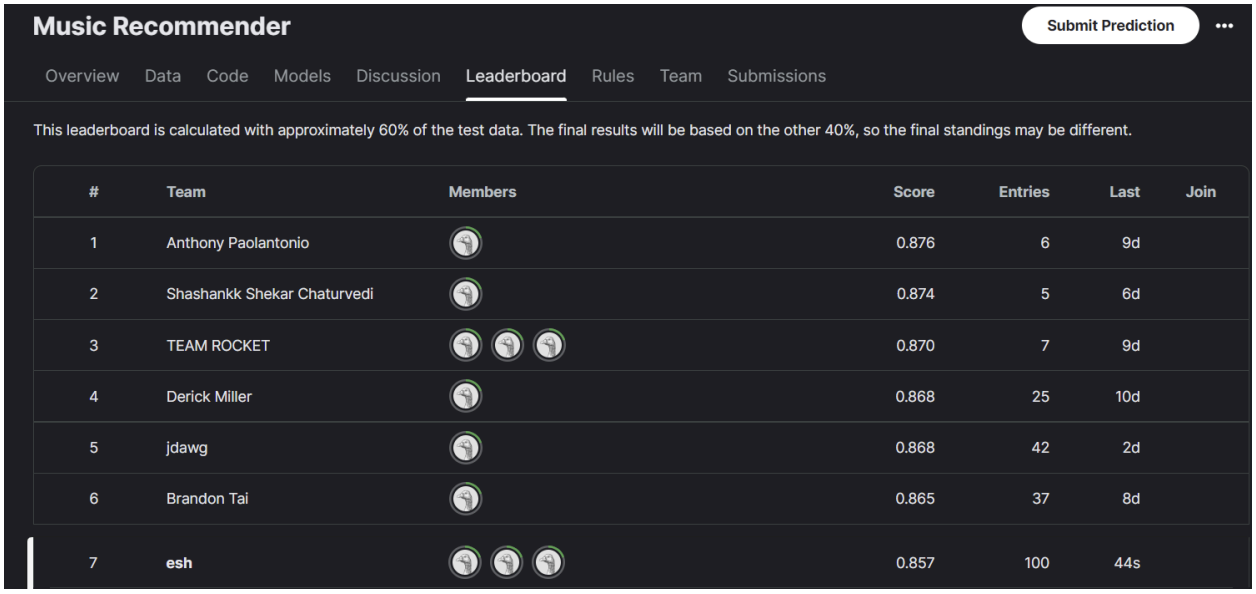
Team Esh: Chris Muro, Andrew Greensweight, Marc DiGeronimo




Current Best Score = .857 Total Successful Submissions = 100

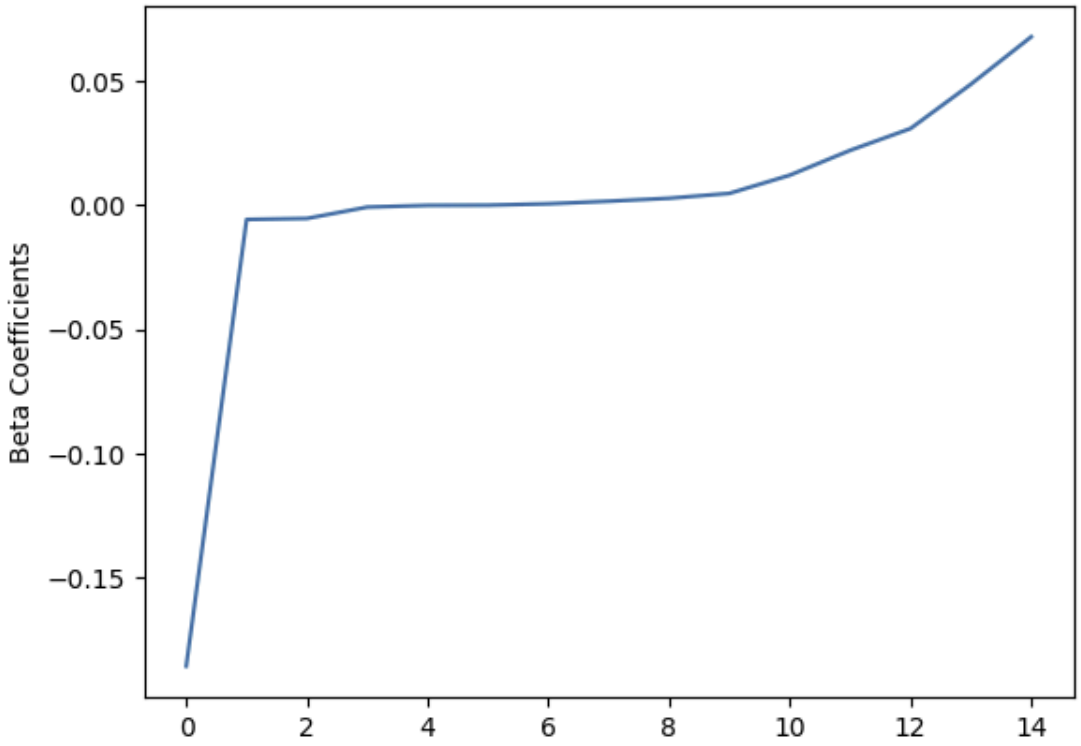
For HW9 the test2_new.txt file provides the ground truth trackID recommendations for 6000 entries which are used as a training set for the various models that were constructed in this assignment. First the environment is configured in google colab by downloading and importing the necessary packages. The txt file test2_new.txt was converted into a dataframe called ratings_df because it contains the ground truth for the 6000 entries provided. The default column names were 0 1 and 2 and were then renamed UserID TrackID and Rating as they are more descriptive and consistent with previous labels. Then a file was read that contained the previous predictions from HW5 and HW6 but slightly different. This file is not in the kaggle specified format, in fact it is a more detailed format that contains all of the calculated measurements used to get the rating. Test_df is the dataframe containing this data but it is the full 120000 entries which will serve as the test data. To get the train_df the entries that overlap between test2_new and test_df are merged so it contains all the information from the test set but the correct ratings from the true ratings. It is broken down into sub_train_df which drops the userID and TrackID

