a few movies per year, and have opinions about how good or bad they are. It's always more interesting to study topics that are relevant to lots of people. We also liked this domain because discussions about the quality of movies are typically grounded in subjective opinions, and don't often include any sort of quantitative metrics. It got us interested in looking at them through a more quantitative lens. We wondered, would it be possible to use data about a film to determine whether it would be successful? This leads us to the last reason for this domain, which is the availability of data. As an industry worth over $25 billion in annual revenue, Hollywood is heavily studied and there exist massive datasets that are publicly accessible, relatively clean, and are overall good fits for the machine learning algorithms that we'll be discussing here.

Defining the Problem

Knowing that we wanted to predict the success of movies, we had to define what success means for our purposes. The simplest measure of success is financial: **the gross revenue of the movie**, and we included this in our analysis. This outcome would be the most relevant for a producer who might want to use a model like ours to determine whether to finance a particular movie. However, this didn't seem to capture the whole story, because some movies that perform exceptionally well in the box office don't really become cultural staples the way other movies can. For example, Marvel Studios has released 17 movies in the past two years alone and they consistently do well at the box office, but they don't achieve the kind of cultural significance that other movies do even with lower box office numbers. This is why we wanted to include a more subjective measure in addition to the box office revenue. We considered using critical aggregation scores like Rotten Tomatoes or IMDB, but eventually settled on **Academy Award (Oscar) nominations** as the best outcome to predict. The Oscars are the most watched award show in the US, and for all its flaws the selections that "The Academy" make are generally highly regarded. We chose to use nominations rather than winners because of the higher volume of nominees, which we believed would produce a more predictive model than one trained on finding patterns among only a few winners out of thousands of movies. By using these two outcome variables, we also had an opportunity to apply both regression and classification methods.

Data Gathering and Cleaning

The first dataset we found was `movies_metadata`, which included metadata about over 45,000 movies released between 1900 and 2017(Appendix A). This was the main dataset that contained important columns, such as whether the movie belonged to a collection, budget, revenue, etc. The second dataset we found was `database`, which contained data about around

10,000 movies released between 1927 and 2015(Appendix B). This dataset included important information about whether the movie was awarded or nominated for an Oscar. The final dataset we found was `MoviesGenre`, which contained data of over 40000 movies. It mainly had the IMDB Score and Genre of movies (Appendix C).

Using the merge function from pandas, we were able to combine `movies_metadata` with `database` to create a new dataset called `merged`. We used a left join to merge these datasets because we only wanted the rows from the `movies_metadata` dataset. Afterwards, we created a new column called "Nominated," which basically stated whether a movie was nominated for an Oscar or not. We were able to deduce this from the "Winner" column in the database dataset. If a movie had a null value in the "Winner" column, this meant that it was not nominated for an Oscar. Movies that had "1.0" or "0.0" meant that they were nominated for an Oscar and received a 1.0 for the new "Nominated" column. Before we could merge the final datasets, we removed the rows without an IMDB_ID in `merged` to prevent any excess rows from being merged. The `merged` dataset had two letters in front of each IMDB_ID, which I removed using regex. Afterwards, we took `merged` and performed a left join with `MoviesGenre` to get the final dataset called `combined_movies`.

After combining all of our datasets, we began to clean our data to use in our models. We first removed the rows that did not include a release date. We used regex to extract the Year from the release date and create a new column called "Year." (Appendix D) Since the Oscars data only included data from 1927 and 2015, we used the new "Year" column to remove movies that were produced before and after this time period. Afterwards, we proceeded to go through all the columns that seemed usable in our models and checked for null values. If a column contained a null value, we checked those rows with null values and removed those rows from the dataset if they did not have any major importance. Since we had over 40,000 rows of data, removing rows would not have a major effect on our models. We also used some more regex to extract the main genre of each movie and removed any rows that did not contain any genre (Appendix E). The most important step was removing movies that did not have an IMDB Score higher than 7.0. This dramatically reduced our dataset from 40,000 rows to 12,000 rows. There are simply too many movies to use the whole dataset, and restricting to movies that received a baseline level of acclaim from critics seemed like a good way to set boundaries. After removing movies with an IMDB Score less than 7, we used dummy variables on the "Genre" and "status" columns to create more viable columns. The final step was to separate the data into training and testing data, which we did by setting training data to movies before 2001 and testing data to years after 2000.

