REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE HOUARI BOUMEDIENE

Faculté d'Informatique

Filière
Informatique

Spécialité
Bio-Informatique

Mini projet : Data Mining

Module: Data mining FD2

Présenté par :
-HARCHAOUI Louiza Feriel
-SMAIL Mehrez

Année Universitaire : 2023-2024

Table des matières

Index des figures

Le Data Mining

Ce concept existe depuis plus d'un siècle mais il est devenu réellement connu dans les années 1980. Depuis, un long chemin a été parcouru. Les entreprises utilisent désormais le data mining et le machine learning pour accomplir de nombreuses tâches, de l'amélioration du processus de vente à l'interprétation des données financières pour l'investissement.

1.1 Définition

Le data mining désigne le processus d'analyse de volumes massifs de données et du Big Data sous différents angles afin d'identifier des relations entre les data et de les transformer en informations exploitables. Ce dispositif rentre dans le cadre de la Business Intelligence et a pour but d'aider les entreprises à résoudre des problèmes, à atténuer des risques et à identifier et saisir de nouvelles opportunités business.

En français, ce processus porte différents noms :

- Exploration de données
- Fouille de données
- Forage de données

1.2 L'utilité de l'exploration de données aujourd'hui

Aujourd'hui, le **data mining est utilisé dans de nombreux secteurs d'activité** comme la recherche, le marketing, le développement de produits, la santé ou encore l'éducation.

Ce processus permet de résoudre rapidement des problèmes qui, jusqu'alors, demandaient énormément de temps pour être réglés manuellement.

L'utilisation de techniques statistiques diverses pour analyser les données permet aux utilisateurs d'identifier des modèles, des tendances et des corrélations qui n'apparaissaient pas clairement au départ. Grâce aux résultats des différentes analyses successives, ils peuvent prédire ce qui est susceptible de se produire et prendre des mesures pour influencer et maximiser les résultats commerciaux.

1.3 Le processus de data mining

Le processus de data mining, également appelé fouille de données, comprend plusieurs étapes qui visent à extraire des informations utiles et des modèles prédictifs à partir de grandes quantités de données. Voici les principales étapes du processus de data mining :

1. Définition de l'objectif :
    - Identifier clairement l'objectif de la fouille de données.
    - Définir les critères de succès.
2. Collect des données :
    - Rassembler les données nécessaires à l'analyse.
    - Assurer la qualité des données en nettoyant et en prétraitant si nécessaire.
3. Exploration des données (Exploratory Data Analysis - EDA) :
    - Analyser les statistiques descriptives.
    - Identifier des tendances, des schémas et des anomalies potentielles.
4. Prétraitement des données :
    - Gérer les valeurs manquantes.
    - Normaliser les données.
    - Éliminer le bruit et les outliers.
5. Choix des algorithmes :
    - Sélectionner les algorithmes de data mining appropriés en fonction de l'objectif.
    - Les algorithmes peuvent inclure la classification, la régression, le clustering, etc.
6. Modélisation :
    - Appliquer les algorithmes choisis sur les données.
    - Créer des modèles prédictifs ou des modèles descriptifs.
7. Évaluation des modèles :
    - Évaluer la performance des modèles en utilisant des mesures appropriées .
    - Réajuster les modèles si nécessaire.
8. Interprétation des résultats :
    - Analyser et interpréter les modèles obtenus.
    - Extraire des informations et des connaissances utiles.
9. Mise en œuvre des résultats :
    - Intégrer les résultats du data mining dans les processus décisionnels de l'entreprise.
    - Mettre en œuvre des actions basées sur les découvertes.

10. Documentation et présentation :
    - Documenter le processus de data mining.
    - Présenter les résultats de manière compréhensible pour les parties prenantes.

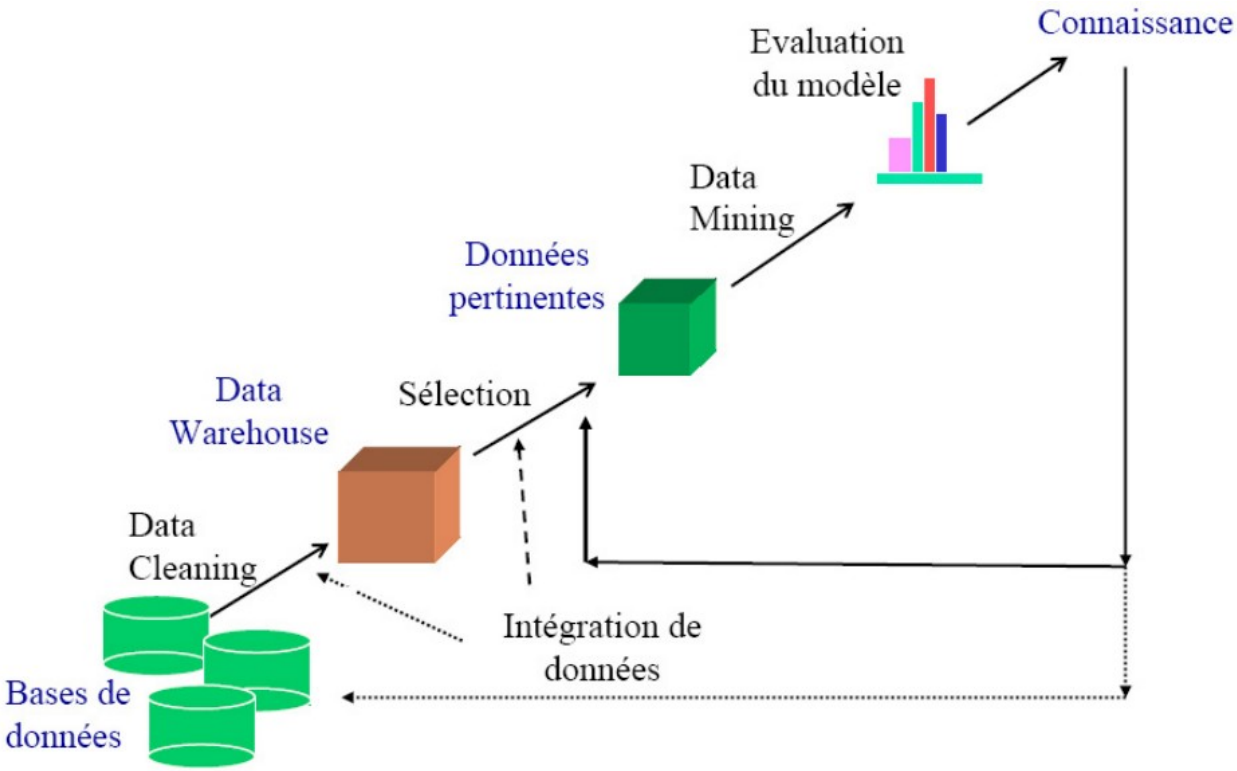


Figure 01 : processus de data mining

Il est important de noter que le processus de data mining n'est pas toujours linéaire, et certaines étapes peuvent nécessiter un retour en arrière ou une réitération en fonction des résultats obtenus. De plus, le choix des outils et des techniques dépend des spécificités du problème et des données disponibles.

