```
                                                     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 [253]:

```
#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))

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

```
Accuracy with all data :  0.9949066213921901
```

Entrée [254]:

```
#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))

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

```
Accuracy with all data :  0.4601018675721562
```

Entrée [255]:

```
#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 with all data :  0.8675721561969439
```

Entrée [258]:

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

#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
```

```
Accuracy with all data :  0.5195246179966044
```

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