The accuracy metrics before data augmentation

```
Features,Labels = pca_features_labels.values[:,:-1], pca_features_labels.values[:,-1]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Features, Labels, test_size = 0.20, random_state = 42)
classifier = KNeighborsClassifier(n_neighbors = 2, algorithm ='ball_tree', metric = 'minkowski', p = 5)
classifier.fit(X_train, y_train)
ypred=classifier.predict(X_train)
```

Train Accuracy

```
result = confusion_matrix(y_train, ypred)
print('Confusion Matrix:')
print(result)
result1 = classification_report(y_train, ypred)
print('Classification Report:')
print (result1)
result2 = accuracy_score(y_train,ypred)
print('Accuracy:',result2)
```

```
Confusion Matrix:
[[85  0  0  0  0  0  0  0  0  0]
 [ 1 75  0  0  0  0  0  0  0  0]
 [ 6  3 75  0  0  0  0  0  0  0]
 [11  0 11 59  0  0  0  0  0  0]
 [ 4  0  2  8 66  0  0  0  0  0]
 [ 4 12  9  1  2 51  0  0  0  0]
 [ 9  3  7  7  1  6 48  0  0  0]
 [ 9  1  7  5 19  0  3 39  0  0]
 [ 6  2  5 14  8  3  4  3 29  0]
 [10  8 11  5  4  5 10  1  8 15]]
Classification Report:
              precision    recall  f1-score   support

         0.0       0.59      1.00      0.74        85
         1.0       0.72      0.99      0.83        76
         2.0       0.59      0.89      0.71        84
         3.0       0.60      0.73      0.66        81
         4.0       0.66      0.82      0.73        80
         5.0       0.78      0.65      0.71        79
         6.0       0.74      0.59      0.66        81
         7.0       0.91      0.47      0.62        83
         8.0       0.78      0.39      0.52        74
         9.0       1.00      0.19      0.33        77

    accuracy                           0.68       800
   macro avg       0.74      0.67      0.65       800
weighted avg       0.73      0.68      0.65       800

Accuracy: 0.6775
```

7. Prediction

```
ypred=classifier.predict(X_test)
result2a = accuracy_score(y_test,ypred)
print('Accuracy:',result2a)
```

```
Accuracy: 0.33
```

Hyperparameter Tuning (with the choice of varying k value)- Grid Search

```
leaf_size = list(range(1,500))
n_neighbors = list(range(1,3))
p=[1,2]
# convert to dictionary
hyperparameters = dict(leaf_size=leaf_size, n_neighbors=n_neighbors, p=p)
# Making model
clf = GridSearchCV(classifier, hyperparameters, cv=20)
best_model = clf.fit(X_train,y_train)
ypred = best_model.predict(X_test)
result2b = accuracy_score(y_test,ypred)
print('Accuracy:',result2b)
```

```
Accuracy: 0.37
```

The accuracy metrics After data augmentation

In order to generate more than 4000 audio files. The execution of following code chunk observed the 'memoryerror' due to local server IDE and its limited boundaries. Thus, I am able to accommodate a maximum of 4000 audio files only.

```
def audio_values(genres=GENRES,path ="genres/"):
    songs = []
    for genre in genres:
        genre_files = []
        for f_name in os.listdir(f"{path}{genre}"):
            genre_files.append(sf.read(os.path.join(f"{path}{genre}",f_name)))
        songs.append(genre_files)
    return songs
```

```
songs = audio_values()
```

Because more than 5.2 GB (4000 audio files, 400 per each genre) is causing a limitation and the usage of GPUs and cloud cluster is not available. The accuracy was only 77%/56% for train accuracy and test accuracy respectively. Although this is not to a required benchmark, the model performance was enhanced through the technique of data augmentation. That is up to 10% for each of the training accuracy as well as test accuracy. The following can be **alternatives** to construct a better model:

1. For KNN, at least 10000 samples of training data with no bias for each genre is required. However, the KNN algorithm has memory considerations as well as multi-class limitation.
2. Deep learning model which can the pull the model accuracy metric up to 90%.

In any case, the usage of GPU clusters and cloud computing is mandatory to avoid the memory error. And to my opinion, the deep learning model is highly preferable.