Below code help you to read your data from your directory [your_TP_data] and extract feature based on [your_feature_extractor]

At the end your data will be available in x_data (features) and y_data (labels)

Entrée [201]:
```
import csv
import numpy as np
#set data_dir to the directory of your data files
data_dir= "H:\Home\Documents\ProjetIA\Dataset\Dataset/"

# Read file info file to get the list of audio files and their labels
file_list=[]
label_list=[]
with open(data_dir+"info.txt", 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        # The first column contains the file name
        file_list.append(row[0])
        # The last column contains the lable (language)
        label_list.append(row[-1])


# create a dictionary for labels
lang_dic={'EN':0,'FR':1,'AR':2,'JP':3}

# create a list of extracted feature (MFCC) for files
x_data=[]

for audio_file in file_list:
    file_feature = feature_extractor_1(data_dir+audio_file)
    #file_feature = feature_extractor_2(data_dir+audio_file)
    #add extracted feature to dataset
    x_data.append(file_feature)

# create a list of labels for files
y_data=[]
for lang_label in label_list:
    #convert the label to a value in {0,1,2,3} as the class label
    y_data.append(lang_dic[lang_label])
```

Entrée [ ]:
```
#random forest prend une matrice de taille inférieure ou égale a 2, donc je
#il a une dimension de taille 3
```

3. Shuffle your data

Using below code your data (features and corresponding labels) will be shuffled

Entrée [202]:
```
import random

# shuffle two lists
temp_list = list(zip(x_data, y_data))
random.shuffle(temp_list)
x_data, y_data = zip(*temp_list)
```

Entrée [203]:
```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_data,
```

```
                                                    y_data,
                                                    test_size=0.20,
                                                    shuffle=True)

# Train model
#clf.fit(X_train, y_train)

# Predict the test data
#y_pred = clf.predict(X_test)
```

4. Build your classifier

Now everything (almost) ready to build your classifier.

Below code is an example for creating an Random Forest classifier, training , and calculating its accuracy

Entrée [ ]:

Entrée [207]:
```
#RANDOM FOREST CLASSIFIER

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

clf = RandomForestClassifier(max_depth=70)
#en mettant max_depth a 9 on obtient 90%
#clf.fit(x_data, y_data)
# Train model
clf.fit(X_train, y_train)
# Predict the test data
y_pred = clf.predict(X_test)
# the resulted accuracy is on a small set which is same for train and test
#print("Accuracy",clf.score(x_data, y_data))
print("Accuracy : ",accuracy_score(y_test,y_pred))
```

```
Accuracy :  0.652542372881356
```

Entrée [214]:
```
#GAUSSIAN NAIVES BAYES CLASSIFIER

from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
#clf.fit(x_data, y_data)
# Train model
clf.fit(X_train, y_train)
# Predict the test data
y_pred = clf.predict(X_test)
# the resulted accuracy is on a small set which is same for train and test
#print("Accuracy with all data : ",clf.score(x_data, y_data))
print("Accuracy : ",accuracy_score(y_test,y_pred))
```

```
Accuracy :  0.4661016949152542
```

Entrée [217]:
```
#C-SUPPORT VECTOR CLASSIFIER

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
```

```
#clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
clf = make_pipeline(StandardScaler(), SVC(gamma='scale'))
#clf.fit(x_data, y_data)
# Train model
clf.fit(X_train, y_train)
# Predict the test data
y_pred = clf.predict(X_test)
print("Accuracy : ",accuracy_score(y_test,y_pred))

#print("Accuracy with all data : ",clf.score(x_data, y_data))
```

```
Accuracy :  0.635593220338983
```

Entrée [221]:

```
#MULTILAYER PERCEPTION CLASSIFIER

from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
#clf = MLPClassifier(random_state=1, max_iter=300).fit(x_data, y_data)
clf = MLPClassifier(random_state=1, max_iter=300).fit(X_train, y_train)


#clf.fit(x_data, y_data)
# Train model
clf.fit(X_train, y_train)
# Predict the test data
y_pred = clf.predict(X_test)

print("Accuracy : ",accuracy_score(y_test,y_pred))

#print("score" clf.score(X_train,y_train))
# the resulted accuracy is on a small set which is same for train and test
#il faut diviser la base de donnée et utiliser 80% des data_test pour test
#utiliser y_pred = clf.predict(X_test) pour prédire l'accuracy au lieu de
#useAccuracy(y_test,y_pred)
#print("Accuracy",clf.score(x_data, y_data))
```

```
Accuracy :  0.5084745762711864
```

5. Have you used different data for train and test?

Entrée [ ]:

```
#Yes i have used shuffle to separe my data base for test and train data
```

6. Find a model with the best accuracy

In order to find the model with highest accuracy the performance of below combiniations should be tested.

1. Compare two feature extractors
2. Find the best hyperparameter for models : for example you can google "sklearn RandomForestClassifier" and go to this link (https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) to find the RandomForestClassifier hyperparameteres (some of RandomForestClassifier's hyperparametere : n_estimators , criterion , max_depth)
3. Compare different classification algorithems