2 L'apprentissage automatique

2.1 Définition

L'apprentissage automatique est un sous-domaine de l'intelligence artificielle (IA) qui se concentre sur la conception de systèmes qui apprennent ou améliorent le rendement en fonction des données qu'ils consomment.

L'intelligence artificielle est un terme général qui se rapporte aux systèmes ou aux machines qui imitent l'intelligence humaine. L'apprentissage automatique et l'intelligence artificielle sont souvent évoqués ensemble les termes sont parfois utilisés de façon interchangeable, mais ne signifient pas la même chose. Une importante distinction est que même si tout

apprentissage automatique repose sur l'intelligence artificielle,cette dernière concerne bien plus que l'apprentissage automatique.

2.2 Apprentissage supervisé

L'apprentissage supervisé commence généralement par un ensemble de données bien défini et une certaine compréhension de la façon dont ces données sont classifiées, l'apprentissage supervisé a pour but de déceler des modèles au sein des données et de les appliquer à un processus analytique. Ces données comportent des caractéristiques associées à des libellés qui définissent leur signification L'apprentissage supervisé peut être regroupé en :

2.2.1 La méthode des K plus proches voisins

Les 'k plus proches voisins' ou k-nearest neighbors en anglais (d'où l'appellation knn) est une méthode non paramétrique dans laquelle le modèle mémorise les observations de l'ensemble d'apprentissage pour la classification des données de l'ensemble de test, cet algorithme est qualifiée comme paresseux (Lazy Learning) car il n'apprend rien pendant la phase d'entraînement. Pour prédire la classe d'une nouvelle donnée d'entrée, il va chercher ses K voisins les plus proches (en utilisant la distance euclidienne, ou autres) et choisira la classe des voisins majoritaires.

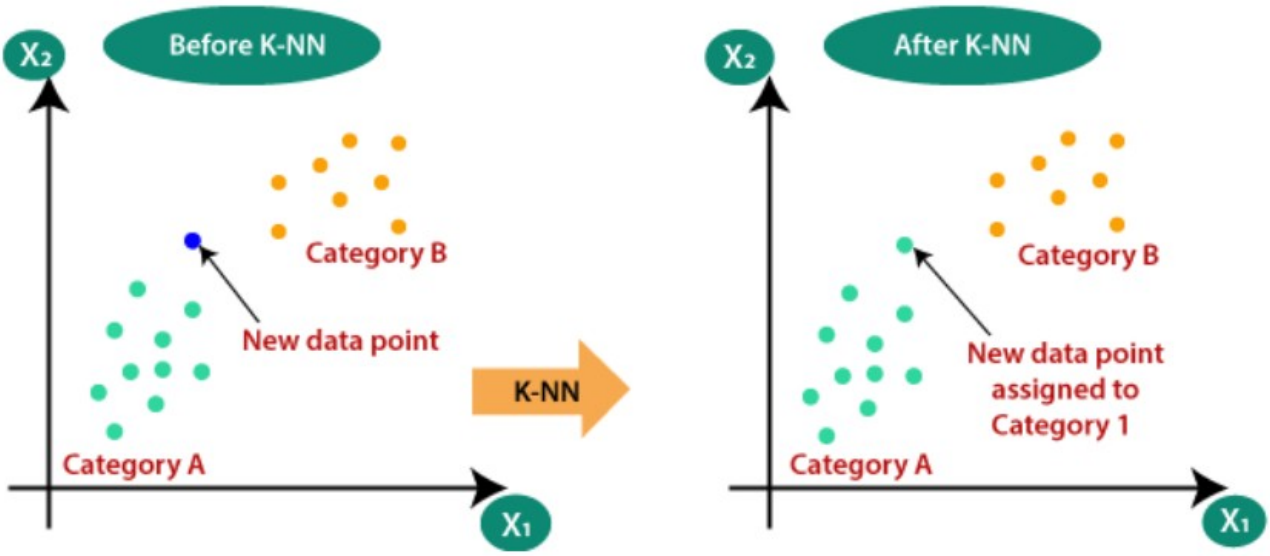


Figure 01 : KNN

Le fonctionnement de K-NN peut être expliqué sur la base de l'algorithme ci-dessous :

- Sélectionnez le nombre K des voisins
- Calculer la distance euclidienne du nombre K de voisins

• Prenez les K voisins les plus proches selon la distance euclidienne

• comptez le nombre de points de données dans chaque catégorie.

• Attribuez les nouveaux points de données à la catégorie pour laquelle le nombre de voisins est maximal.

• Notre modèle est prêt.

Exemple :

Importing librairies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.naive_bayes import GaussianNB,CategoricalNB
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
plt.style.use('seaborn')
from sklearn.svm import SVC
from sklearn import svm
import warnings
warnings.filterwarnings('ignore')
```

Figure 02 : Importe les Bibio

Lecture des fichiers de données

Pour ce TP, nous allons lire les données à partir d'un fichier csv.

