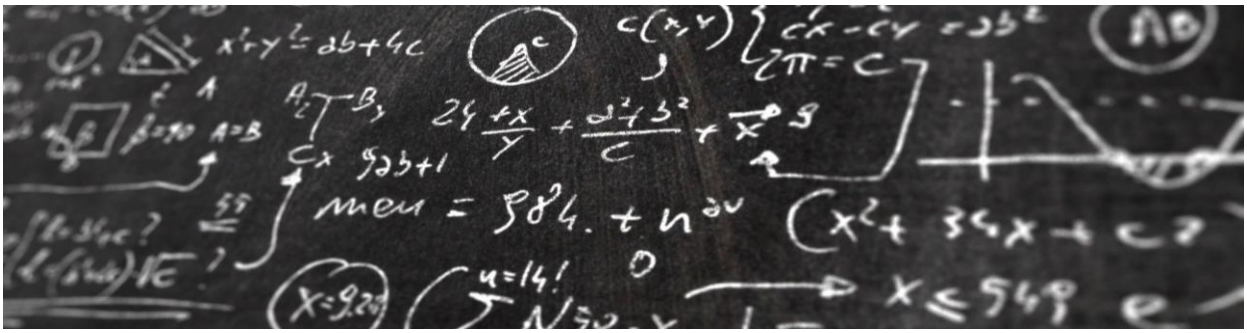
Northeastern University



INFO 6105 – Data Science Engineering Methods & Tools (Final Project)

Topic:

Spotify Music Prediction Algorithm

Team Name:

Team Refactor

Team Members:

1. Aditya Mulik
2. Tushar Kurhekar

Problem Statement:

To detect *popularity* of a track using the track's attributes.

Background:

Some key points which were performed for tuning the model for the dataset.

- Data analysis and Data Wrangling
- Cleaning and hyperparameter tuning
- Building the KNN Classification model
- Testing the model on train, validation, and data for accuracy
- Building the Confusion Matrix

Northeastern University

Dataset:

The dataset used for the project was Spotify tracks from the year 1921 – 2020 of around 600k track records (References[1])

Dataset Classification:

1. **Primary**
    a. id (id of track generated by Spotify)
2. **Numerical**
    a. speechiness (Ranges from 0 to 1)
    b. liveness (Ranges from 0 to 1)
    c. acousticness (Ranges from 0 to 1)
    d. danceability (Ranges from 0 to 1)
    e. energy (Ranges from 0 to 1)
    f. instrumentalness (Ranges from 0 to 1)
    g. valence (Ranges from 0 to 1)
    h. popularity (Ranges from 0 to 100)
    i. duration_ms (Ranges from 200k to 300k)
    j. tempo (Ranging from 50 to 150)
    k. loudness (Ranging from – 60 to 0)
3. **Dummy**
    a. mode (0 = Minor, 1 = Major)
    b. explicit (0 = No explicit content, 1 = Explicit content)
4. **Categorical**
    a. key
    b. timesignature
    c. artists
    d. release_date
    e. name

Northeastern University

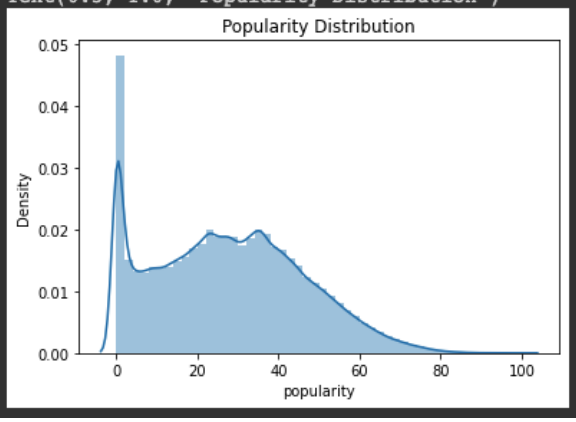
Algorithm Used:

KNeighBorsClassifier (References[2])

