Dante Bajarias and Suvass Ravala

Professor Yi Fang

09 December 2022

# COEN 140 Project: Music Genre Classifier

## INTRODUCTION & CONTRIBUTIONS:

Music is a universal aspect of cultures and societies all over the world, which inherently diversifies the amount of genres that express songs. As eclectic music enjoyers, we love listening to many different types of music and correctly identifying the genre of a track helps us find similar tracks or genres. Additionally, accurately identifying genre helps establish clearer boundaries for what constitutes a song to be that specific genre.

The coding work was distributed evenly with Dante contributing to the training of the data and implementing the metrics plus their calculations, while Suvass worked on setting up the classifiers and visualizing the data on graphs. The report was split into two halves with work being done respectively.

## METHODOLOGY:

Before we applied any machine learning techniques, we did an 80/20 training testing split on 8000 tracks with our x-value being Mel Frequency Cepstral Coefficient (MFCC) feature data and our y-value being the track's genre, then finally shuffling the training samples. To pre-process our data we utilized Python's sklearn.preprocessing library to get the StandardScaler function which standardizes our features by removing the mean and scaling to unit variance. Essentially,

we calculated z-score: $z = (x - u)/s$, where u is the mean, s is the standard deviation, and x is our sample. Eleven classifier models were fit onto our data: logistic regression, k-nearest neighbors, support vector classification (radial basis function, polynomial, and linear), decision tree, random forest, Gaussian Naive Bayes, quadratic discriminant analysis, and linear discriminant analysis. SVC RBF and linear models were modified to use a regularization parameter, c = 1000, to become intolerant to incorrect classifications. Decision tree is a model that identifies possible options and weighs the risks and benefits against each option that can be yielded. Random Forest is a model that uses decision trees to generate an output by getting the prediction from each tree then averaging those predictions to produce a single output. The other models were discussed in class and their equations will be displayed in the results section. We wanted to use a multitude of classifiers to understand the underlying data distribution of our model and make assumptions about the model.

## EXPERIMENTS:

We only tested against one feature called Mel Frequency Cepstral Coefficient (MFCC), coefficients that compose a short-term power spectrum of a sound. There were 16 categories in our dataset corresponding to the 16 top level genres we classified the data into.

## EVALUATION METRICS:

Four main evaluation metrics were used: accuracy, precision, recall, and F1 score. Accuracy calculates the positive classifications of our model or matched genres:

$Accuracy = (Correct\ predictions)/(Total\ predictions)$. This alone does not show the entire picture since our data set is imbalanced which can cause issues for when a track is not classified. The other three metrics are used to help improve the evaluation of our imbalanced data set. Precision measures the true positives or which genre is actually correctly matched: $Precision = (True\ Positive)/(True\ Positive + False\ Positive)$. Recall measures the percentage of positives well predicted by our model or minimizes incorrectly matched genres: $Recall = (True\ Positive)/(True\ Positive + False\ Negative)$. These two metrics alone are also insufficient at properly evaluating our model. However, F1 is a combination of recall and precision that calculates a harmonic mean or the average of percentage which is a good evaluation of the performance of our model:

$F1\ Score = 2 \times (Recall \times Precision)/(Recall + Precision)$. We ended up using the F1 score as the metric that was most valuable in determining the effectiveness of each model, since a valid approach to comparing models is through their precision and recall scores which F1 score accounts for in its score calculation.

# RESULTS:

**Logistic Regression (LR):** $P(y_i = 0|x_i) = \frac{e^{-w^T x_i}}{1+e^{-w^T x_i}}$

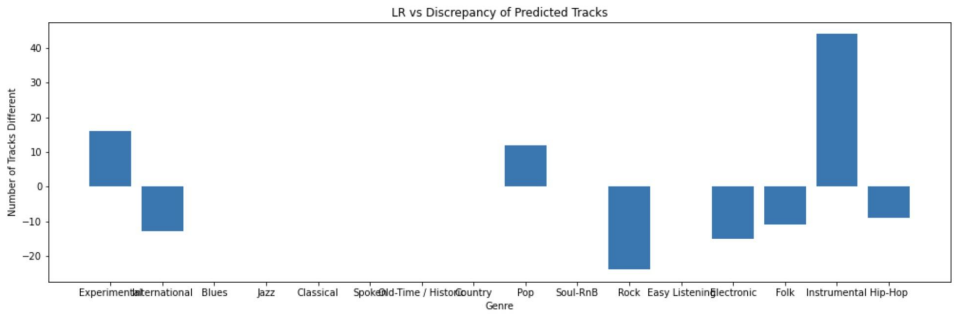
LR :

