Arrius Clarke

Dr. Barough

DS-655/ Deliverable #1

11 October 2023

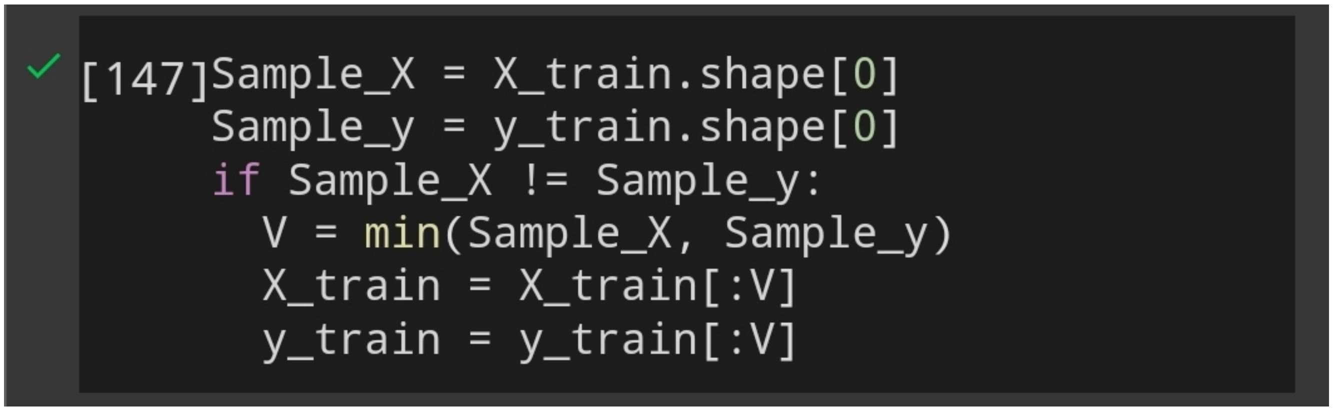
Linear Regression and Decision Tree Implementation on a Movie Dataset

The dataset chosen for this project is a movie database. Movies have been around for a little over a century, and a good number of them each year tend to make millions (sometimes billions) of dollars in the box office. The process of determining what ensures how much a film will make can be very difficult as a lot of it is based on subjectivity. The movie industry is a business, so like any business there must be marketing, testing of the product, noting feedback from the consumer, etc. If people respond well to what is being shown to them, they will spread the word to people they know. Other factors can include: when the movie was released, how long the movie was, and the genre of the movie. The dataset collected for this assignment is from this website: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset/data . The oldest films listed in the dataset had been released in 1995.

There are seven tables in this dataset. The tables are as follows: ratings, ratings_small, movies metadata, keywords, credits, links_small, and links. These were all csv files downloaded from the website. Python code was used in Google Colaboratory, where the data for each table was inspected. During the inspection of each table, there was a search for the number of unique values in each column and the number of null values in each column. The main table inspected was the movies metadata table, which had the most information on it. One of the goals was to merge that table with at least one of the other tables. Most of the other tables could not be joined with the main table. The ratings tables did not have features that connected it to the movie metadata table. The keywords and credits tables both had the feature “id” in common. The only problem is that both of those tables had no other numerical columns except for “id”. The only two tables that seemingly had a direct connection to the other (without any categorical variables) were the movies metadata table and the links table. When attempting to merge the tables together, there was one problem. The problem was that even though both tables had a column dedicated to IMDB ratings, there was a difference in how the data was distributed. The movies metadata table listed the ratings with two ‘t’s in front of each identification number, whereas the links table listed the ratings as fully numerical figures. The only table that could advance into the next step was the movies metadata table.