Key Proofs:

a. Data Distribution for Popularity & Accousticness

```
[24] sns.distplot(data['popularity']).set_title('Popularity Distribution')

/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619:
  warnings.warn(msg, FutureWarning)
Text(0.5, 1.0, 'Popularity Distribution')
```



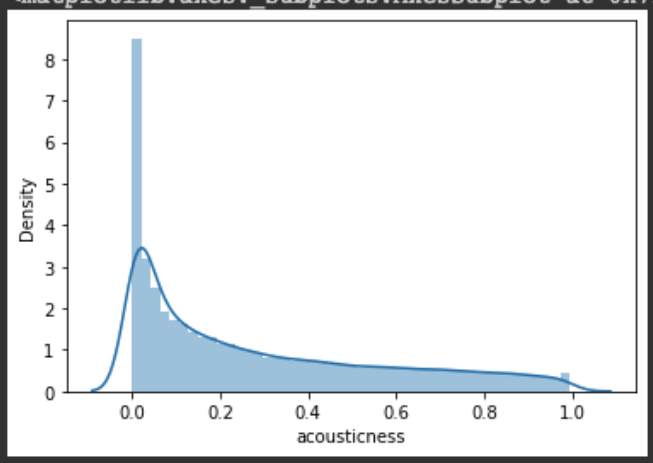
b. Hyperparameter Tuning: (Using RandomizedSearchCV)

```
Imputation using RandomizedSearchCV

[58] rand = RandomizedSearchCV(knn, param_dist, cv=5, scoring='accuracy', n_iter=5, random_state=5)

[59] rand.fit(X_train, y_train)

RandomizedSearchCV(cv=5, estimator=KNeighborsClassifier(n_neighbors=1),
                   n_iter=5, param_distributions={'n_neighbors': range(1, 22)},
                   random_state=5, scoring='accuracy')

[60] # examining the best model
print(rand.best_score_)
print(rand.best_params_)
print(rand.best_estimator_)

0.7287103089063802
{'n_neighbors': 3}
KNeighborsClassifier(n_neighbors=3)
```

Northeastern University

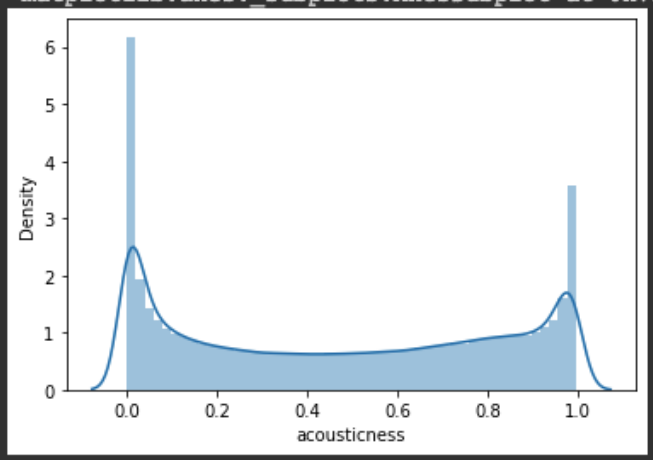
```
[26] popular_greater_50 = data[data.popularity > 50]
     sns.distplot(popular_greater_50['acousticness'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributic
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f850f85ca50>
```

