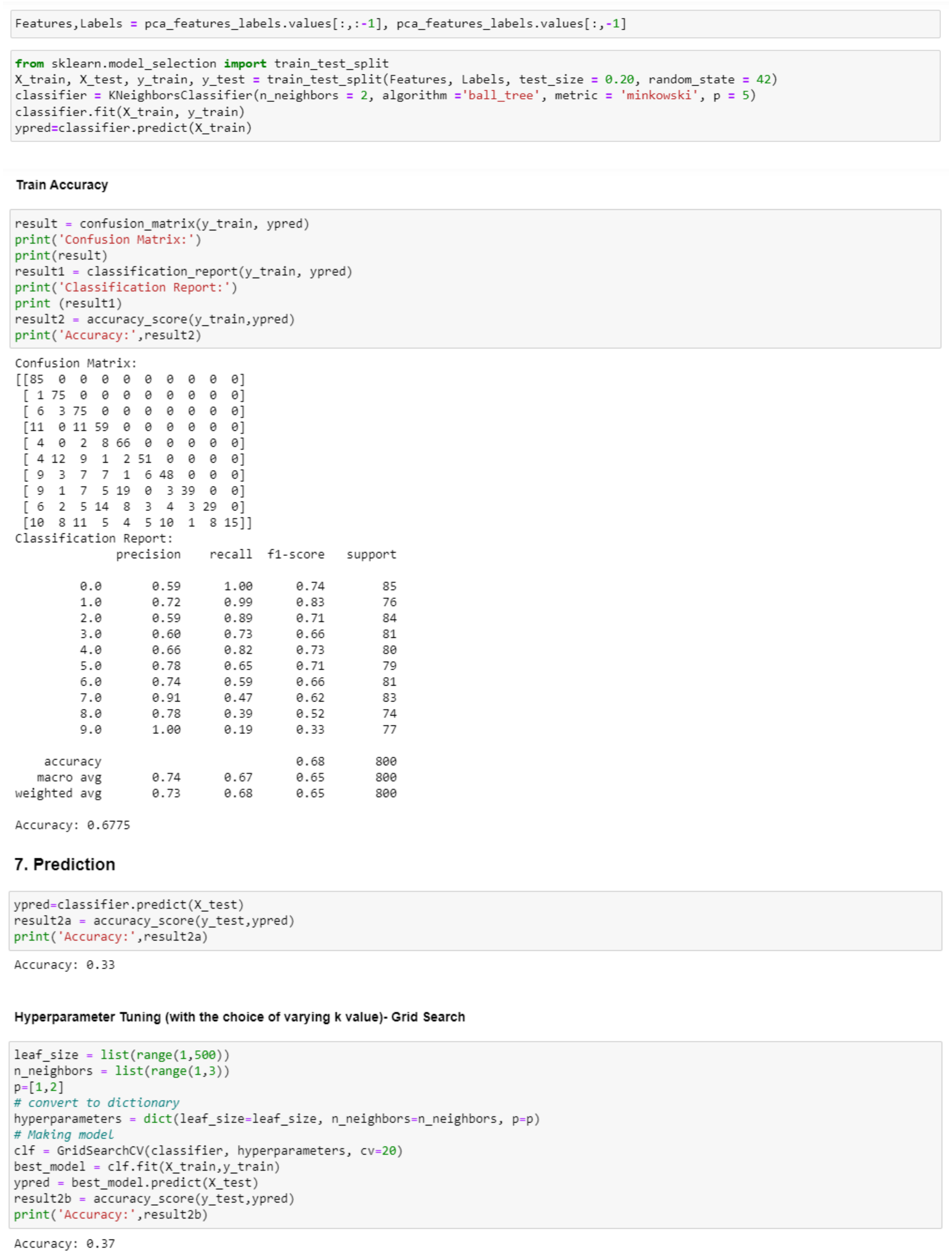
The accuracy metrics before data augmentation



```
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
```

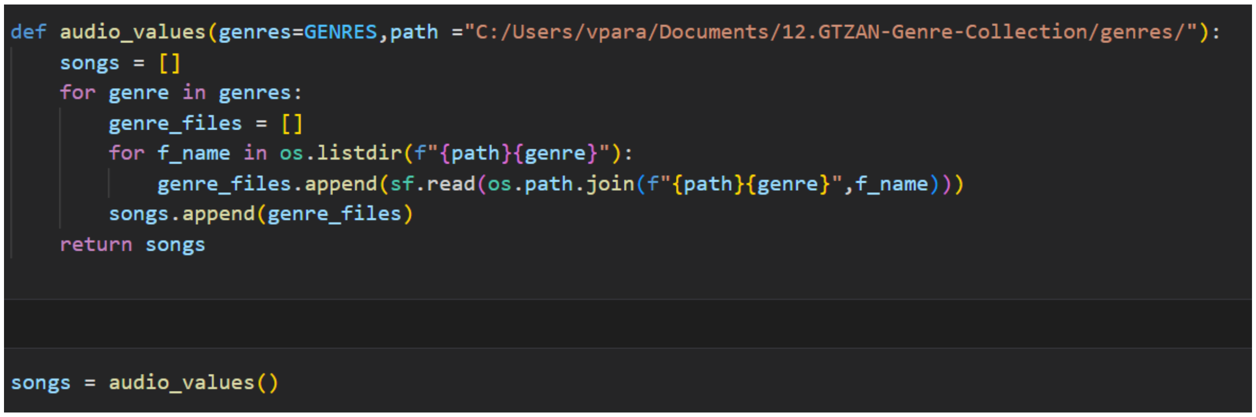
Hyperparameter Tuning (with the choice of varying k value)- Grid Search

```
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 actual criteria is to create the 400 sample for each genre and use that as training methodology for better behavior of the model.

However, while running the following code chunk the 'memoryerror' is observed and because of the IDE limitation. Only 2000 total audio files were able to participate for the data augmentation to build a model.



```
def audio_values(genres=GENRES,path ="C:/Users/vpara/Documents/12.GTZAN-Genre-Collection/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()
```

For the above-mentioned reason two alternatives shall be drafted to solve the issue to utilize the KNN as the multi-class classifier:

1. Use 400 samples to classify 5 genres (OR)
2. Use 202 samples to classify 10 genres

This amounts to 2000 audio files and will be no problem as far as the utilized Visual Studio IDE with local server is regarded. Thus, respective folders are created that contains the corresponding codes and the audio to data-frame conversion csv file.

Because 6GB is causing a limitation and the usage of GPUs and cloud cluster is not available. The accuracy was not to a required benchmark. For instance, for the alternative one, the accuracy of 71%/42% of train accuracy and test accuracy respectively.

The following can be alternatives to construct a better model:

1. For KNN, at least 4000 – 5000 samples of training data with no bias for each genre is required.
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.