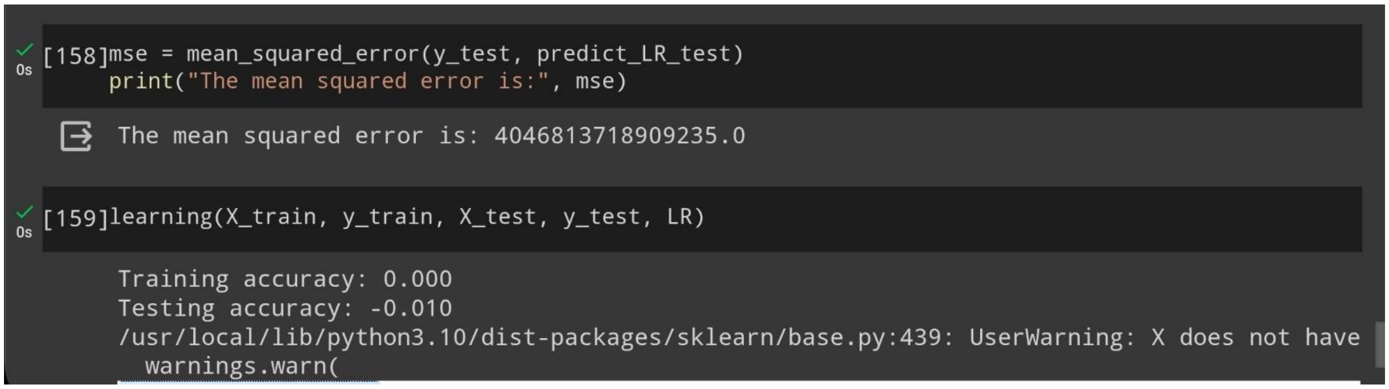
The classification problems involved with this dataset had to do with Linear Regression and Decision Trees (with some pruning). The main table (movies metadata) initially had 24 columns: adult, belongs_to_collection, budget, genres, homepage, id, imdb_id, original_language, original_title, overview, popularity, poster_path, production_companies, production_countries, release_date, revenue, runtime, spoken_languages, status, tagline, title, video, vote_average, and vote_count. There were 19 out of 24 of these that needed to be dropped from the table since most of them were categorical data. The next thing was to inspect the budget column because that was the only column out of the remaining five that had ‘object’ as a datatype. To change this, it needed to be converted into a ‘float64’ datatype. The one thing to remember was to drop any null values in the remaining data before doing anything else. Once this was done, the next step was to implement a Linear Regression model.

Trying to implement the Linear Regression model for this table was an involved process that produced various challenges. The first thing to do was to split the data into training and testing sets. The reason for this is to make sure that the performance predictions are not biased. The label X was assigned to the budget, runtime, vote average, and vote count columns. The label y was assigned to the revenue column. The goal at first was to use the four features in a linear regression model and to test the accuracy of that model regarding the established y-value. After the data was split into the training and testing sets, both sets for each variable had to have the same number of samples, otherwise the data could not be fitted to the training data. The model would not fit into the training data because X_train had over 30,000 samples while y_train only had just over 9,000 samples. This is important because without fitting the model to this data, there is no way to check the accuracy of a model. To fix this problem, X_train had to be reshaped to the size of y_train. To do this there had to be the if statement shown below:



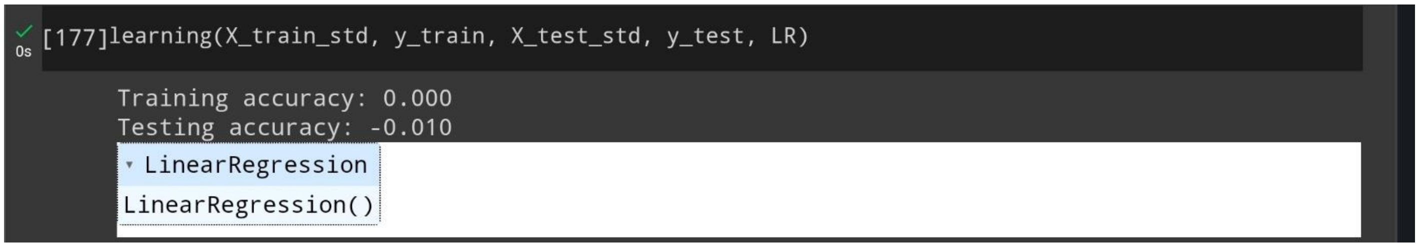
```
[147]Sample_X = X_train.shape[0]
     Sample_y = y_train.shape[0]
     if Sample_X != Sample_y:
       V = min(Sample_X, Sample_y)
       X_train = X_train[:V]
       y_train = y_train[:V]
```

After this change, the training data was able to be fitted. When implementing a function to predict the testing set, an error message popped up saying that there is no room for all four features. As a result of this, the only column used (at this point) moving forward was the budget column. This was an unforeseen issue. The next step was to implement the mean squared value error, which was the next problem. The value received for the mean squared error was too big to be considered possible.