```
Data = pd.read_csv('Data/breast.csv')
Data
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...	...	...	...	...	...	...	...	...
495	14.87	20.21	96.12	680.9	0.09587	0.08345	0.06824	0.04951
496	12.65	18.17	82.69	485.6	0.10760	0.13340	0.08017	0.05074
497	12.47	17.31	80.45	480.1	0.08928	0.07630	0.03609	0.02369
498	18.49	17.52	121.30	1068.0	0.10120	0.13170	0.14910	0.09183
499	20.59	21.24	137.80	1320.0	0.10850	0.16440	0.21880	0.11210

500 rows × 19 columns

Figure 04 :

Figure 03 : Affichier Data

Box-Plot

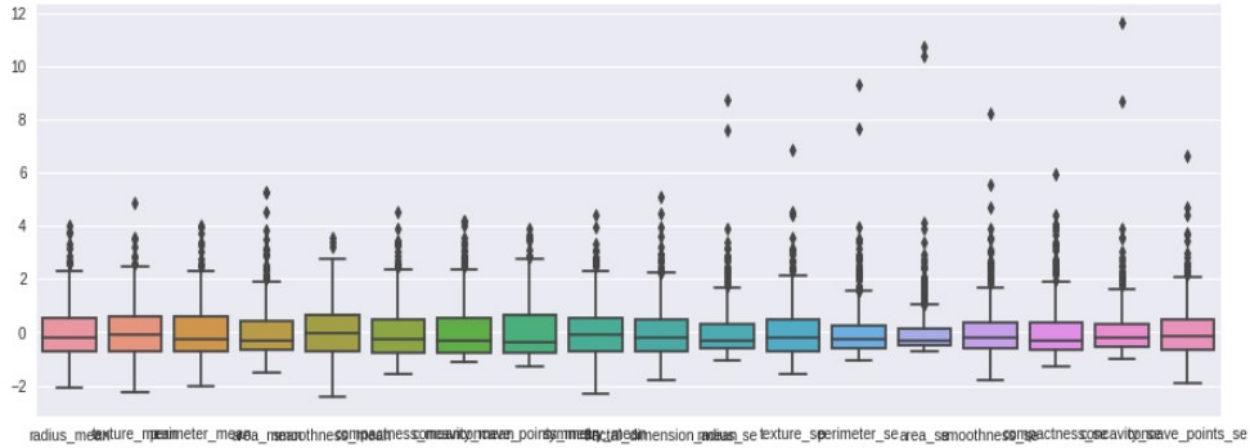


Figure 06 :

Figure 05 : Box-Plot

Affichage scater de Data

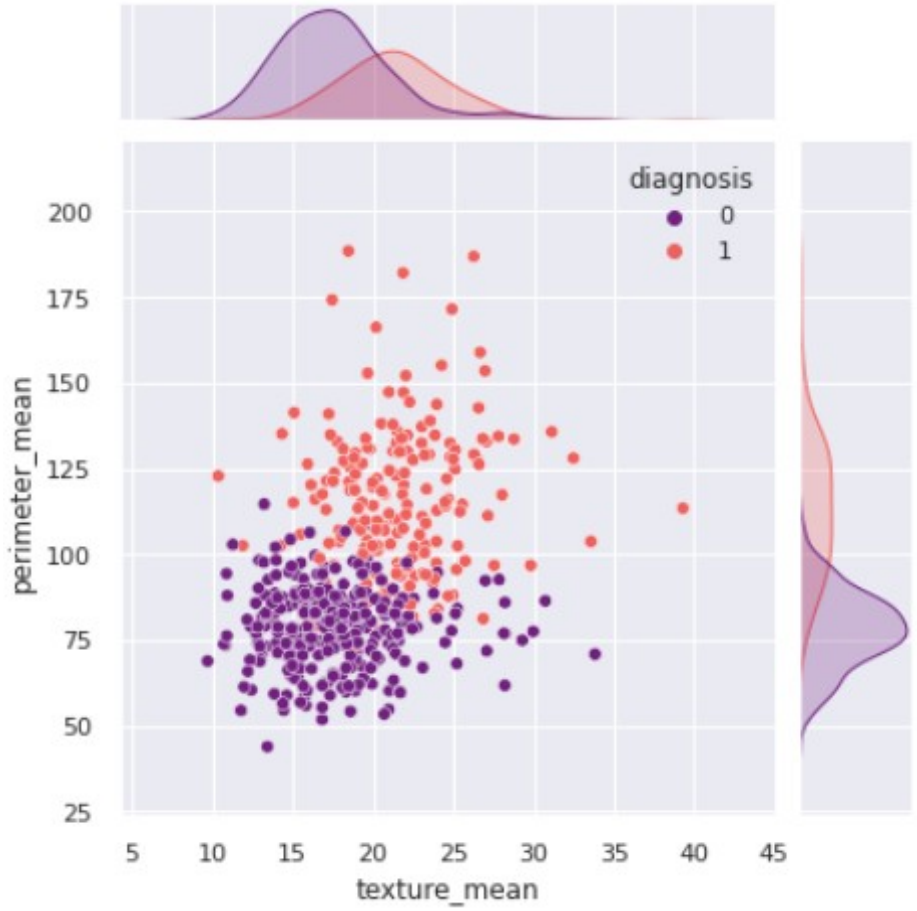


Figure 06 : Affichage scater de Data

Applications de KNN

```
KNN=KNeighborsClassifier(n_neighbors=3)
KNN.fit(x_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
from prettytable import PrettyTable

x = PrettyTable(["Model", "Train SCORE", "Test SCORE"])
z=str(int(KNN.score(x_train,y_train)*100))+"%"
v=str(int(KNN.score(x_test,y_test)*100))+"%"
x.add_row(["KNN",z,v])
print(x)
```

```
+-------+-------------+------------+
| Model | Train SCORE | Test SCORE |
+-------+-------------+------------+
|  KNN  |     96%     |    90%     |
+-------+-------------+------------+
```

Figure 08 :

Figure 07 : Applications de KNN

Confusion Matrices KNN

```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92        61
           1       0.87      0.87      0.87        39

    accuracy                           0.90       100
   macro avg       0.89      0.89      0.89       100
weighted avg       0.90      0.90      0.90       100
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay
```