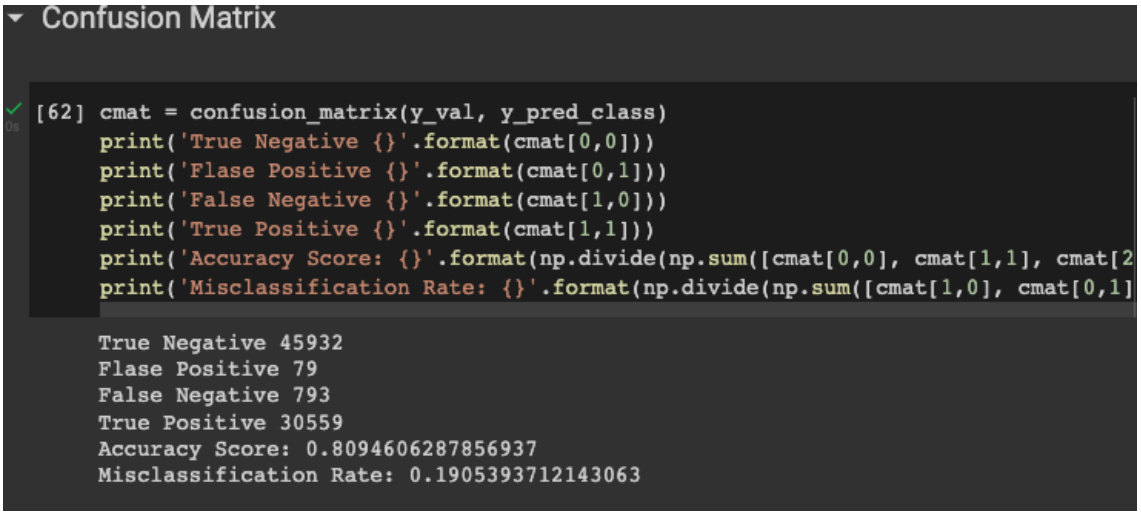



```
[27] popular_less_50 = data[data.popularity < 50]
     sns.distplot(popular_less_50['acousticness'])
```

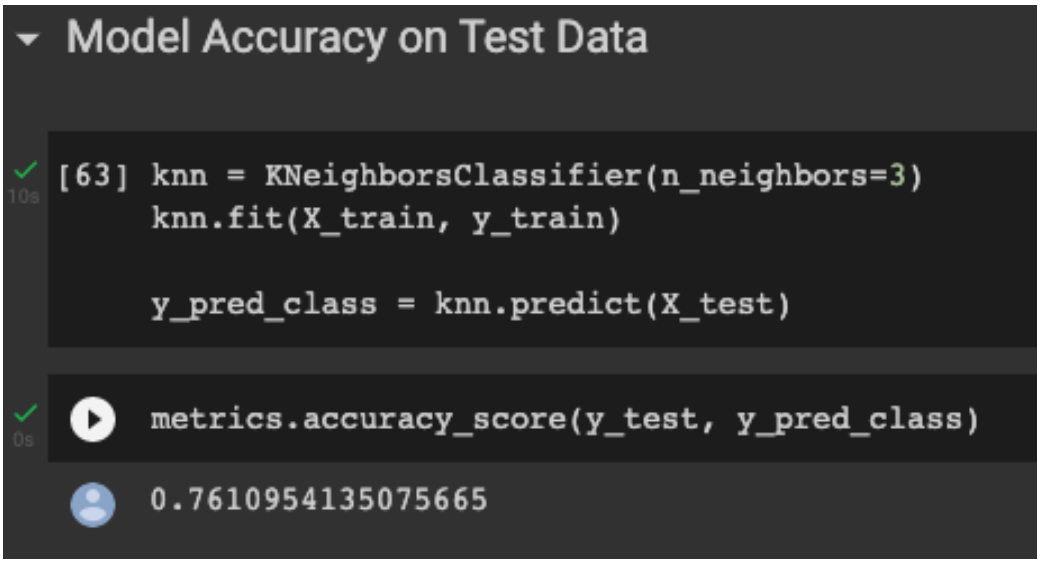
```
/usr/local/lib/python3.7/dist-packages/seaborn/distributic
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f850f73d190>
```



c. Confusion Matrix:



d. Model Accuracy Check for Test Data:



Results/ Conclusion:

The best accuracy score using RandomizedSearchCV to find the optimal n_neighbors, we get the best prediction & **accuracy score of 76%** which seems optimal and ensures no overfitting or underfitting of the model.

References:

1. https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks?select=tracks.csv
2. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html