Predictive Models

Baseline Models:
In order to find the Baseline Model for our classification objective, we found the most common value in the "Nominated" column of the training set. Since 0 or not being nominated for an Oscar was the most common value, we predicted that none of the movies in the testing set would receive an Oscar. This received an accuracy of 95.92%. However this does not give us a baseline for which to compare precision of our models; for this purpose we created a "precision-baseline" model which takes the positive rate within training data and predicts test data to be positive at the same rate. Defined as such, the precision-baseline model had <5% precision on the test data.

Linear Regression:
Originally, there were 18 independent variable inputs to predict the revenue of a movie in our linear regression model, but we used the p-values to remove certain variables. If the p-value

was greater than .05, we felt that the variable was statistically insignificant and would most likely have a coefficient of 0 in the model (Please refer to Appendix F). After removing 10 variables, we achieved a linear regression that had statistically significant variables, as well as VIF scores of less than 5 (Appendix G). We were able to achieve a model with a .659 R2 value, as well as a RMSE value of 53880210 and OSR2 value of .804. At first glance, this is a huge RMSE value. However, we have to consider that movies often make millions of dollars and certain movies tend to make more than others. Therefore, a difference of approximately 54 million dollars for each movie is reasonable.

Logistic Regression Classifier:
Similar to linear regression, we originally used 18 independent variable inputs to predict whether a movie would be nominated for an Oscar. We removed variables based on the p-values and ended with 12 variables for our logistic regression model(Appendix H). After making a confusion matrix and solving for the accuracy, we were able to receive a .957 accuracy. Although this seems like a very high number, it is actually lower than our baseline model. The reason for this can be explained in our ROC Curve (Appendix I). The graph clearly shows that there is an extremely low TPR rate at the kink. Upon manually solving for the TPR, there was a TPR of .112. For our objective of predicting an Oscar, precision is also a useful metric to gauge the hit-rate of positive predictions as we have learned in this class. The precision of a logistic regression with threshold p = 0.5 was 0.276, much higher than the TPR. Surprisingly, higher p thresholds did not lead to better precision even though the data is heavily negative-skewed.

Decision Tree Regression:
We used cross-validation classification based on mean squared error scoring to find the best alpha value for our model. Afterwards, we used this best alpha value to predict the revenue of movies in the training and testing datasets (Appendix J). We were able to find that the training set had a RMSE value of 28226066, while the testing set had a RMSE value of 70093476. As expected, the training set had a lower RMSE value because decision trees tend to overfit. The decision tree regression model RMSE had a higher RMSE value than the linear regression model.

Decision Tree Classification:
In the decision tree classification model, we used cross validation once again to get the best alpha value. We were able to find that the best alpha value is .00235, which we promptly used to predict whether a movie would be nominated for an Oscar (Appendix K). After using it in the model, we were able to achieve an accuracy of .8969 on the training set and .9371 on the testing set. This is somewhat surprising because our model performed better on the testing set than the training set. The precision of this model on the testing data was 0.319. Although the accuracy is slightly lower than the baseline test accuracy of 0.959, the precision of this model of 0.319 is much higher than the baseline precision of ~0.05.

Random Forest Regression:
Similar to the decision tree models, we used cross validation for the random forest regression model to predict the revenue of movies. However, we did not search for an alpha value this time, and we used cross validation to determine the maximum number of features. This cross validation took nearly 15 minutes to complete, but we were able to find the maximum number of features and use it in our model. Using mean squared error as our scoring feature and square rooting it, we were able to achieve a RMSE value of 30170832 for our training set and 75727885 for our testing set. Once again, the training and testing set RMSE values have increased, which is not a good sign. Our model once again overfits on our training set, so it does a slightly poorer job on our testing set.