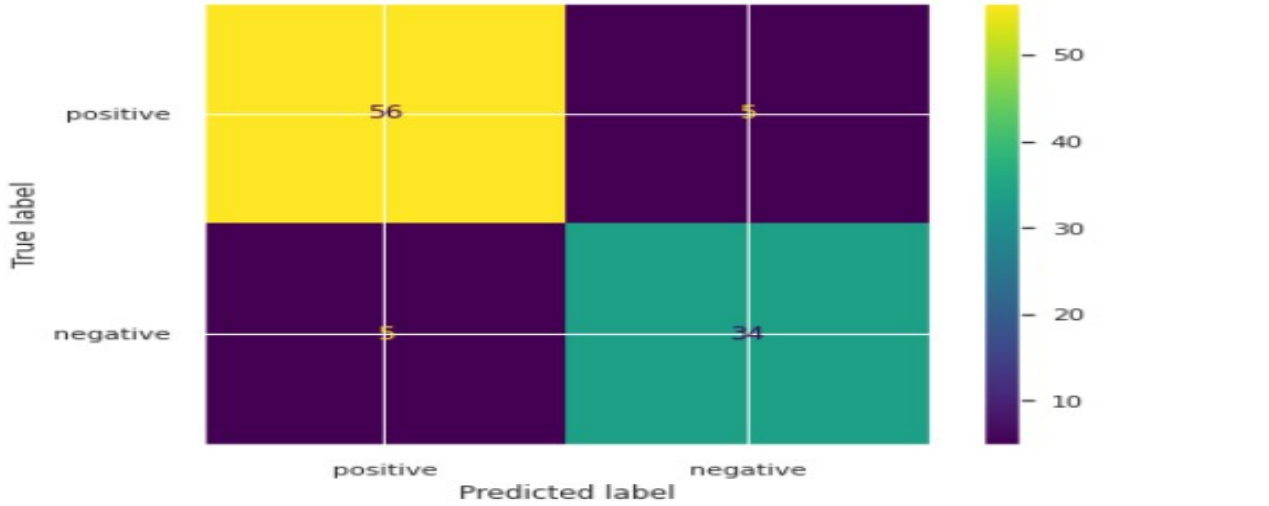


Figure 08 : Confusion Matrices KNN

Cross-validation: evaluating estimator performance

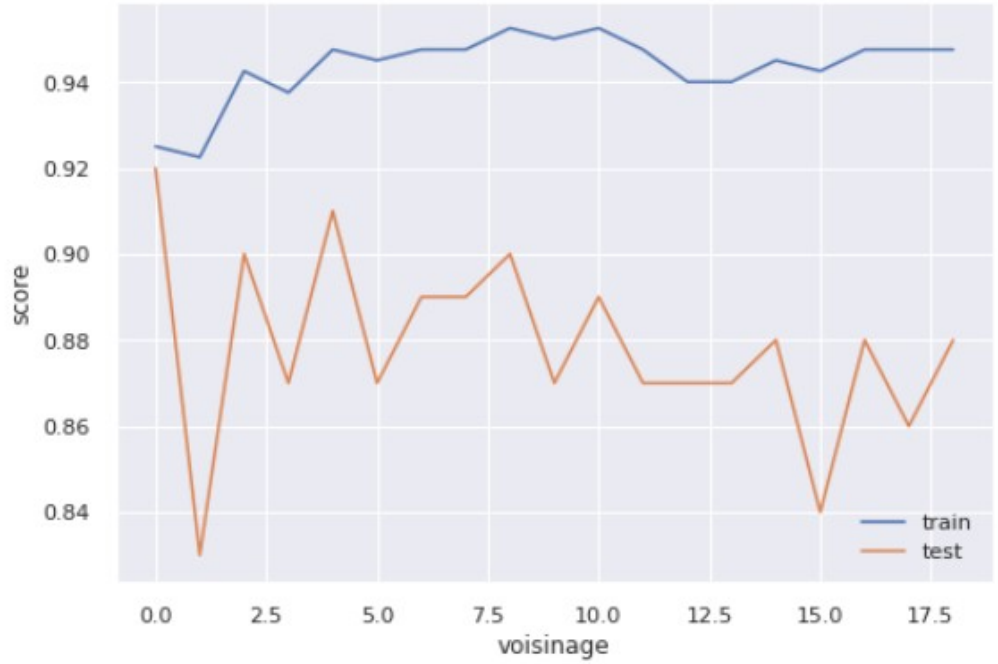


Figure 10 :

Figure 09 : Cross-validation: evaluating estimator performance

la courbe (ROC AUC)

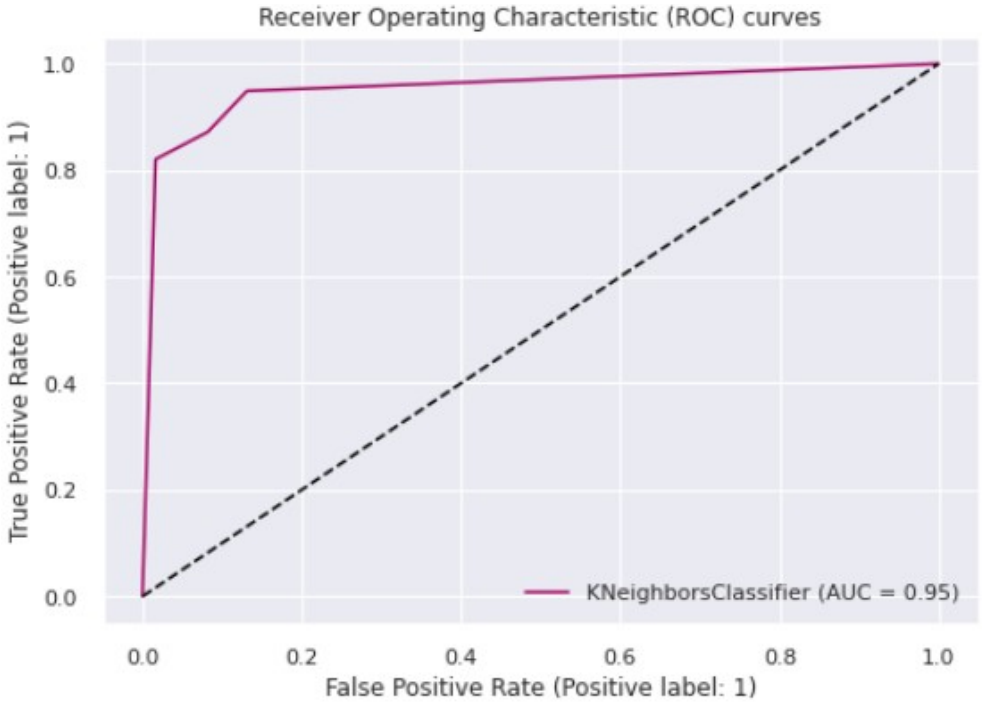


Figure 10 : la courbe (ROC AUC)