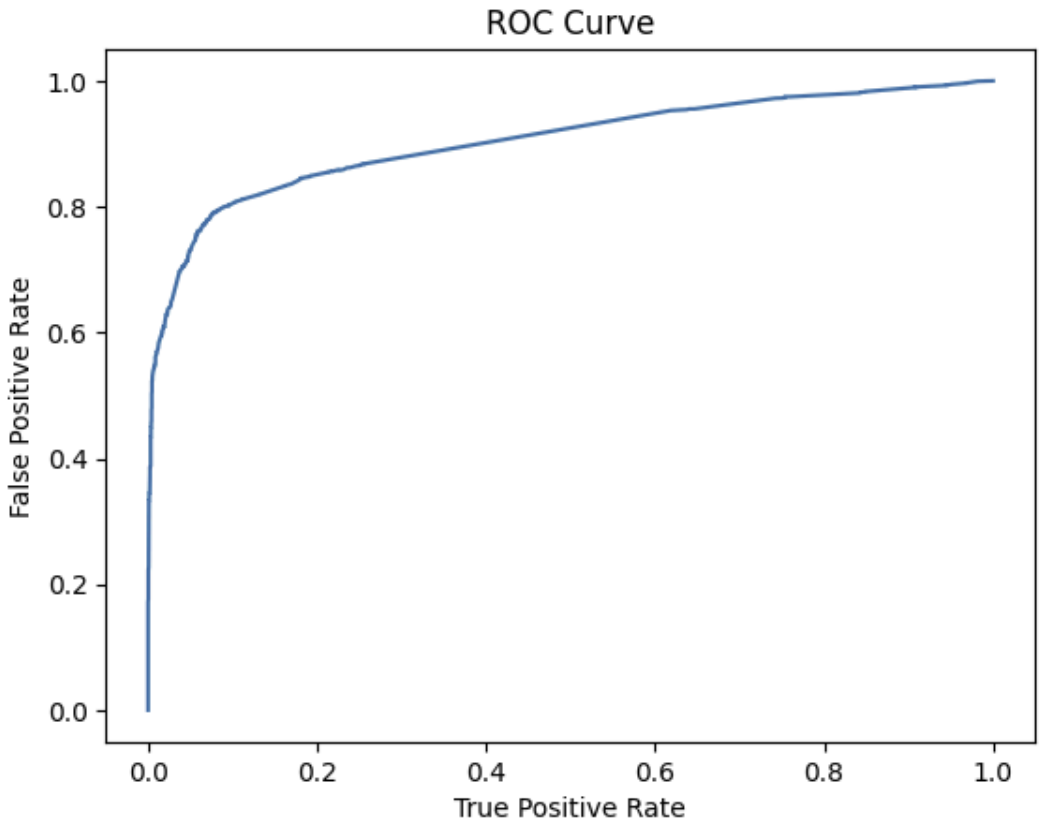
```
|AlbumRating|ArtistRating|Genre1Rating|Genre2Rating|Genre3Rating|Genre4Rating|Genre5Rating|Genre6Rating|Genre7Rating|NumberRa
tedGenres|MaxGenreScore|MinGenreScore|SumGenreScores| AverageGenreScore|VarianceGenreScore|Rating|
```

