TECHNOCOLABS DATA SCIENCE INTERNSHIP

FINAL PROJECT REPORT

Project: Classify Song Genres from Audio Data

Aim:

Using the dataset comprised of songs of two music genres (Hip-Hop and Rock) train a classifier to distinguish between the two genres based only on track information derived from Echo nest (now part of Spotify).

Abstract:

Over the past few years, streaming services with huge catalog have become the primary means through which most people listen to their favourite music. But at the same time, the sheer amount of music on offer can mean users might be a bit overwhelmed when trying to look for newer music that suits their tastes.

For this reason, streaming services have looked into means of categorizing music to allow for personalized recommendations. One method involves direct analysis of the raw audio information in a given song, scoring the raw data on a variety of metrics. I will be examining data compiled by a research group known as The Echo Nest. My Goal is to look through this dataset and classify songs as being either 'Hip-Hop' or 'Rock' - all without listening to a single one ourselves.

Overview of Tasks:

1. Preparing the Dataset
2. Pairwise relationships between continuous variables
3. Normalizing the feature data
4. Principal Component Analysis on scaled data
5. Further visualization of PCA
6. Train a decision tree to classify genre
7. Compare decision tree to a logistic regression
8. Balance out data for greater performance
9. Using cross-validation to evaluate models
10. Deploying to cloud

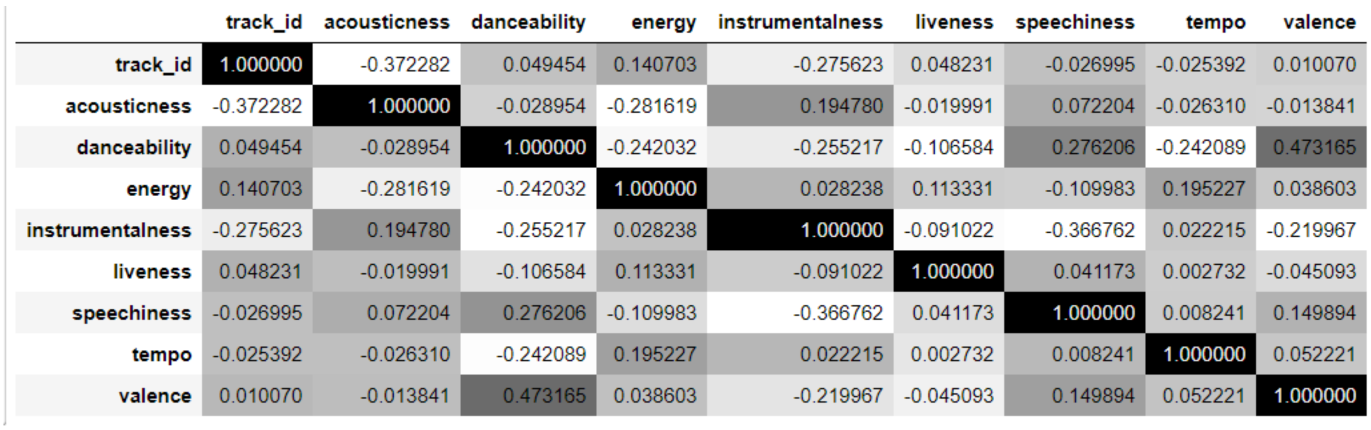
Preparing the Dataset:

The dataset was provided in two files, one CSV and one JSON file. The CSV file contained general information about the song including track_id, comments, composer, genres, licence and other details. The json file holds the technical details of a song like acousticness, energy, tempo etc. For this classification problem, I have considered all the technicalities of a song from JSON file and track_id and genres from CSV file. The final dataset has the following columns:

```
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   track_id          4802 non-null   int64  
 1   genre_top         4802 non-null   object 
 2   acousticness      4802 non-null   float64
 3   danceability      4802 non-null   float64
 4   energy            4802 non-null   float64
 5   instrumentalness  4802 non-null   float64
 6   liveness          4802 non-null   float64
 7   speechiness       4802 non-null   float64
 8   tempo             4802 non-null   float64
 9   valence           4802 non-null   float64
```

Pairwise relationships between continuous variables:

I want to avoid using variables that have strong correlations with each other, hence avoiding feature redundancy. Therefore, to visualize this, I used correlation matrix and here is the result of that:



	track_id	acousticness	danceability	energy	instrumentalness	liveness	speechiness	tempo	valence
track_id	1.000000	-0.372282	0.049454	0.140703	-0.275623	0.048231	-0.026995	-0.025392	0.010070
acousticness	-0.372282	1.000000	-0.028954	-0.281619	0.194780	-0.019991	0.072204	-0.026310	-0.013841
danceability	0.049454	-0.028954	1.000000	-0.242032	-0.255217	-0.106584	0.276206	-0.242089	0.473165
energy	0.140703	-0.281619	-0.242032	1.000000	0.028238	0.113331	-0.109983	0.195227	0.038603
instrumentalness	-0.275623	0.194780	-0.255217	0.028238	1.000000	-0.091022	-0.366762	0.022215	-0.219967
liveness	0.048231	-0.019991	-0.106584	0.113331	-0.091022	1.000000	0.041173	0.002732	-0.045093
speechiness	-0.026995	0.072204	0.276206	-0.109983	-0.366762	0.041173	1.000000	0.008241	0.149894
tempo	-0.025392	-0.026310	-0.242089	0.195227	0.022215	0.002732	0.008241	1.000000	0.052221
valence	0.010070	-0.013841	0.473165	0.038603	-0.219967	-0.045093	0.149894	0.052221	1.000000

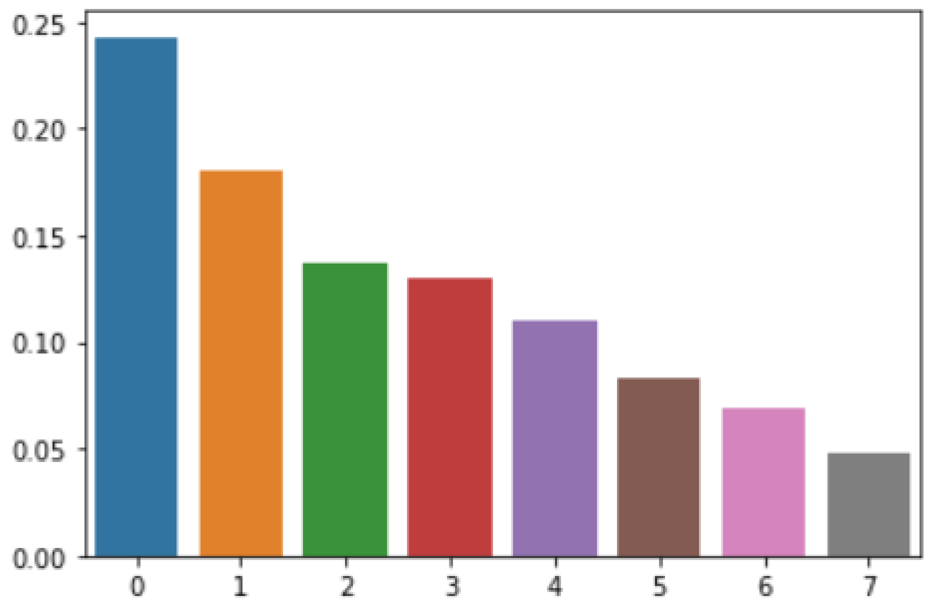
Dancebility and Valence are correlated maximum in this dataset.

Normalizing the feature data:

Since I didn't find any particular strong correlations between our features, I can use PCA (Principal Component Analysis) to reduce the number of features but PCA uses the absolute variance of a feature to rotate the data, a feature with a broader range of values will overpower and bias the algorithm relative to the other features. Therefore, the data was normalised using Standard Scaler method of sklearn pre-processing package.

Principal Component Analysis on our scaled data:

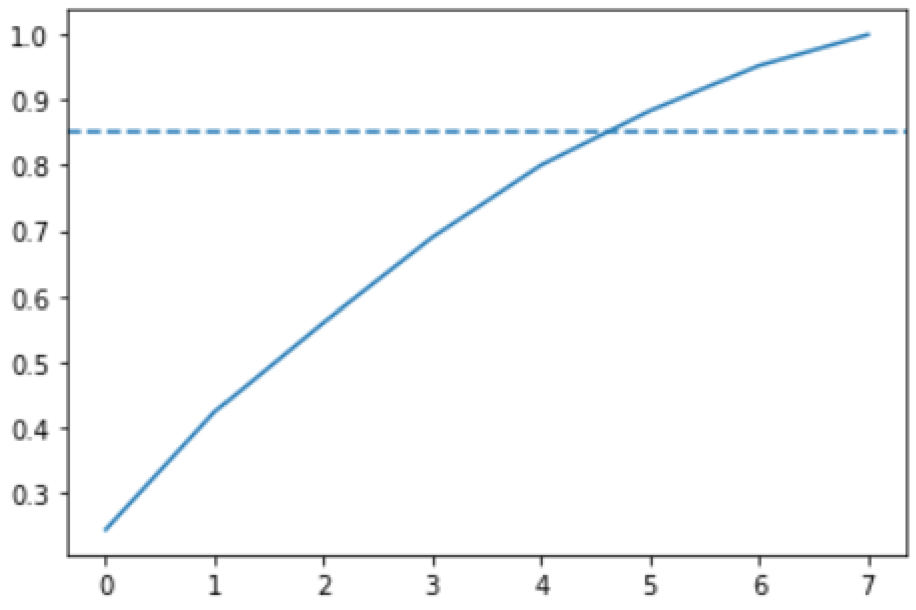
Using scree-plot, I need to find the number of components to use in further analyses. Scree-plots display the number of components against the variance explained by each component, sorted in descending order of variance. Scree-plots help get a better sense of which components explain a sufficient amount of variance in the data.