2.2.2 La méthode de Naive Bayes

Est une ancienne méthode de classification et de sélection de prédicteurs qui connaît une certaine renaissance en raison de sa simplicité et de sa stabilité. Les problèmes auxquels le modèle Naive Bayes est généralement appliqué se répartissent en deux grandes catégories: la sélection et la classification des caractéristiques.**Sélection de fonction.** Il s'agit d'applications dans lesquelles vous choisissez un sous-ensemble de prédicteurs à partir d'un plus grand ensemble de variables. La plupart des méthodes de classification ne fonctionnent pas bien lorsqu'il y a trop de prédicteurs. Etant donné que, dans la pratique, de nombreux prédicteurs ne contribuent pas à la classification, l'étape de préclassification consiste à trouver un sous-ensemble de prédicteurs pertinents.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

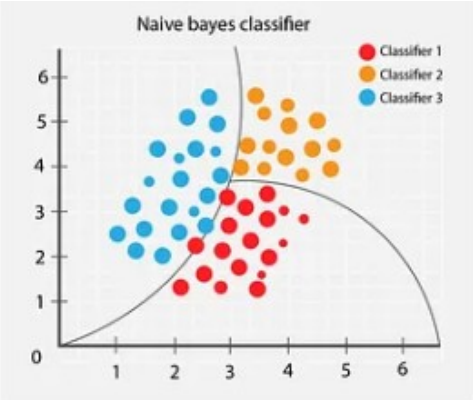


Figure 11 : La méthode de Naive Bayes

Applications de Naive Bayes

```
1 Naive=GaussianNB()
2 Naive.fit(x_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
1 from prettytable import PrettyTable
2
3 x = PrettyTable(["Model", "Train SCORE", "Test SCORE"])
4 z=str(int(Naive.score(x_train,y_train)*100))+"%"
5 v=str(int(Naive.score(x_test,y_test)*100))+"%"
6 x.add_row(["Naive Bayes",z,v])
7 print(x)
```

```
+-------------+-------------+------------+
|    Model    | Train SCORE | Test SCORE |
+-------------+-------------+------------+
| Naive Bayes |     91%     |    88%     |
+-------------+-------------+------------+
```

Figure 11 : Applications de Naive Bayes

Confusion Matrices Naive Bayes

	precision	recall	f1-score	support
0	0.89	0.92	0.90	61
1	0.86	0.82	0.84	39
accuracy			0.88	100
macro avg	0.88	0.87	0.87	100
weighted avg	0.88	0.88	0.88	100

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
```

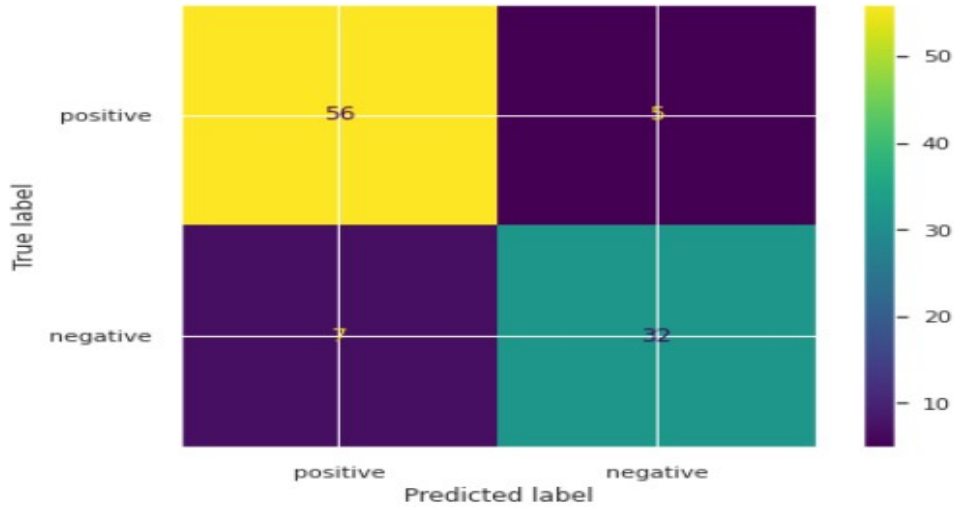


Figure 12 : Confusion Matrices KNN

Cross-validation: evaluating estimator performance

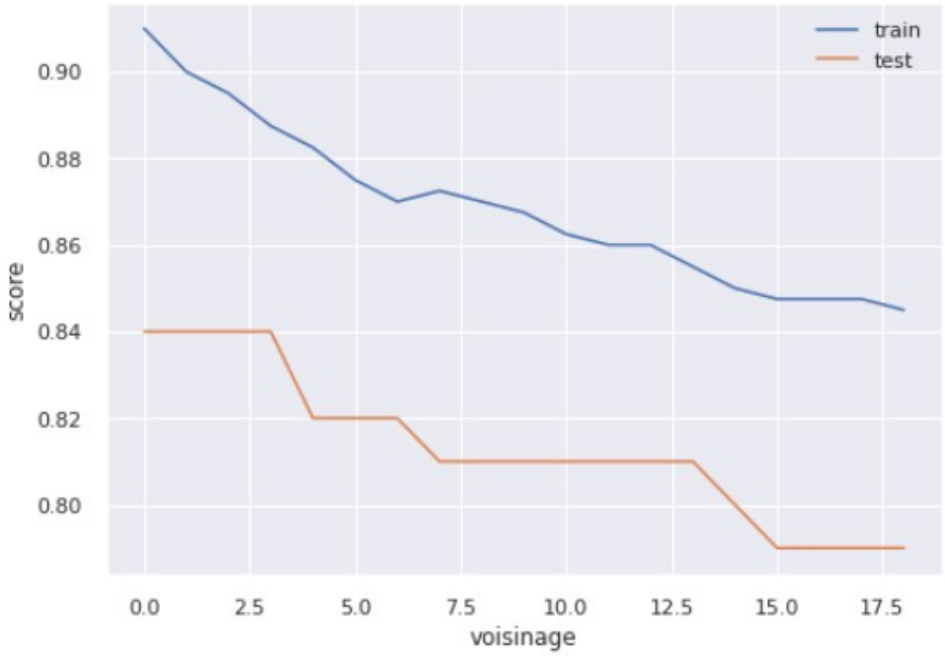


Figure 13 : Cross-validation: evaluating estimator performance

la courbe (ROC AUC)

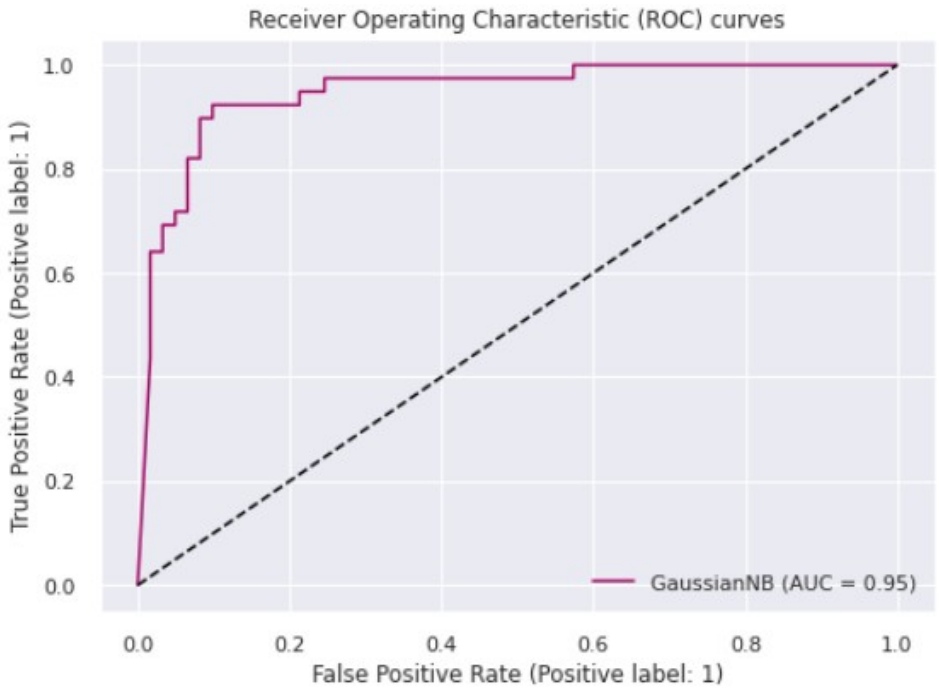


Figure 14 : la courbe (ROC AUC)

2.2.3 La méthode d'arbre de décisions

Un arbre de décisions est un algorithme d'apprentissage supervisé non paramétrique, qui est utilisé à la fois pour les tâches de classification et régression. Il a une structure hiérarchique, une structure arborescente, qui se compose d'un noeud racine, de branches, de nœuds interne et de nœuds feuille.

Entropie et gain d'informations

$$\text{Entropy}(S) = -\sum_{c \in C} p(c)\log_2 p(c)$$

$$\text{Information Gain}(S,a) = \text{Entropy}(S) - \sum_{\text{vevcalues}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

Applications d'arbre de décisions