Accuracy: 44.94%

F1 Score:
44.52607031577631

Recall Score:
44.9375

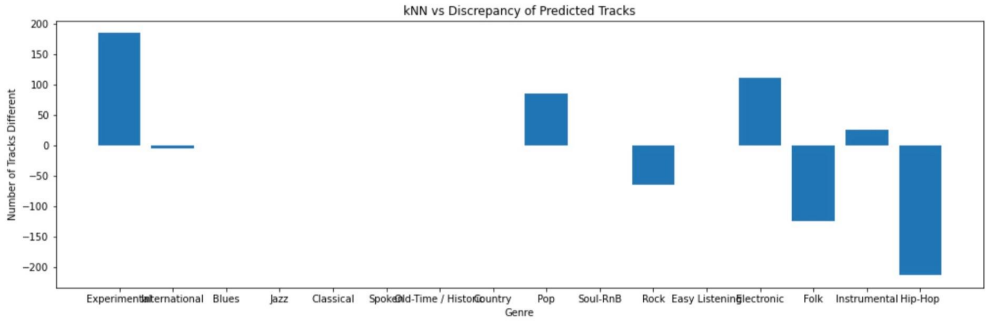
Precision Score:
44.37767385706559




Average Discrepancy: 9.0

**K-Nearest Neighbors (KNN):** $\sum_{i=1}^{n} = ||x_i - u_{c^{(i)}}||^2$

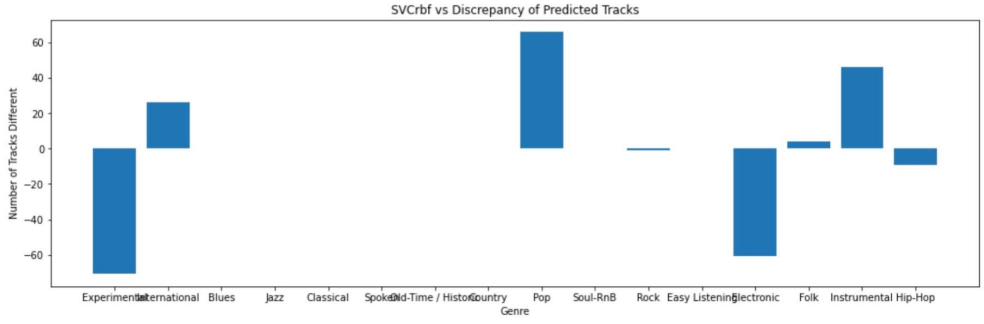
```
kNN :

Accuracy: 40.69%

F1 Score:
36.748243449716405

Recall Score:
40.6875

Precision Score:
42.22590766117166
```




```
Average Discrepancy: 51.0
```

**Support vector classifier using a radial basis function (SVCrbf):**

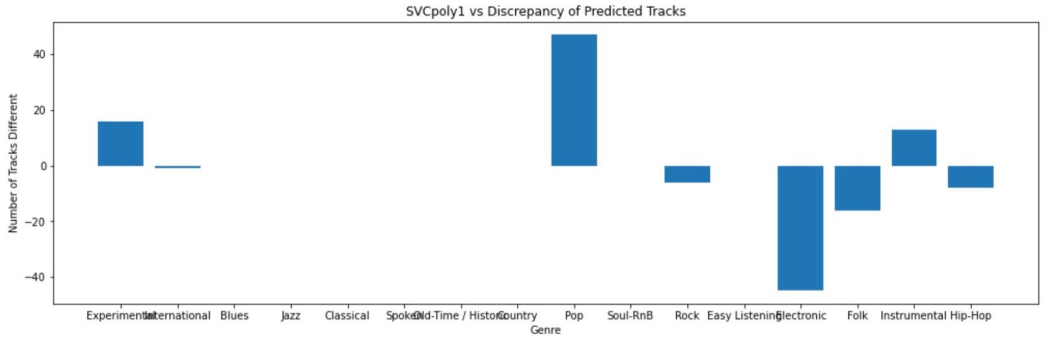
```
SVCrbf :

Accuracy: 49.75%

F1 Score:
49.26936035269999

Recall Score:
49.75000000000001

Precision Score:
49.764432931763395
```




```
Average Discrepancy: 17.75
```