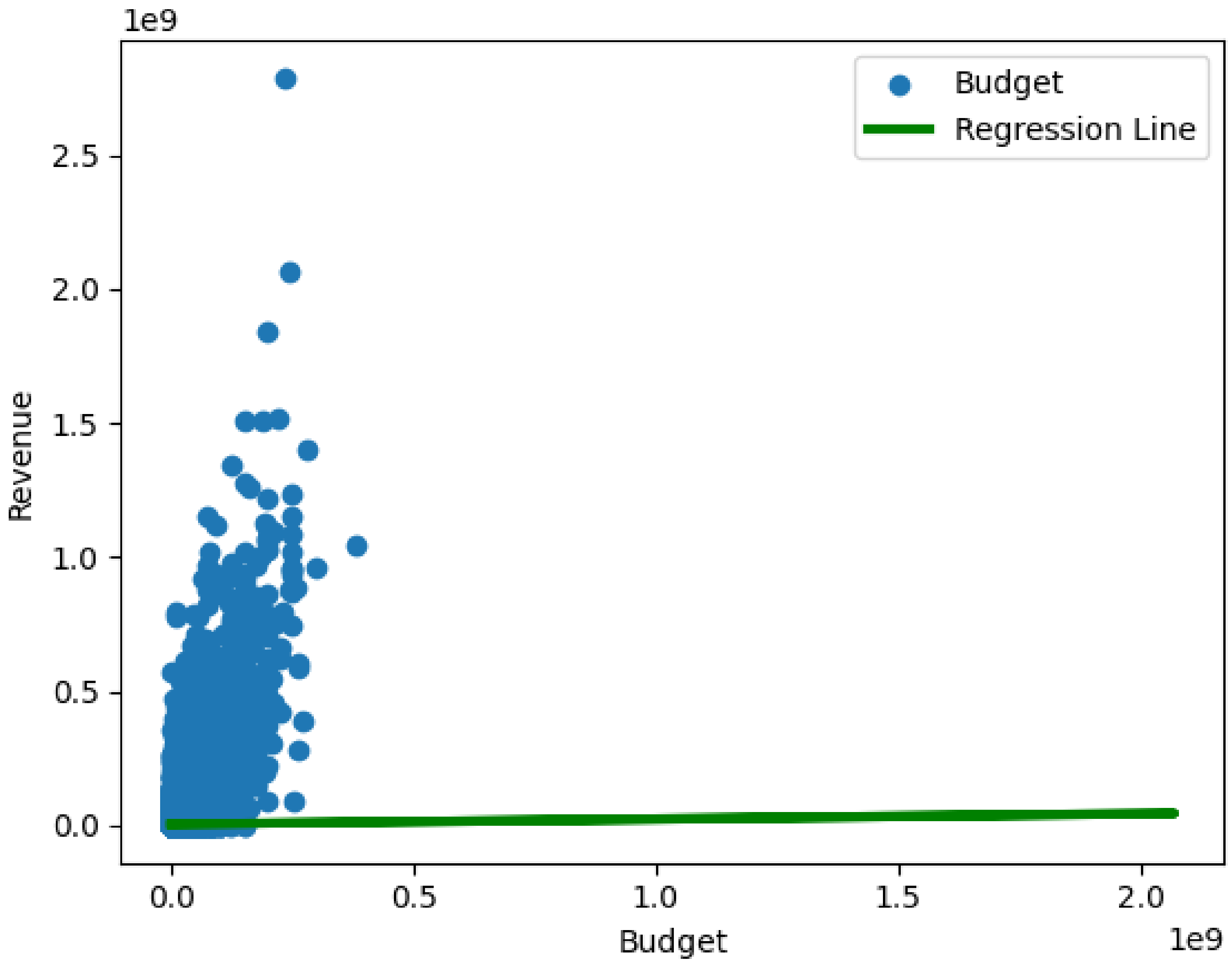
```
[158]mse = mean_squared_error(y_test, predict_LR_test)
     print("The mean squared error is:", mse)

     The mean squared error is: 4046813718909235.0

[159]learning(X_train, y_train, X_test, y_test, LR)

      Training accuracy: 0.000
      Testing accuracy: -0.010
      /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have
        warnings.warn(
```

The above mean squared error value is basically stating that this model has a huge amount of error in it. It is an indication that this model will most likely not work with this dataset. This following this step was checking the training and testing accuracy for this model (linear regression). The training accuracy was zero and the test accuracy was negative. This meant that the model is not at all accurate in any way. This also meant that this model, in addition to using the budget column as the only X variable, proved to be a very poor predictor for the revenue a movie makes. When the data was transformed to the Standard Scaler model, the results were still the same.