```
#entropy ,gini
AD=DecisionTreeClassifier(criterion = 'entropy',max_depth=10)
AD.fit(x_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=10)
```

```
from prettytable import PrettyTable

x = PrettyTable(["Model", "Train SCORE", "Test SCORE"])
z=str(int(AD.score(x_train,y_train)*100))+"%"
v=str(int(AD.score(x_test,y_test)*100))+"%"
x.add_row([" DecisionTree",z,v])
print(x)
```

```
+---------------+-------------+------------+
|     Model     | Train SCORE | Test SCORE |
+---------------+-------------+------------+
|  DecisionTree |     100%    |    90%     |
+---------------+-------------+------------+
```

Figure 15 : Applications d'arbre de décisions

Confusion Matrices d'arbre de décisions

	precision	recall	f1-score	support
0	0.79	0.81	0.80	99
1	0.64	0.62	0.63	55
accuracy			0.74	154
macro avg	0.72	0.71	0.71	154
weighted avg	0.74	0.74	0.74	154

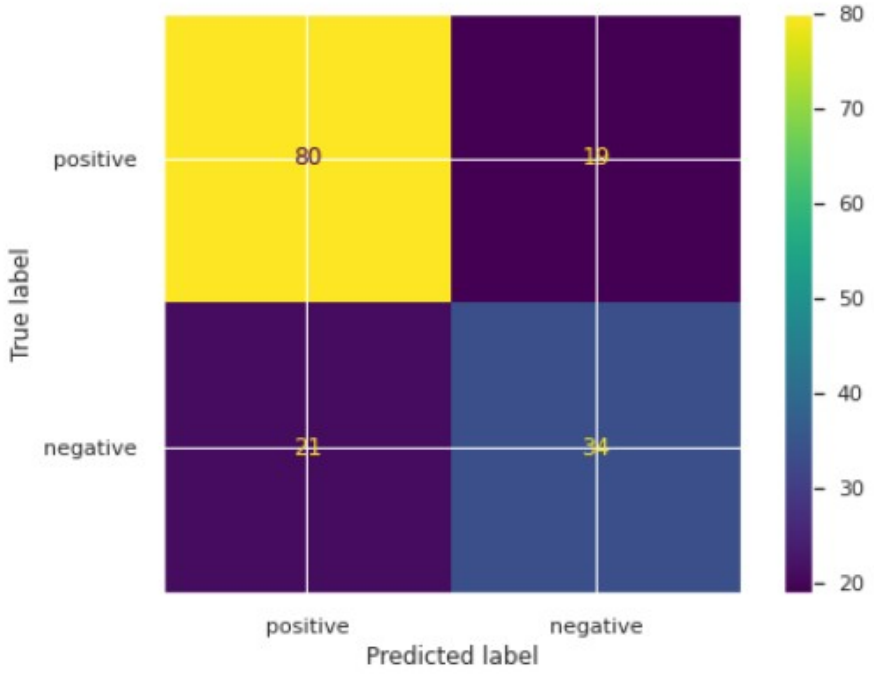


Figure 16 : Confusion Matrices d'arbre de décisions

Cross-validation: evaluating estimator performance

Figure 18 :

Figure 17 : Cross-validation: evaluating estimator performance

la courbe (ROC AUC)

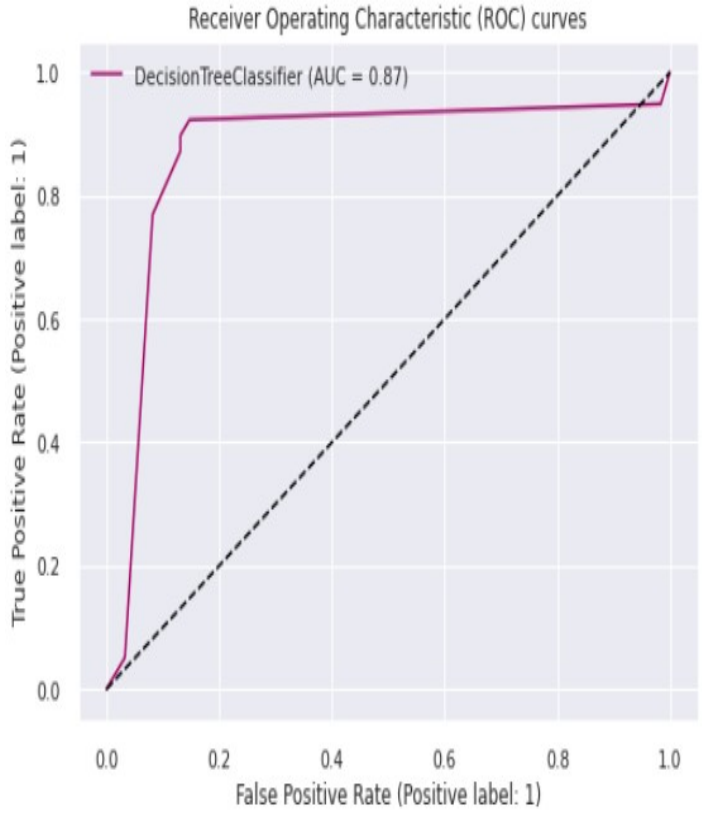


Figure 18 : la courbe (ROC AUC)

DecisionTreeClassifier

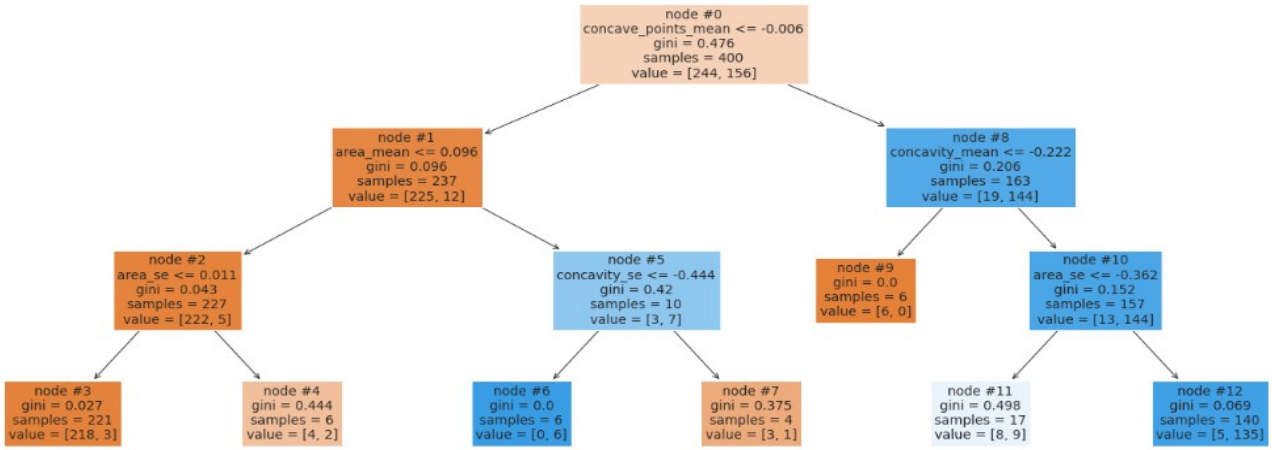


Figure 19 : DecisionTreeClassifier

2.2.4 La méthode de Reseau Neurones