## Support vector classifier using a polynomial function with degree 1 (SVCpoly1):

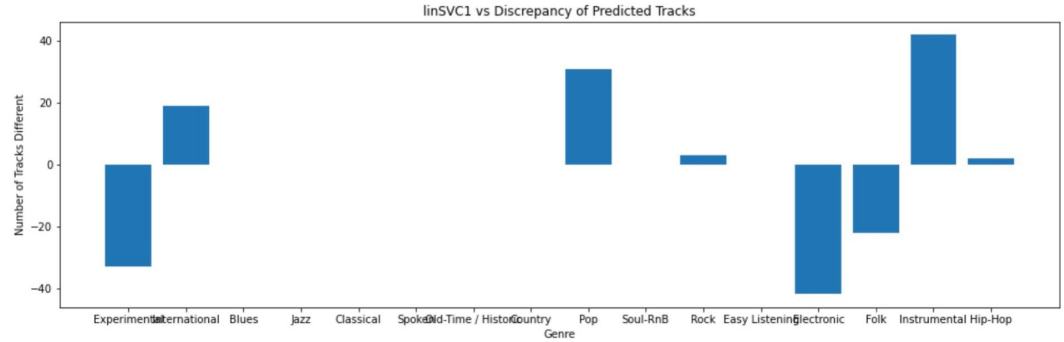
```
Average Discrepancy: 17.75
SVCpoly1 :

Accuracy: 45.19%


F1 Score:
44.64439541571211


Recall Score:
45.1875


Precision Score:
44.39464062387268
```




```
Average Discrepancy: 9.5
```

## Linear support vector classifier (linSVC1):

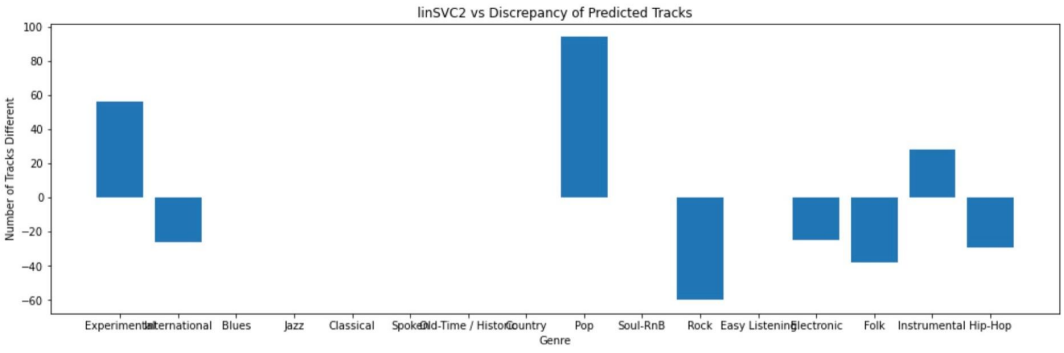
```
linSVC1 :

Accuracy: 43.88%


F1 Score:
43.65862641897108


Recall Score:
43.875


Precision Score:
43.839013409238156
```




```
Average Discrepancy: 12.125
```

## Linear support vector classifier with penalty parameter, c = 1000 (linSVC2):

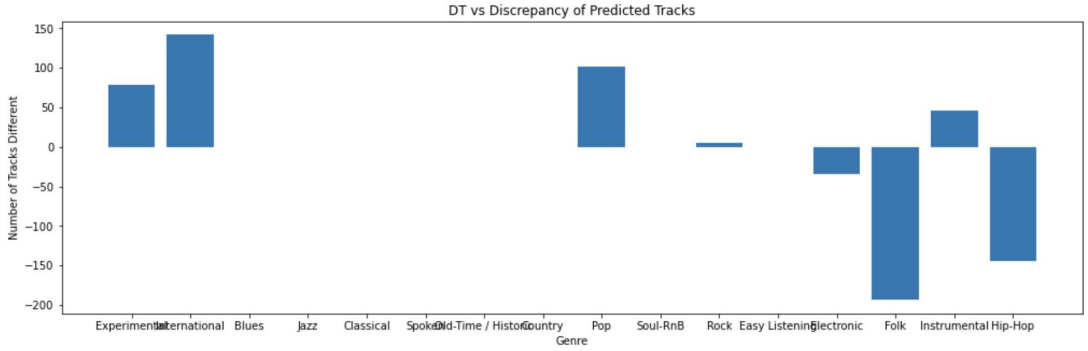
```
linSVC2 :

Accuracy: 46.12%


F1 Score:
44.66663396993995


Recall Score:
46.125


Precision Score:
44.5318434019474
```




```
Average Discrepancy: 22.25
```