Random Forest Classification:
Using a combination of grid search and our intuition for the underlying structure of the random forest classifier, we settled with number of trees in the forest = 500, min samples per leaf of 5, and a very low value of max_features of 1 which creates a much more random and diverse set of individual tree learners. This selection of hyperparameters provided a good balance between the size and dimensions of our data and the structure of the RF model. The resulting test accuracy of 0.958 was essentially the same as our baseline model, however the precision of our RF classifier was 0.48, extremely high compared to the precision of our other models as well as the precision-baseline model. Out of the classification models, our RF classifier was by far the most precise model for predicting Oscars, and much of this is likely down to the hyperparameter optimization.

Ensemble Predictor:
Our ensemble model gathered the predictions of the random forest, decision tree, and logistic regression models yielding a new column, representing the mean of these models' target columns. The ensemble predicted positive if this mean was above a certain threshold. Higher threshold values led the predictor to degenerate into the naive baseline model, so after tuning the threshold parameter we ended up at a near optimal value of ¼ giving the ensemble model test accuracy of 0.930 and test precision of 0.300.

Conclusion: Impact, Uses, and Drawbacks

In this project we dealt with two main objectives: implementing regression models to predict gross revenue, and implementing classification models to predict Oscar nominations. Using metrics reflecting our goals, we gauged the performance of these models, and used the techniques detailed above in adjusting models and improving their performance. Provided in this conclusive section are discussions of impact and practical usage of our projects, possible drawbacks of the methods used, and future directions for improvement.

The impact of the models dealing with the classification versus those dealing with regression objectives are different, and have very different potential uses. The overarching impact of our project's analysis is on creating models that can predict the "quality" of movies (whether defined as revenue or award winning capability), and this discussion may be expanded to the question of whether movie critics and ratings can be replaced by such models. But beyond the overarching impact of the questions raised by our project, there are very specific and practical use cases of our implemented models. Our regression models are tasked with predicting gross revenue, and thus may serve as a useful tool for movie producers, or any other individuals in the process of deciding whether to invest in a movie or not. Our classification models are tasked with predicting whether a movie will win an Oscar, whose practical uses may include providing a bettor an edge over the gambling market, or conversely may help a bookmaking company post better odds and reduce their risk.

A discussion of the practical uses of the project brings us also to the possible drawbacks of our current methods, which do have solutions but must be addressed. The primary one to mention is the data we are currently using in predicting gross revenue. Currently, our models use a set of input features that includes quantities such as popularity. In the practical usage of deciding whether or not to invest in a movie, we won't have information about popularity of the movie at that stage and thus it may be problematic for our models to use popularity as an input feature in predicting gross revenue. But there is a solution to this type of problem; we can substitute this feature for a similar one that we would indeed have data on before the time of movie production. For example the popularity of an upcoming movie concept is often assessed ahead of time through private focus groups, and with access to that data we might be able to replicate the predictive value of the popularity feature.

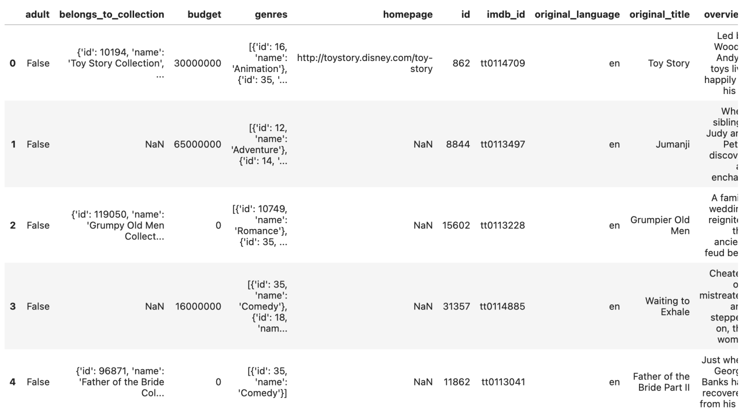
Lastly, in seeking future improvements of performance for our objectives: there may be room for using a neural network on analyzing text data (an exceptionally well designed neural net may even use on screen dialogue) and image data relevant to the movie. We attempted implementing a neural network, similar in structure to the neural network in discussion, earlier in the project but did not succeed in applying it appropriately for our input feature data, and dedicated our efforts instead to optimizing our current set of models. Also, it may be beneficial to add gradient boosting classifiers and gradient boosting regressors to our project models, which would indeed be very appropriate for the datasets and target columns we are currently working with, and we will also reserve this idea for possible future improvements of this project.