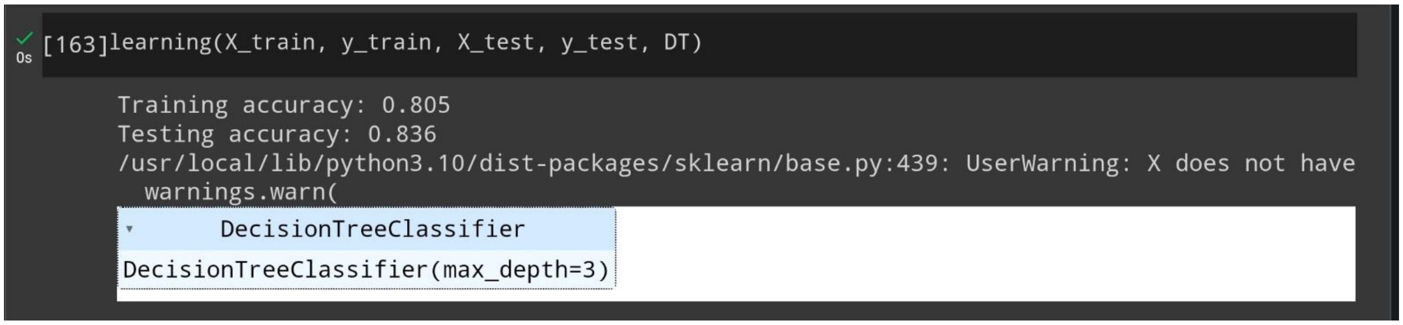
```
[177]learning(X_train_std, y_train, X_test_std, y_test, LR)

      Training accuracy: 0.000
      Testing accuracy: -0.010
      ▾ LinearRegression
      LinearRegression()
```

The graph of the result is shown below:

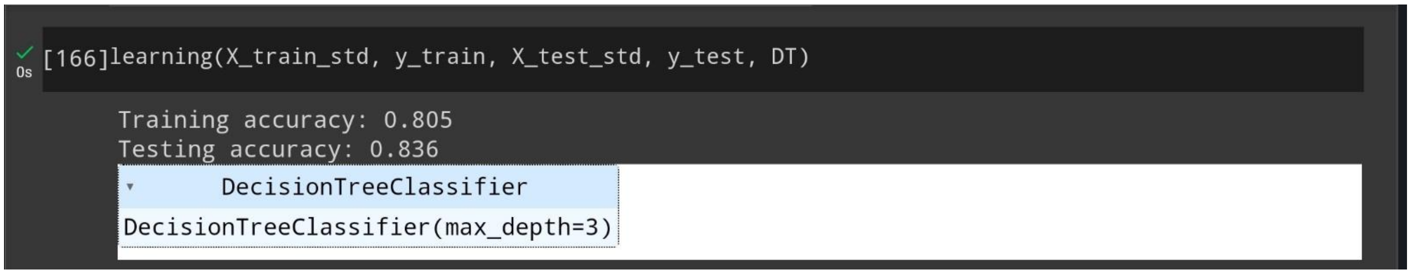


This shows a very poor correlation between Budget and Revenue.

