

▾ Packages Importation

```
1 import pandas as pd # pandas is used to read files of the datasets
2 from sklearn.model_selection import train_test_split # train_test_split is used to part
3 from sklearn.naive_bayes import GaussianNB # GaussianNB() is the naive bayes classifier
4 from sklearn.svm import SVC # SVC() is the Support Vector Machines Classifier
5 from sklearn.neural_network import MLPClassifier # MLPClassifier us the Neural Network
6 from sklearn.metrics import confusion_matrix, classification_report # Confusion_matrix
```

▾ Dataset Preparation

```
1 df=pd.read_csv('bill_authentication.csv') # Read the dataset in a new data frame(df)
2 df.head() # Display the first five rows (5 premières lignes)
```

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

```
1 df.tail() # Display the last five rows (5 dernières lignes)
```

	Variance	Skewness	Curtosis	Entropy	Class
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

We notice that:

- We have **4 features**: Variance, Skewness, Curtosis and Entropy;
- We have **2 classes**: Class 0 and Class 1;
- We have at all **1372 samples**.

▼ Partitioning Data

`[X_train,X_test,y_train,y_test]=train_test_split(X,y,test_size=0.2)` This function create two partitions of the dataset with a test size of 0.2:

- Train dataset (**80%** of the overall dataset)
- Test dataset (**20%** of the overall dataset)
- X denotes the matrix of features X-> delete from df the coloumn class
- y denotes the label coloumn y-> troncate the df only on the coloumn class

```
1 X=df.drop('Class',axis=1)
2 y=df['Class']
3 X.head()
```

	Variance	Skewness	Curtosis	Entropy
0	3.62160	8.6661	-2.8073	-0.44699
1	4.54590	8.1674	-2.4586	-1.46210
2	3.86600	-2.6383	1.9242	0.10645
3	3.45660	9.5228	-4.0112	-3.59440
4	0.32924	-4.4552	4.5718	-0.98880

```
1 [X_train,X_test,y_train,y_test]=train_test_split(X,y,test_size=0.2)
```

- Train dataset = 80% * Number of samples (1372) = 1372 * 0.8
- Test dataset = 20% * Number of samples (1372) = 1372 * 0.2
- A.N: Train dataset = 1097.6 & Test dataset = 274.4

```
1 print("Train dataset size: {}/{}".format(len(X_train),len(y)))
2 print("Test dataset size: {}/{}".format(len(X_test),len(y)))
```

```
Train dataset size: 1097/1372
Test dataset size: 275/1372
```

- X_train: Features of train;
- y_train: Labels of X_train;
- X_test : Fetaures of test;
- y_test : Labels of X_test.

▼ Machine Learning: NB Vs SVM Vs Neural Network

We will compare between these 3 classifiers on the same partitioned data. Let's start by the initialization of the classifier which we will compare.

```
1 gnb=GaussianNB() # gnb is a naive bayes classifier
2 linear_svm =SVC(kernel='linear') # linear_svm is a Linear Support Vectors
3 rbf_svm     =SVC(kernel='rbf')    # rbf_svm is a RBF support vectors
4 sigmoid_svm =SVC(kernel='sigmoid')# sigmoid support vectors
5 ploy_svm    =SVC(kernel='poly',degree=2) # Ploynom with degree=2 as support vectors
6 neural=MLPClassifier(hidden_layer_sizes=(100,20),activation='relu',solver='adam') # neu
```

neural=MLPClassifier parametres:

- hidden_layer_sizes=(100,20): 4x100x20x2
- activation='relu': activation function in all neurons is Relu(x)
- solver='adam' : algorithm for weights' update during the training
- defalut value of learning rate (alph): 0.001