Appendix

Data Code:
https://github.com/00alan/IEOR-142-Final-Proj/blob/main/movieanalysis.ipynb

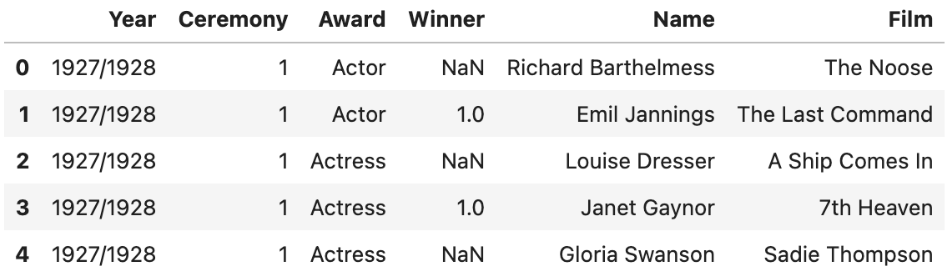
Data Sources:
movies_metadata:
https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?select=ratings.csv
database:
https://www.kaggle.com/datasets/theacademy/academy-awards
Movies_Genre:
https://www.kaggle.com/datasets/neha1703/movie-genre-from-its-poster?select=MovieGenre.csv

Reference Figures:



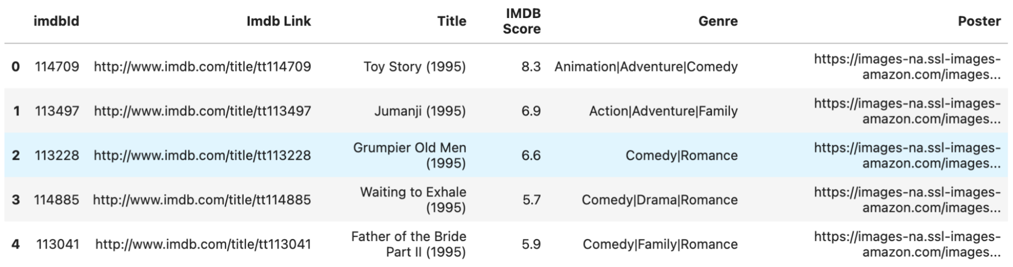
	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overvie
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led I Wooc Andy toys li happily his
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN	8844	tt0113497	en	Jumanji	Whe sibling Judy ar Pet discov a encha
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN	15602	tt0113228	en	Grumpier Old Men	A fami weddir reignite th ancie feud be
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN	31357	tt0114885	en	Waiting to Exhale	Cheate o mistreate ar steppe on, th wom
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN	11862	tt0113041	en	Father of the Bride Part II	Just whe Georg Banks ha recovere from his

A.



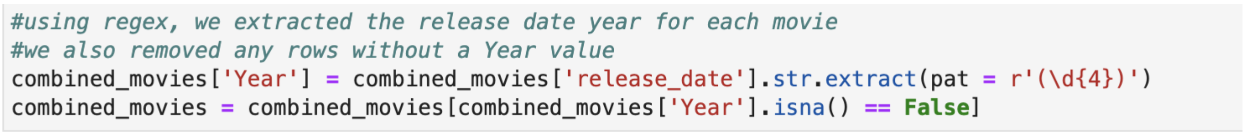
	Year	Ceremony	Award	Winner	Name	Film
0	1927/1928	1	Actor	NaN	Richard Barthelmess	The Noose
1	1927/1928	1	Actor	1.0	Emil Jannings	The Last Command
2	1927/1928	1	Actress	NaN	Louise Dresser	A Ship Comes In
3	1927/1928	1	Actress	1.0	Janet Gaynor	7th Heaven
4	1927/1928	1	Actress	NaN	Gloria Swanson	Sadie Thompson

B.