Il offre plusieurs avantages par rapport à d'autres modèles et a une architecture complexe et efficace.

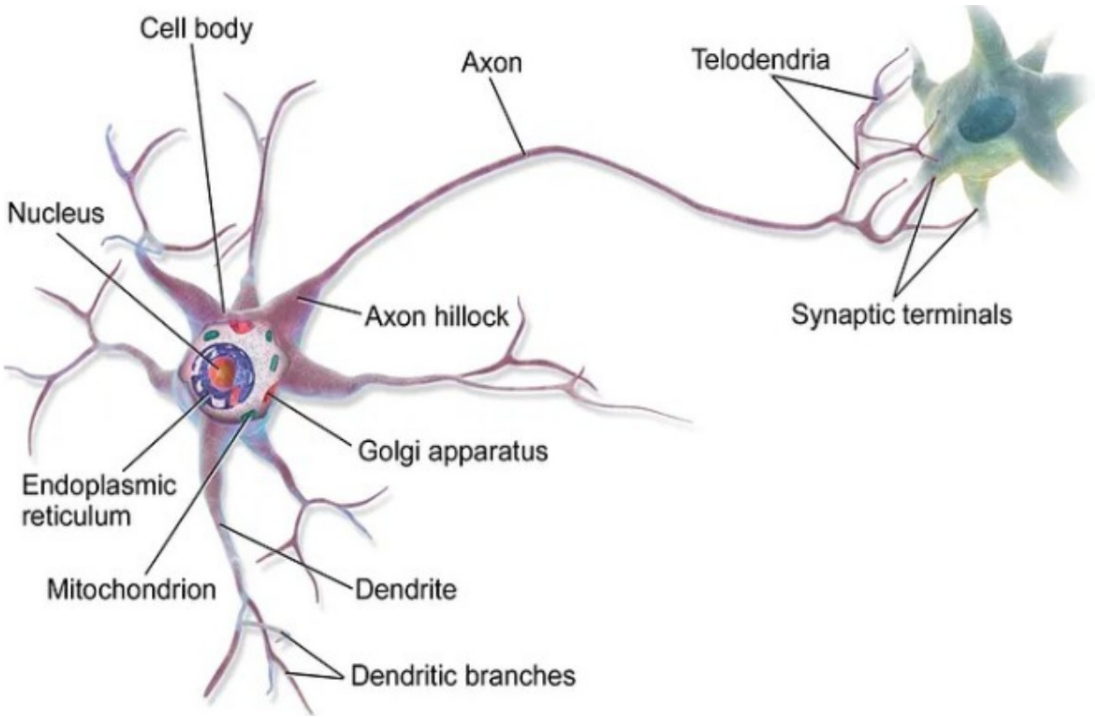


Figure 20 : La méthode de Reseau Neurones

Comprendre l'architecture du MLPClassifier

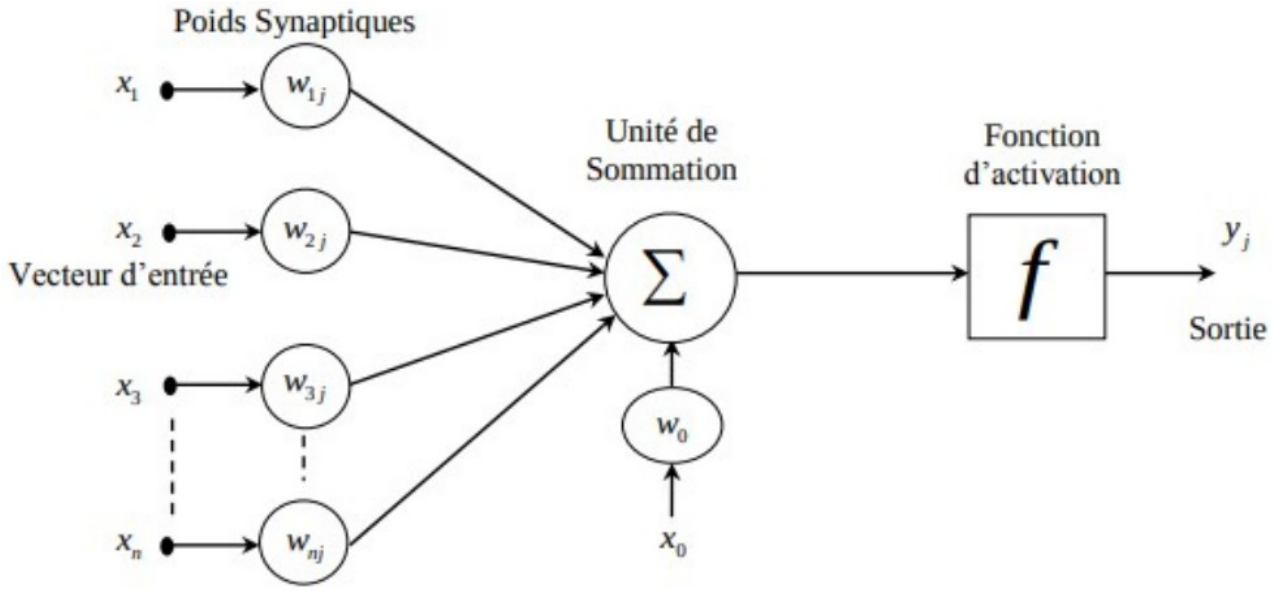


Figure 21 : Comprendre l'architecture du MLPClassifier

Initialisation(dimensions) : cette fonction nous permettre d'initialiser les paramètres poids w et bais b, elle prend comme paramètre la dimension de notre benchmark ou bien plus exactement le nombre de variable contenant dans notre benchmark

• forward_propagation(X, parametres) : cette fonction nous permet de générer la fonction d'activation en partant des paramètres et en utilisant la fonction sigmoïde

$$Z = w1\,x1 + w2\,x2 + \cdots + b$$

$$A = \frac{1}{\left(1 + e^{(-z)}\right)}$$

back_propagation(y, parametres, activations) : cette fonction retourne le gradients il consiste à améliorer les paramètres wi de toutes les connexions en cherchant à minimer le coût (loss)

$$Loss = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(a^{(i)}\right) + \left(1 + y^{(i)} \log\left(1 + a^{(i)}\right)\right)$$

$$G = \frac{1}{m}\sum \left(a - y\right) x_i$$

update(gradients, parametres, learning_rate) : cette fonction est une fonction qui mis a jours les paramètres w et b en les améliorant grâce a la fonction du gradient

Applications de Reseau Neurones