When implementing the Decision Tree model, similar steps had to be taken in comparison to the Linear Regression model. The model was fitted to the training data. The accuracy yielded better results as shown below:

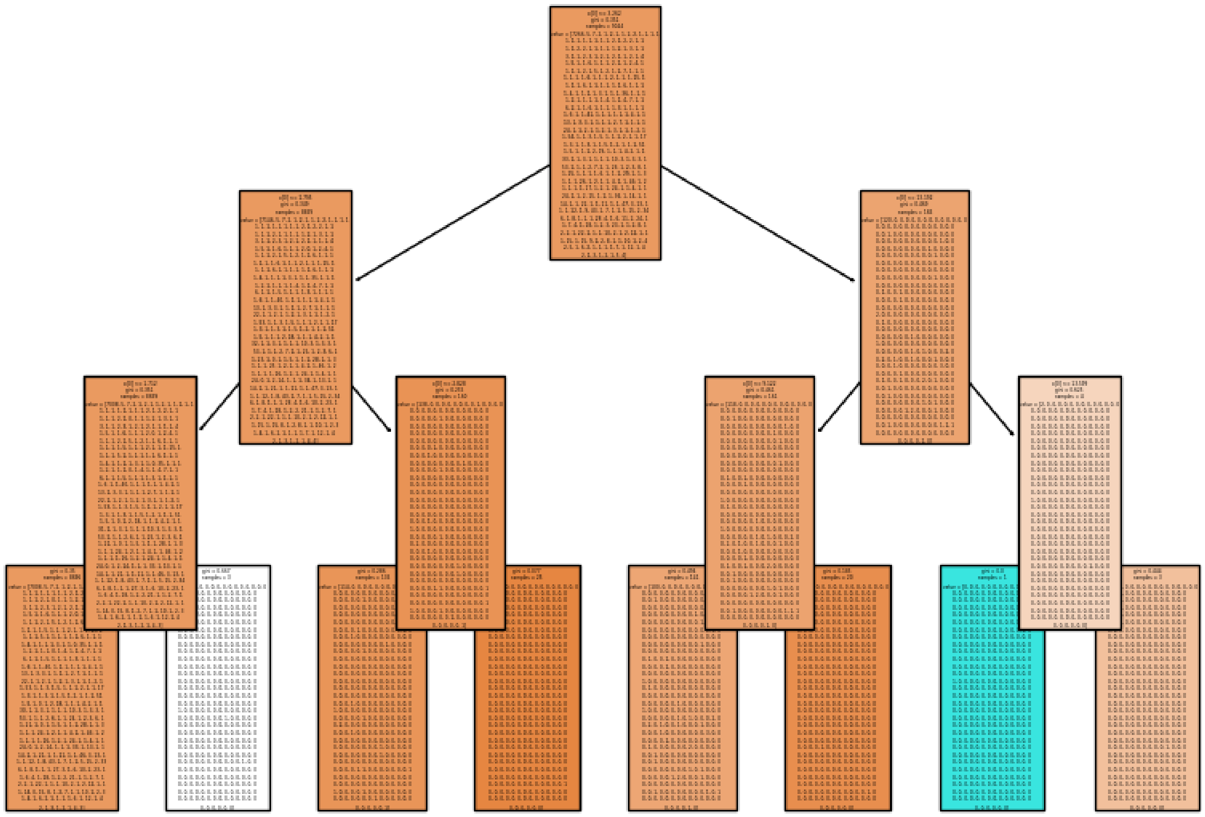


```
[163] learning(X_train, y_train, X_test, y_test, DT)

      Training accuracy: 0.805
      Testing accuracy: 0.836
      /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have
        warnings.warn(
      ▾ DecisionTreeClassifier
      DecisionTreeClassifier(max_depth=3)
```



```
[166] learning(X_train_std, y_train, X_test_std, y_test, DT)

      Training accuracy: 0.805
      Testing accuracy: 0.836
      ▾ DecisionTreeClassifier
      DecisionTreeClassifier(max_depth=3)
```

This would suggest that the Decision Tree model was a better model than the Linear Regression model for this table. The visualization of the decision tree is shown below:



The results described in this analysis have shown that the dataset chosen was not good enough to implement these models. It was a struggle to implement the Linear Regression model due to the number of errors in the code. The code itself is not the issue, but the data being used made working with the model more difficult than usual. The Decision Tree model yielded better results. The changes that could be made to improve the overall performance of each model would be to utilize more than one feature in the X variable (even though it was tried before and it did not work out because of the dataset). Overall, between the two models, the decision tree model performed the best due to the accuracy score being quite a bit higher than that of the linear regression model. Another way to say this is that the decision tree model is the best because the training and testing accuracies are both above 0.8. Either the dataset needs to change, or the approach at coding used for the dataset needs to change.

References: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset/data