## Decision Tree (DT):

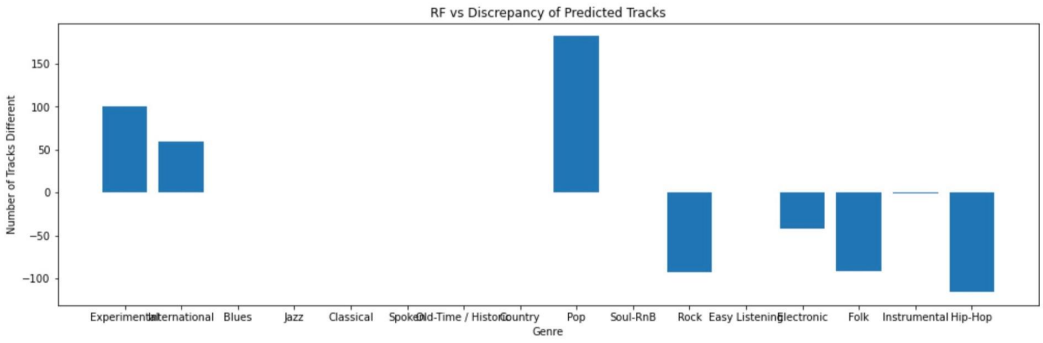
```
DT :

Accuracy: 31.00%


F1 Score:
28.589188581644702


Recall Score:
31.000000000000007


Precision Score:
29.88697337625294
```




```
Average Discrepancy: 46.625
```

## Random Forest (RF):

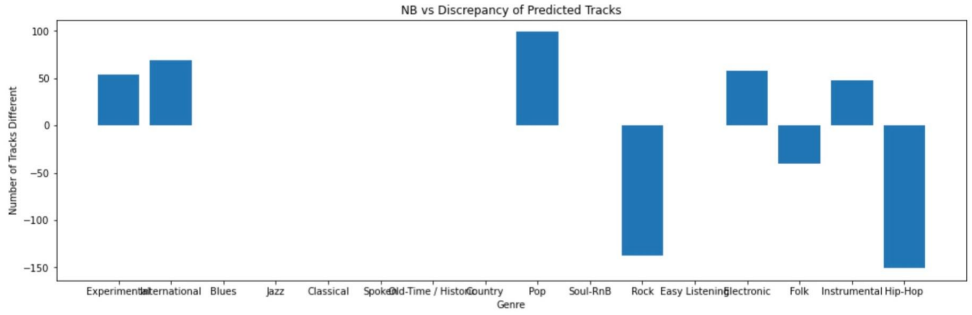
```
RF :

Accuracy: 38.12%


F1 Score:
34.76964963859797


Recall Score:
38.125


Precision Score:
35.090133000781634
```




```
Average Discrepancy: 42.75
```

## Gaussian Naive Bayes (NB):

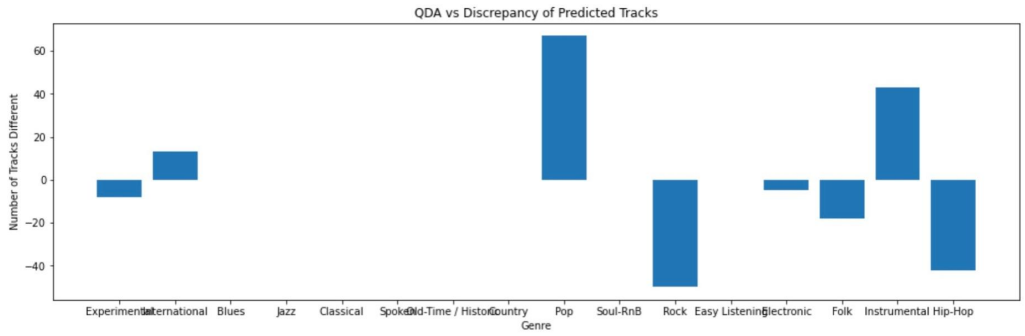
```
NB :

Accuracy: 40.75%


F1 Score:
38.37129434415334


Recall Score:
40.75


Precision Score:
39.575232826514736
```




```
Average Discrepancy: 41.0
```