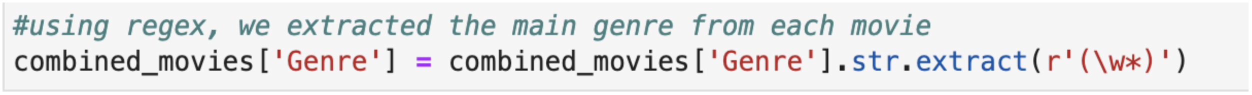
	imdbId	Imdb Link	Title	IMDB Score	Genre	Poster
0	114709	http://www.imdb.com/title/tt114709	Toy Story (1995)	8.3	Animation|Adventure|Comedy	https://images-na.ssl-images-amazon.com/images...
1	113497	http://www.imdb.com/title/tt113497	Jumanji (1995)	6.9	Action|Adventure|Family	https://images-na.ssl-images-amazon.com/images...
2	113228	http://www.imdb.com/title/tt113228	Grumpier Old Men (1995)	6.6	Comedy|Romance	https://images-na.ssl-images-amazon.com/images...
3	114885	http://www.imdb.com/title/tt114885	Waiting to Exhale (1995)	5.7	Comedy|Drama|Romance	https://images-na.ssl-images-amazon.com/images...
4	113041	http://www.imdb.com/title/tt113041	Father of the Bride Part II (1995)	5.9	Comedy|Family|Romance	https://images-na.ssl-images-amazon.com/images...

C.



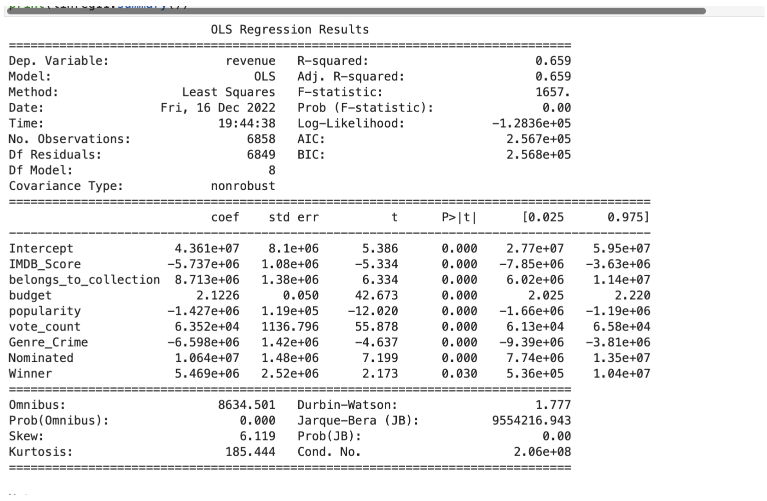
```
#using regex, we extracted the release date year for each movie
#we also removed any rows without a Year value
combined_movies['Year'] = combined_movies['release_date'].str.extract(pat = r'(\d{4})')
combined_movies = combined_movies[combined_movies['Year'].isna() == False]
```

D.



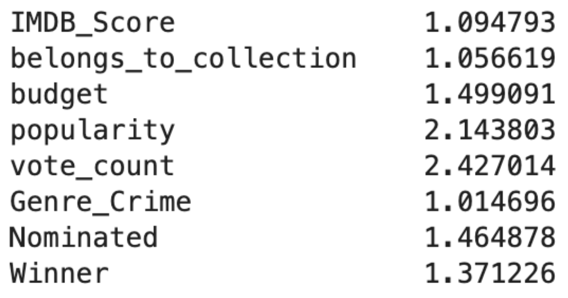
```
#using regex, we extracted the main genre from each movie
combined_movies['Genre'] = combined_movies['Genre'].str.extract(r'(\w*)')
```

