

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

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

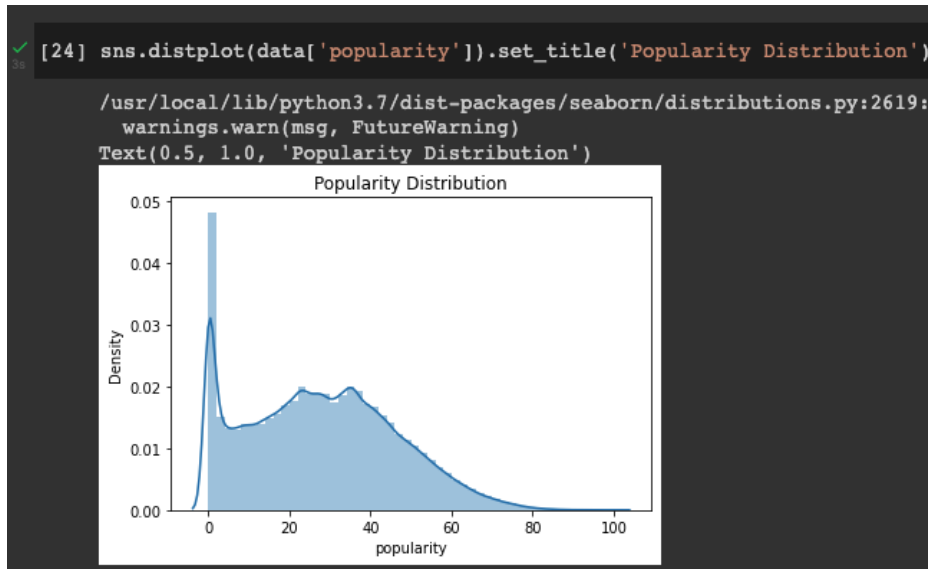
- a. key
- b. timesignature
- c. artists
- d. release_date
- e. name

Algorithm Used:

KNeighborsClassifier (References[2])

Key Proofs:

a. Data Distribution for Popularity



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)
```



c. Confusion Matrix:

```
▼ Confusion Matrix

[61] cmat = confusion_matrix(y_test, y_pred_class)
      print('True Negative {}'.format(cmat[0,0]))
      print('Flase Positive {}'.format(cmat[0,1]))
      print('False Negative {}'.format(cmat[1,0]))
      print('True Positive {}'.format(cmat[1,1]))
      print('Accuracy Score: {}'.format(np.divide(np.sum(
      print('Misclassification Rate: {}'.format(np.divide

True Negative 179699
Flase Positive 159
False Negative 4078
True Positive 114478
Accuracy Score: 0.7800762352370078
Misclassification Rate: 0.21992376476299227
```

d. Model Accuracy Check for Test Data:

```
▼ Model Accuracy on Test Data

[62] knn = KNeighborsClassifier(n_neighbors=3)
      knn.fit(X_train, y_train)

      y_pred_class = knn.predict(X_test)

[63] metrics.accuracy_score(y_test, y_pred_class)

0.7411054293242719
```

Results/ Conclusion:

The best accuracy score using RandomizedSearchCV to find the optimal n_neighbors, we get the best prediction & **accuracy score of 74%** which seems optimal and ensures no overfitting or underfitting.

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>