## Quadratic Discriminant Analysis (QDA):

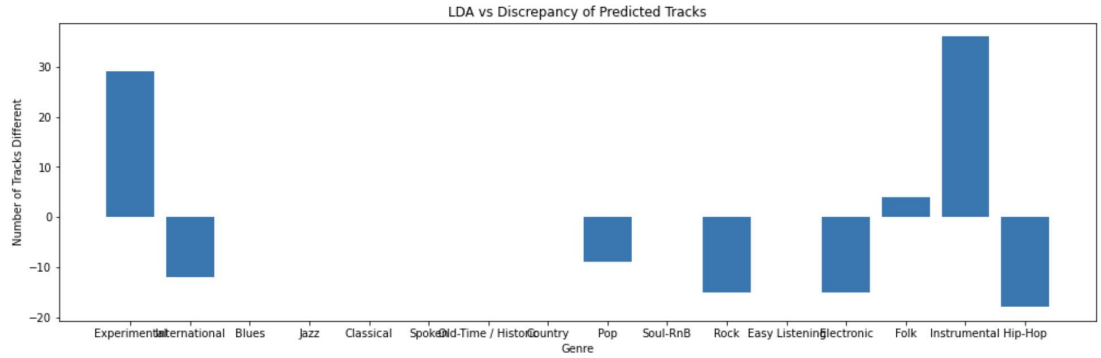
```
QDA :

Accuracy: 42.44%


F1 Score:
41.55128939910061


Recall Score:
42.43749999999999


Precision Score:
41.30759666286071
```




```
Average Discrepancy: 15.375
```

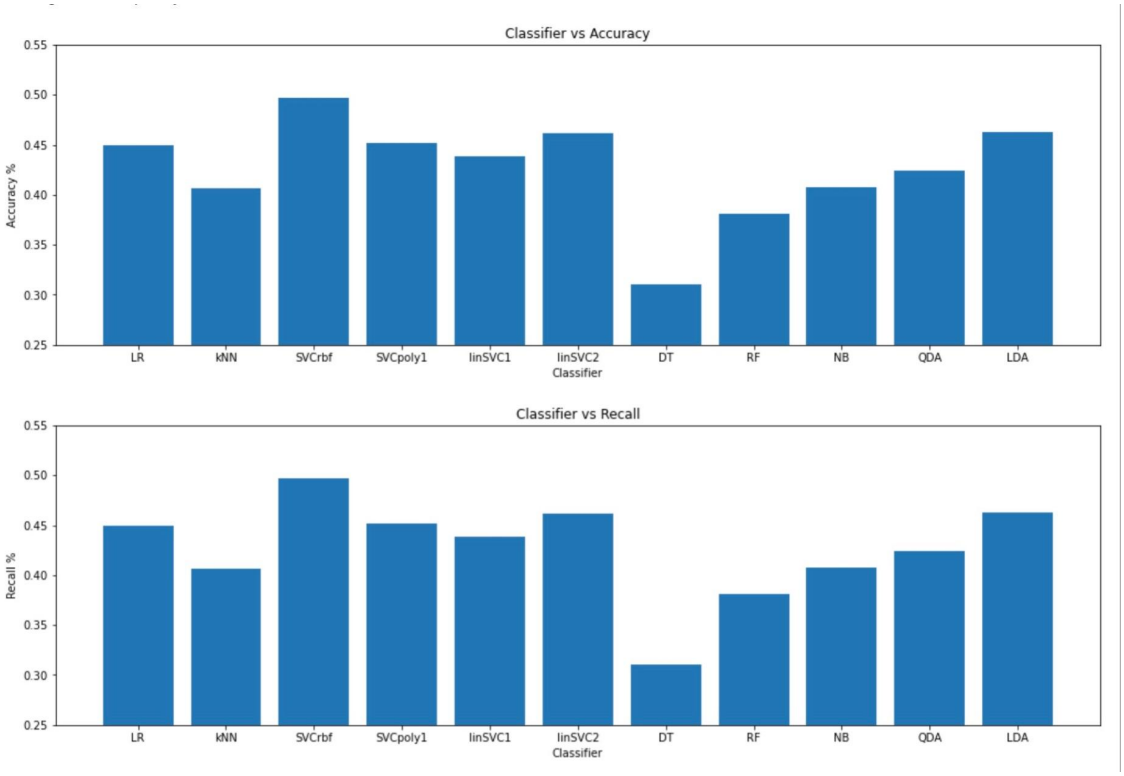
## Linear Discriminant Analysis (LDA):

```
LDA :

Accuracy: 46.31%


F1 Score:
46.06287218429701


Recall Score:
46.3125


Precision Score:
46.045888192803844
```