Here all of the measurements can be used to train the model as features and the rating column will serve as labels. Also there is sub_test_df which is the full 120000 entries but drop the rating column as that will be the job of the model to produce. Next the features need to be added to a vector assembler object before training. The features used are

```
features = ["AlbumRating", "ArtistRating", "Genre1Rating", "Genre2Rating",
"Genre3Rating", "Genre4Rating", "Genre5Rating", "Genre6Rating", "Genre7Rating",
"NumberRatedGenres", "MaxGenreScore", "MinGenreScore", "SumGenreScores",
"AverageGenreScore", "VarianceGenreScore" ]
```

This is passed to a vector assembler and the assembler is used to transform the sub_train_df and sub_test_df to add a feature column with all of the features combined into a feature vector for each entry. In this assignment the data did not need to be encoded or other string preprocessing because all of the feature data is numeric. Now the data is prepared for training. The first model used is logistic regression. Here a LogisticRegression model is used with the features column selected as the features vector generated by the vector assembler and label column as rating. The default value for maxIter is 10. The model is fit to the training data sub_train_df. The beta coefficients and ROC curve are plotted. Here is the result for the default model.




The trained model is then used to generate the predictions for the test data sub_test_df.

In attempts to optimize the model the parameters maxIter, regParam, aggregationDepth, standardization, elasticNetParam, family and fitIntercept were modified to see how the score was effected. Most did not change the effect or made it worse. The default values seemed to be the best. Increasing maxIter helps to a point but then plateaus around 10, Increasing regParam above 0 results in a worse score. The other default values were also better than manually modifying. The next model used is the decision tree classifier. Here a DecsionTreeClassifier is used with the features column set to features, labels set to Rating and maxDepth default to 3. The model is trained on sub_train_df and makes predictions on sub_test_df. The modifications to optimize were the parameters depth, impurity, maxBins, minInstancesPerNode and minInfoGain. The deeper the tree typically the worse the performance. Depths at 2 and 3 were high with depth of 1 having poor performance. Changing impurity does not affect results as well as maxBins and minInstancesPerNode. Anything above minInfoGain default of 0 resulted in worse performance. The next model is Random Forrest Classifier. Here a RandomForestClassifier object is used with the features column set to features, labels set to Rating. The model is trained on sub_train_df and makes predictions on sub_test_df. The parameters modified were maxDepth and numTrees. Depth followed the same patterns as DecisionTreeClassifier numTrees default of 20 was changed to 50 which resulted in a slight increase in performance. Changing it to 100 did not result in any additional changes.
The last model used in this assignment is Gradient Boosted Tree Classifier. Here a GBTClassifier is used with the features column set to features, labels set to Rating and maxIter default is set to 10. The model is trained on sub_train_df and makes predictions on sub_test_df. MaxIter was increased and performance improved very slightly. Unfortunately modifying the parameters did not lead to significant performance gains for any of the models, which is disappointing. Other methods are needed to improve the scores. Based on these 4 models, DecisionTreeClassifier probably has the most room to be optimized since it has the lowest AUC of all of the model predictions slightly.For each of the 4 models the process for converting the predictions to a format that is acceptable for kaggle submission is the same, with the only difference being the different file names for different models. The process is as follows.

```
predictions_pandas = predictions.toPandas()
kaggle_output = 'myprediction1_kaggle_lr.csv'
fOut_submission = open(kaggle_output, 'w')
csv_writer = csv.writer(fOut_submission)
header_submission = ["TrackID", "Predictor"]
csv_writer.writerow(header_submission)
for i in range(len(predictions_pandas)):

csv_writer.writerow([f"{predictions_pandas['UserID'][i]}_{predictions_pandas['T
rackID'][i]}", int(predictions_pandas["prediction"][i])])
fOut_submission.close()
```

This code comes from HW8, kaggle_output is the name of the file for the model predictions with the ending of the file describing which model the predictions came from (lr, dt, rf, gbt) the file is opened and csv writer object is initialized. Header_submission is the format for kaggle. For the total length of the predictions df which is 120000 entries the relevant components are written to

the csv file and the file is closed. The files were then submitted to kaggle automatically by configuring colab environment and submitting using terminal commands for each model.

```
kaggle competitions submit -c aai627-spring2024 -f myprediction1_kaggle_lr.csv -m "Logistic Regression Predictions
```