After looking at this I can’t choose number of features and therefore, further analysis is needed.

Further visualization of PCA:

By looking at cumulative explained variance plot at about 85% of the variance, I determined the appropriate number of components, 6, and then performed PCA with that many components, ideally reducing the dimensionality of data.



Train a decision tree to classify genre:

Now using this lower dimensional PCA projection of the data, I trained a decision tree to classify song genres.

```
X_train, X_test, y_train, y_test = train_test_split(pca_projected, y, stratify=y)
```

```
tree = DecisionTreeClassifier(random_state=42)
```

```
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(random_state=42)
```

```
y_pred = tree.predict(X_test)
```

```
accuracy_score(y_true=y_test, y_pred=y_pred)
```

```
0.855120732722731
```

```
confusion_matrix(y_true=y_test, y_pred=y_pred)
```

```
array([[150,  78],
       [ 96, 877]], dtype=int64)
```

```
print(classification_report(y_true=y_test, y_pred=y_pred))
```

```
              precision    recall  f1-score   support

     Hip-Hop       0.61      0.66      0.63       228
        Rock       0.92      0.90      0.91       973

    accuracy                           0.86      1201
   macro avg       0.76      0.78      0.77      1201
weighted avg       0.86      0.86      0.86      1201
```

Precision for Hip-Hop is low in this model which means that the model is biased towards Rock.

Comparison of decision tree to logistic regression:

Logistic regression makes use of what's called the logistic function to calculate the odds that a given data point belongs to a given class.

```
log_reg = LogisticRegression(random_state=42)
```

```
log_reg.fit(X_train, y_train)
```

```
LogisticRegression(random_state=42)
```

```
y_pred_reg = log_reg.predict(X_test)
```

```
accuracy_score(y_true=y_test, y_pred=y_pred_reg)
```

```
0.8884263114071607
```

```
confusion_matrix(y_true=y_test, y_pred=y_pred_reg)
```

```
array([[124, 104],
       [ 30, 943]], dtype=int64)
```

```
print(classification_report(y_true=y_test, y_pred=y_pred_reg))
```

```
              precision    recall  f1-score   support

     Hip-Hop       0.81      0.54      0.65       228
        Rock       0.90      0.97      0.93       973

    accuracy                           0.89      1201
   macro avg       0.85      0.76      0.79      1201
weighted avg       0.88      0.89      0.88      1201
```

Balance out data for greater performance:

Both the models did similarly well. However, looking at our classification report, one can see that rock songs are fairly well classified, but hip-hop songs are disproportionately misclassified as rock songs. This is because the rock genre has more samples than rock. To get a better performance, I balanced the Dataset and here is what I got:

```
Decision Tree:
              precision    recall  f1-score   support

     Hip-Hop       0.76      0.76      0.76       225
        Rock       0.76      0.76      0.76       225

    accuracy                           0.76       450
   macro avg       0.76      0.76      0.76       450
weighted avg       0.76      0.76      0.76       450

Logistic Regression:
              precision    recall  f1-score   support

     Hip-Hop       0.78      0.83      0.80       225
        Rock       0.81      0.76      0.79       225

    accuracy                           0.79       450
   macro avg       0.79      0.79      0.79       450
weighted avg       0.79      0.79      0.79       450
```

The results are promising in a balanced dataset.

Using cross-validation to evaluate models:

Now doing the last check on actual data to determine which model would give the best performance and model will be used for deployment purpose.

```
kf = KFold(n_splits=30, random_state=42, shuffle=True)
```

```
tree_kf = DecisionTreeClassifier(random_state=42)
log_reg_kf = LogisticRegression(random_state=42)
```