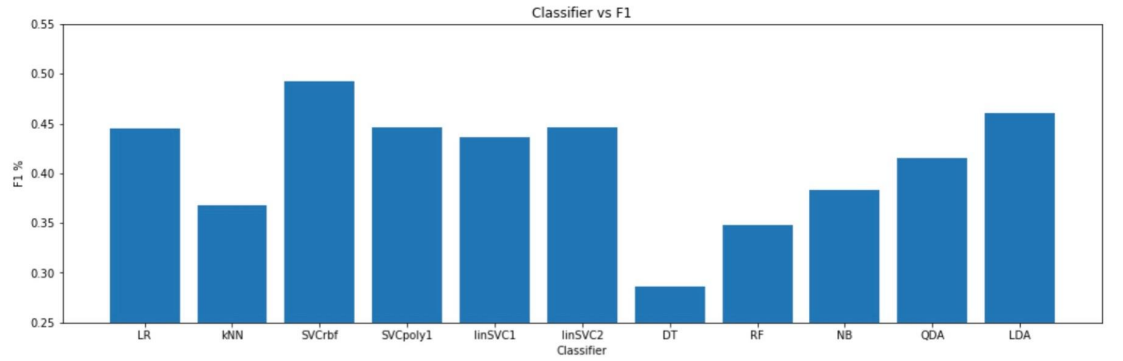

```
Average Discrepancy: 8.625
```

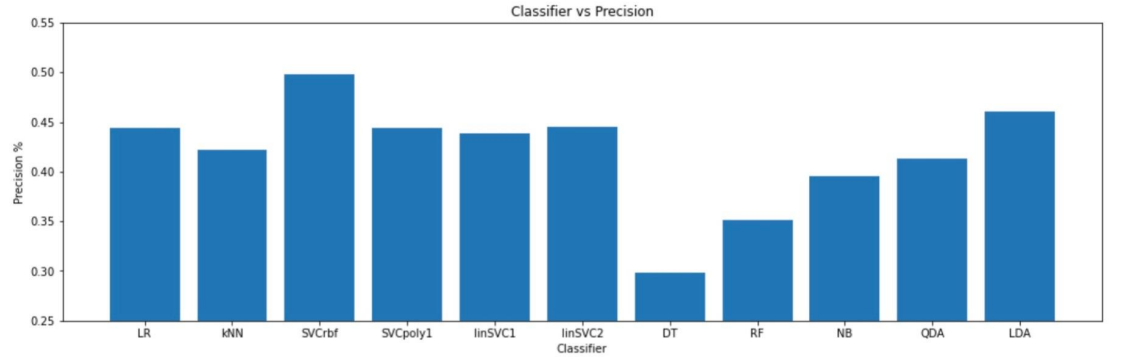
**Results of the classifiers using different metrics:**









```
X_train, y_train = skl.utils.shuffle(X_train, y_train, random_state=42)

# Standardize features by removing the mean and scaling to unit variance.
scaler = skl.preprocessing.StandardScaler(copy=False)
scaler.fit_transform(X_train)
scaler.transform(X_test)

y_test = np.array(y_test)
y_test = y_test.T
print(X_test.shape, y_test.shape)
#classification models
classifiers = {
    'LR': LogisticRegression(max_iter=10000),
    'kNN': KNeighborsClassifier(n_neighbors=200),
    'SVCrbf': SVC(kernel='rbf'),
    'SVCpoly1': SVC(kernel='poly', degree=1),
    'linSVC1': SVC(kernel="linear"),
    'linSVC2': LinearSVC(),
    'DT': DecisionTreeClassifier(max_depth=5),
    'RF': RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    'NB': GaussianNB(),
    'QDA': QuadraticDiscriminantAnalysis(),
    'LDA': LinearDiscriminantAnalysis()
 }
classifier_list=[]
accuracy_list=[]
prec_list=[]
rec_list = []
f1_list=[]
genre_list_test = genres.loc[genres['top_level'].unique()].title
genre_list =[]
y_test_categories = []
for genre in genre_list_test:
    genre_list.append(genre)
y_test_categories = Counter(y_test)
```

```
for key in classifiers:
        y_pred_categories = []
        genre_val = []
        avg = 0
        count = 0
        classifier = classifiers[key]
        classifier.fit(X_train, y_train)
        pred = classifier.predict(X_test)
        y_pred_categories = Counter(pred)
        for genre in genre_list:
            genre_val.append(y_test_categories[genre]- y_pred_categories[genre])
            avg += abs(genre_val[count])
            count+=1
        avg = avg /len(genre_val)
        f1 = f1_score(y_test, pred, average = None)
        f1 = sum(f1)/len(f1)
        score = classifier.score(X_test, y_test)
        prec_score = precision_score(y_test, pred, average = None)
        prec_score = sum(prec_score)/len(prec_score)
        rec = recall_score(y_test, pred, average = None)
        rec = sum(rec)/len(rec)
        print(key + " :\n")
        print('Accuracy: {:.2%}'.format(score))
        print("\n")
        print("F1 Score: ")
        print(f1*100)
        print("\n")
        print("Recall Score: ")
        print(rec*100)
        print("\n")
        print("Precision Score: ")
        print(prec_score*100)
        print("\n")
        classifier_list.append(key)
        accuracy_list.append(score)
        prec_list.append(prec_score)
        rec_list.append(rec)
        f1_list.append(f1)

        plt.bar(genre_list, genre_val)
        plt.title(f'{key} vs Discrepancy of Predicted Tracks')
        plt.xlabel('Genre')
        plt.ylabel('Number of Tracks Different')
        plt.show()

        print("Average Discrepancy:", avg)
```

```
plt.bar(classifier_list, accuracy_list)
plt.title('Classifier vs Accuracy')