Now, we will move to the training process with using of the fit() function.

```
1 gnb.fit(X_train,y_train) # Train Guassian NB classifier
2 linear_svm.fit(X_train,y_train) # Train SVM
3 rbf_svm.fit(X_train,y_train)
4 sigmoid_svm.fit(X_train,y_train)
5 ploy_svm.fit(X_train,y_train)
6 neural.fit(X_train,y_train) # Train Neural Network - finding the best weight matrix

MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100, 20), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

Now, we will test the learned models!

- We will ask the model to give a prediction based on its learning
- Each Classifier will produce a prediction; y_nb,y_linear_svm,etc.

We have two types of labels:

- y_test: true label coming from the initial dataset
- y_nb, y_linear_svm, y_rbf_svm, y_sigmoid_svm, y_ploy_svm et y_neural: are the labels predicted by the models: naive bayes, svm with all kernels and neural network !!! Le modèle est performant si et seulement si sa prédiction égale aux vrais labels !!!

```

1 y_nb=gnb.predict(X_test)
2 y_linear_svm=linear_svm.predict(X_test)
3 y_rbf_svm=rbf_svm.predict(X_test)
4 y_ploy_svm=ploy_svm.predict(X_test)
5 y_sigmoid_svm=sigmoid_svm.predict(X_test)
6 y_neural=neural.predict(X_test)

```

▼ Performance Evaluation

```

1 print ('***** Peformance Evaulation of Naive Bayes *****')
2 print(confusion_matrix(y_test,y_nb))
3 print(classification_report(y_test,y_nb))
4 print ('***** Peformance Evaulation of Linear SVM *****')
5 print(confusion_matrix(y_test,y_linear_svm))
6 print(classification_report(y_test,y_linear_svm))
7 print ('***** Peformance Evaulation of RBF SVM *****')
8 print(confusion_matrix(y_test,y_rbf_svm))
9 print(classification_report(y_test,y_rbf_svm))
10 print ('***** Peformance Evaulation of Sigmoid SVM *****')
11 print(confusion_matrix(y_test,y_sigmoid_svm))
12 print(classification_report(y_test,y_sigmoid_svm))
13 print ('***** Peformance Evaulation of Polynomial (2) SVM *****')
14 print(confusion_matrix(y_test,y_ploy_svm))
15 print(classification_report(y_test,y_ploy_svm))
16 print ('***** Peformance Evaulation of Neural Network *****')
17 print(confusion_matrix(y_test,y_neural))
18 print(classification_report(y_test,y_neural))

```

***** Peformance Evaulation of Naive Bayes *****

```

[[129  17]
 [ 29 100]]

```

	precision	recall	f1-score	support
0	0.82	0.88	0.85	146
1	0.85	0.78	0.81	129
accuracy			0.83	275
macro avg	0.84	0.83	0.83	275
weighted avg	0.83	0.83	0.83	275

***** Peformance Evaulation of Linear SVM *****

```

[[143   3]
 [  0 129]]

```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	146
1	0.98	1.00	0.99	129
accuracy			0.99	275
macro avg	0.99	0.99	0.99	275
weighted avg	0.99	0.99	0.99	275

***** Peformance Evaulation of RBF SVM *****

```

[[144   2]

```

```
[ 0 129]]
      precision    recall  f1-score   support

     0       1.00      0.99      0.99       146
     1       0.98      1.00      0.99       129

 accuracy          0.99
 macro avg          0.99
weighted avg          0.99
```

***** Peformance Evauation of Sigmoid SVM *****

```
[[106  40]
 [ 50  79]]
      precision    recall  f1-score   support

     0       0.68      0.73      0.70       146
     1       0.66      0.61      0.64       129

 accuracy          0.67
 macro avg          0.67
weighted avg          0.67
```

***** Peformance Evauation of Polynomial (2) SVM *****

```
[[140   6]
 [  0 129]]
      precision    recall  f1-score   support

     0       1.00      0.96      0.98       146
     1       0.96      1.00      0.98       129

 accuracy          0.98
 macro avg          0.98
weighted avg          0.98
```