E.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                revenue   R-squared:                       0.659
Model:                            OLS   Adj. R-squared:                  0.659
Method:                 Least Squares   F-statistic:                     1657.
Date:                Fri, 16 Dec 2022   Prob (F-statistic):               0.00
Time:                        19:44:38   Log-Likelihood:            -1.2836e+05
No. Observations:                6858   AIC:                         2.567e+05
Df Residuals:                    6849   BIC:                         2.568e+05
Df Model:                           8
Covariance Type:            nonrobust
=========================================================================================
                            coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Intercept              4.361e+07    8.1e+06      5.386      0.000    2.77e+07    5.95e+07
IMDB_Score            -5.737e+06   1.08e+06     -5.334      0.000   -7.85e+06   -3.63e+06
belongs_to_collection  8.713e+06   1.38e+06      6.334      0.000    6.02e+06    1.14e+07
budget                    2.1226      0.050     42.673      0.000       2.025       2.220
popularity            -1.427e+06   1.19e+05    -12.020      0.000   -1.66e+06   -1.19e+06
vote_count             6.352e+04   1136.796     55.878      0.000    6.13e+04    6.58e+04
Genre_Crime           -6.598e+06   1.42e+06     -4.637      0.000   -9.39e+06   -3.81e+06
Nominated              1.064e+07   1.48e+06      7.199      0.000    7.74e+06    1.35e+07
Winner                 5.469e+06   2.52e+06      2.173      0.030    5.36e+05    1.04e+07
==============================================================================
Omnibus:                     8634.501   Durbin-Watson:                   1.777
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          9554216.943
Skew:                           6.119   Prob(JB):                         0.00
Kurtosis:                     185.444   Cond. No.                     2.06e+08
==============================================================================
```

F.

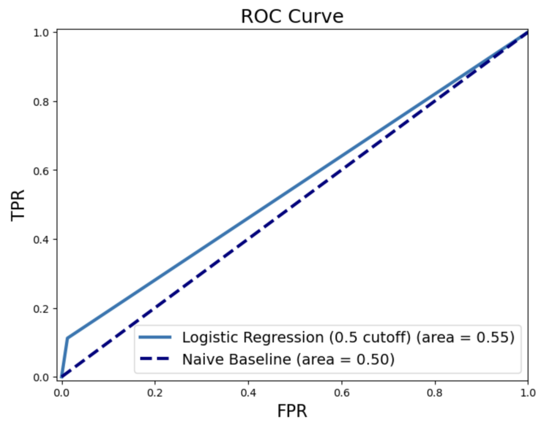


```
IMDB_Score               1.094793
belongs_to_collection    1.056619
budget                   1.499091
popularity               2.143803
vote_count               2.427014
Genre_Crime              1.014696
Nominated                1.464878
Winner                   1.371226
```

G.

```
Optimization terminated successfully.
         Current function value: 0.286657
         Iterations 11
                           Logit Regression Results
==============================================================================
Dep. Variable:              Nominated   No. Observations:                 6858
Model:                          Logit   Df Residuals:                     6845
Method:                           MLE   Df Model:                           12
Date:                Thu, 15 Dec 2022   Pseudo R-squ.:                  0.2081
Time:                        05:48:43   Log-Likelihood:                -1965.9
converged:                       True   LL-Null:                       -2482.4
Covariance Type:            nonrobust   LLR p-value:                 1.526e-213
=========================================================================================
                            coef    std err          z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------------
Intercept                47.8647      4.308     11.110      0.000      39.421      56.309
belongs_to_collection    -1.3759      0.220     -6.260      0.000      -1.807      -0.945
popularity                0.1913      0.012     16.443      0.000       0.169       0.214
revenue                9.554e-09   1.07e-09      8.920      0.000     7.45e-09    1.17e-08
runtime                   0.0059      0.001      7.113      0.000       0.004       0.008
vote_average              0.1135      0.039      2.915      0.004       0.037       0.190
vote_count               -0.0008      0.000     -6.333      0.000      -0.001      -0.001
Year                     -0.0264      0.002    -11.967      0.000      -0.031      -0.022
Genre_Drama               0.4288      0.100      4.285      0.000       0.233       0.625
Genre_Documentary        -3.8275      1.070     -3.578      0.000      -5.924      -1.731
Genre_Comedy             -0.2101      0.122     -1.721      0.085      -0.449       0.029
Genre_Action             -1.3092      0.241     -5.422      0.000      -1.782      -0.836
Genre_Animation          -2.1824      0.470     -4.641      0.000      -3.104      -1.261
=========================================================================================
```

H.



I.