plt.xlabel('Classifier')
plt.ylabel('Accuracy %')
plt.ylim([0.25, 0.55])
plt.show()

plt.bar(classifier_list, rec_list)
plt.title('Classifier vs Recall')
plt.xlabel('Classifier')
plt.ylabel('Recall %')
plt.ylim([0.25, 0.55])
plt.show()

plt.bar(classifier_list, f1_list)
plt.title('Classifier vs F1')
plt.xlabel('Classifier')
plt.ylabel('F1 %')
plt.ylim([0.25, 0.55])
plt.show()

plt.bar(classifier_list, prec_list)
plt.title('Classifier vs Precision')
plt.xlabel('Classifier')
plt.ylabel('Precision %')
plt.ylim([0.25, 0.55])
plt.show()
```

## ANALYSIS:

After we ran all of our different tests and metrics on our data we discovered multiple things. The first thing was that we realized that we had very low percentages across our data metrics. For most of the classifiers, we got the scores to be within 25% to 50% which is very low. However, we attribute it to the low amount of data that we trained and tested on as we only used 8000 data points, instead of the full 100K data points in the dataset. This is a big issue as we had 16 genres to classify the data points as, so there would be a high rate of error.

In terms of the best classifier, we were able to identify that SVCrbf was the best performing classifier. This is because, in the bar graphs above, it consistently ranked first in accuracy, f1 score, recall and precision. On the other end, we found out that the Decision Tree classifier ranked last among all four metrics as each of its metrics were around half of what the SVCrbf values were. Moreover, we realized that LDA and QDA did decently well on the dataset.

As a result, we were able to conclude several things about this dataset. The first was that since QDA and LDA were able to perform well, we concluded that the data is Gaussian. Furthermore, as a result of SVCrbf performing the best, we can conclude that the data is not linearly separable as it performs the best when the data is not linearly separable. In addition to that, the Decision Tree values that we were getting make sense as in general they perform worse than most ML classifiers and with the large amount of genres that we had, it would be tough for the decision tree to work well.

## CONCLUSION/FUTURE WORK:

In order to better our project one thing we could do is use the full 100K data set as it would better all the classifiers as it would have more data to train and test on. Furthermore, in addition to the 16 top level genres, we could also introduce a few sub level genres and attempt to classify tracks into multiple genres as that is more realistic to our current music culture. By doing the full genres, we would be able to work with the full 163 genres. Another thing we could do is work with multiple features instead of just “mfcc” to truly get the most accurate results. We could also end up using deep learning and neural networks to create a more accurate and complex machine learning model that can better classify the data points than we currently have.