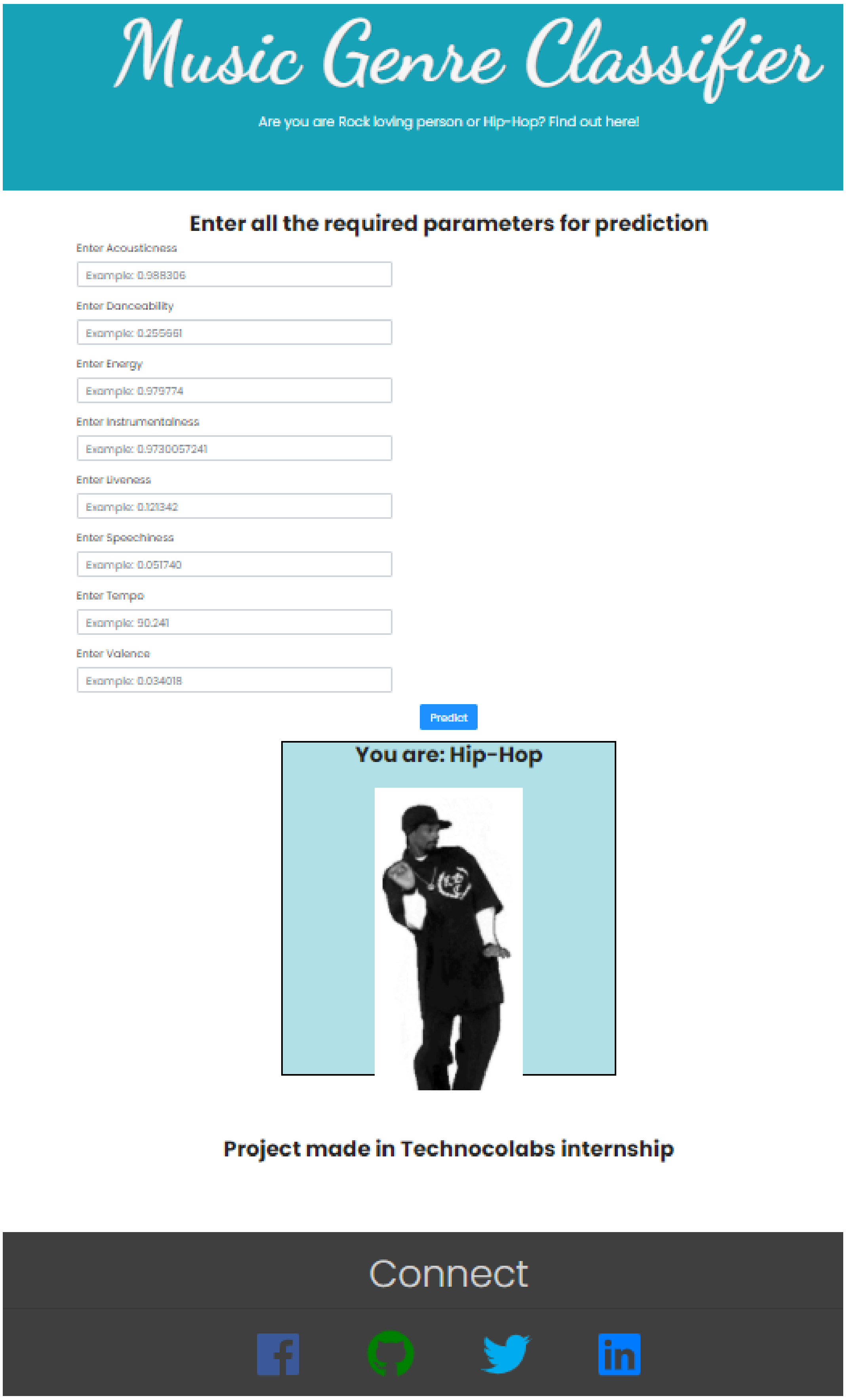
```
tree_score = cross_val_score(tree_kf, pca_projected, y, cv = kf)
log_reg_score = cross_val_score(log_reg_kf, pca_projected, y, cv = kf)

print("Decision Tree:", np.mean(tree_score), "Logistic Regression:", np.mean(log_reg_score))
```

```
Decision Tree: 0.8602562111801241 Logistic Regression: 0.8775478778467909
```

According to results, logistic regression proves to be better but I will go with decision tree as it doesn't require normalizing of data.

Deploying the Model:

After model building, the next step is model deployment and I choose Heroku. It is easy to deploy a model in Heroku and takes a few minutes to bootup the application. The final output of the web application is placed below:



Music Genre Classifier

Are you are Rock loving person or Hip-Hop? Find out here!

Enter all the required parameters for prediction

Enter Acousticness

Example: 0.988306

Enter Danceability

Example: 0.255661

Enter Energy

Example: 0.979774

Enter Instrumentalness

Example: 0.9730057241

Enter Liveness

Example: 0.121342

Enter Speechiness

Example: 0.051740

Enter Tempo

Example: 90.241

Enter Valence

Example: 0.034018

Predict

You are: Hip-Hop

Project made in Technocolabs internship

Connect