```
[111]: indices = c_xte.index
```

```
GridSearchCV(cv=10, estimator=DecisionTreeRegressor(),
             param_grid={'ccp_alpha': array([0.00e+00, 5.00e-05, 1.00e-04, 1.50e-04, 2.00e-04, 2.50e-04,
       3.00e-04, 3.50e-04, 4.00e-04, 4.50e-04, 5.00e-04, 5.50e-04,
       6.00e-04, 6.50e-04, 7.00e-04, 7.50e-04, 8.00e-04, 8.50e-04,
       9.00e-04, 9.50e-04, 1.00e-03, 1.05e-03, 1.10e-03, 1.15e-03,
       1.20e-03, 1.25e-03, 1.30e-03, 1.35e-03, 1.40e-03, 1.45e...
       8.70e-03, 8.75e-03, 8.80e-03, 8.85e-03, 8.90e-03, 8.95e-03,
       9.00e-03, 9.05e-03, 9.10e-03, 9.15e-03, 9.20e-03, 9.25e-03,
       9.30e-03, 9.35e-03, 9.40e-03, 9.45e-03, 9.50e-03, 9.55e-03,
       9.60e-03, 9.65e-03, 9.70e-03, 9.75e-03, 9.80e-03, 9.85e-03,
       9.90e-03, 9.95e-03, 1.00e-02]),
                         'max_depth': [10], 'min_samples_leaf': [5],
                         'min_samples_split': [20], 'random_state': [88]},
             scoring=make_scorer(mean_squared_error), verbose=1)
```

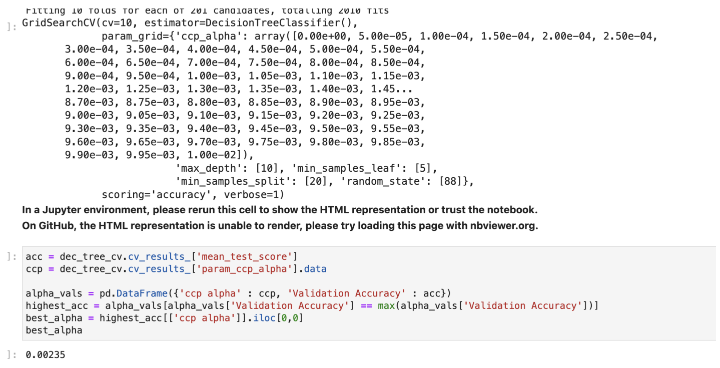
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
acc = dec_tree_cv.cv_results_['mean_test_score']
ccp = dec_tree_cv.cv_results_['param_ccp_alpha'].data

alpha_vals = pd.DataFrame({'ccp alpha' : ccp, 'Validation RMSE' : acc**.5})
highest_acc = alpha_vals[alpha_vals['Validation RMSE'] == max(alpha_vals['Validation RMSE'])]
best_alpha = highest_acc[['ccp alpha']].iloc[0,0]
best_alpha
```

```
0.0
```

J.



```
Fitting 10 folds for each of 201 candidates, totalling 2010 fits
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'ccp_alpha': array([0.00e+00, 5.00e-05, 1.00e-04, 1.50e-04, 2.00e-04, 2.50e-04,
       3.00e-04, 3.50e-04, 4.00e-04, 4.50e-04, 5.00e-04, 5.50e-04,
       6.00e-04, 6.50e-04, 7.00e-04, 7.50e-04, 8.00e-04, 8.50e-04,
       9.00e-04, 9.50e-04, 1.00e-03, 1.05e-03, 1.10e-03, 1.15e-03,
       1.20e-03, 1.25e-03, 1.30e-03, 1.35e-03, 1.40e-03, 1.45...
       8.70e-03, 8.75e-03, 8.80e-03, 8.85e-03, 8.90e-03, 8.95e-03,
       9.00e-03, 9.05e-03, 9.10e-03, 9.15e-03, 9.20e-03, 9.25e-03,
       9.30e-03, 9.35e-03, 9.40e-03, 9.45e-03, 9.50e-03, 9.55e-03,
       9.60e-03, 9.65e-03, 9.70e-03, 9.75e-03, 9.80e-03, 9.85e-03,
       9.90e-03, 9.95e-03, 1.00e-02]),
                         'max_depth': [10], 'min_samples_leaf': [5],
                         'min_samples_split': [20], 'random_state': [88]},
             scoring='accuracy', verbose=1)
```

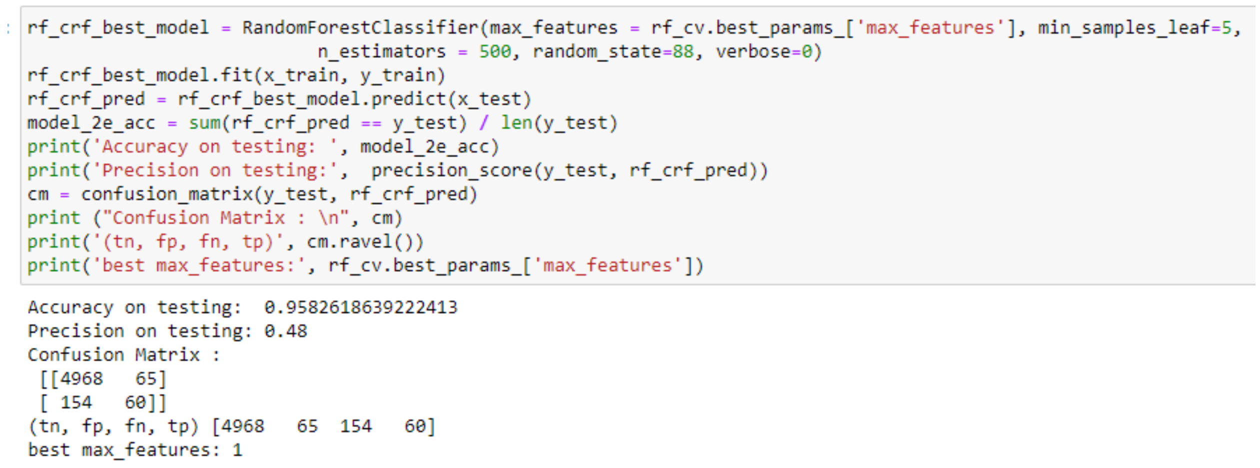
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```python
acc = dec_tree_cv.cv_results_['mean_test_score']
ccp = dec_tree_cv.cv_results_['param_ccp_alpha'].data

alpha_vals = pd.DataFrame({'ccp alpha' : ccp, 'Validation Accuracy' : acc})
highest_acc = alpha_vals[alpha_vals['Validation Accuracy'] == max(alpha_vals['Validation Accuracy'])]
best_alpha = highest_acc[['ccp alpha']].iloc[0,0]
best_alpha
```

```
0.00235
```

K.



```python
rf_crf_best_model = RandomForestClassifier(max_features = rf_cv.best_params_['max_features'], min_samples_leaf=5,
                                           n_estimators = 500, random_state=88, verbose=0)
rf_crf_best_model.fit(x_train, y_train)
rf_crf_pred = rf_crf_best_model.predict(x_test)
model_2e_acc = sum(rf_crf_pred == y_test) / len(y_test)
print('Accuracy on testing: ', model_2e_acc)
print('Precision on testing:',  precision_score(y_test, rf_crf_pred))
cm = confusion_matrix(y_test, rf_crf_pred)
print ("Confusion Matrix : \n", cm)
print('(tn, fp, fn, tp)', cm.ravel())
print('best max_features:', rf_cv.best_params_['max_features'])
```

```
Accuracy on testing:  0.9582618639222413
Precision on testing: 0.48
Confusion Matrix :
 [[4968   65]
 [ 154   60]]
(tn, fp, fn, tp) [4968   65  154   60]
best max_features: 1
```

L.