```
# Create an instance of MLPClassifier
mlf = MLPClassifier(hidden_layer_sizes=(45), max_iter=1000)
# Train the model
mlf.fit(x_train, y_train)
```

```
MLPClassifier(hidden_layer_sizes=45, max_iter=1000)
```

```
from prettytable import PrettyTable

x = PrettyTable(["Model", "Train SCORE", "Test SCORE"])
z=str(int(mlf.score(x_train,y_train)*100))+"%"
v=str(int(mlf.score(x_test,y_test)*100))+"%"
x.add_row([" Reseau Neurones",z,v])
print(x)
```

```
+-----------------+-------------+------------+
|      Model      | Train SCORE | Test SCORE |
+-----------------+-------------+------------+
|  Reseau Neurones |     99%     |    92%     |
+-----------------+-------------+------------+
```

Figure 22 : Applications de Reseau Neurones

Confusion Matrices Réseau Neurones

```
              precision    recall  f1-score   support

           0       0.95      0.92      0.93        61
           1       0.88      0.92      0.90        39

    accuracy                           0.92       100
   macro avg       0.91      0.92      0.92       100
weighted avg       0.92      0.92      0.92       100
```

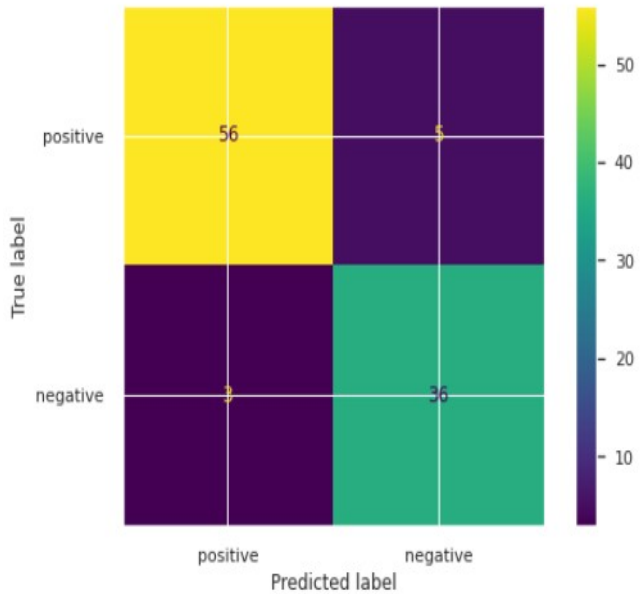


Figure 23 : Confusion Matrices Reseau Neurones

Cross-validation: evaluating estimator performance

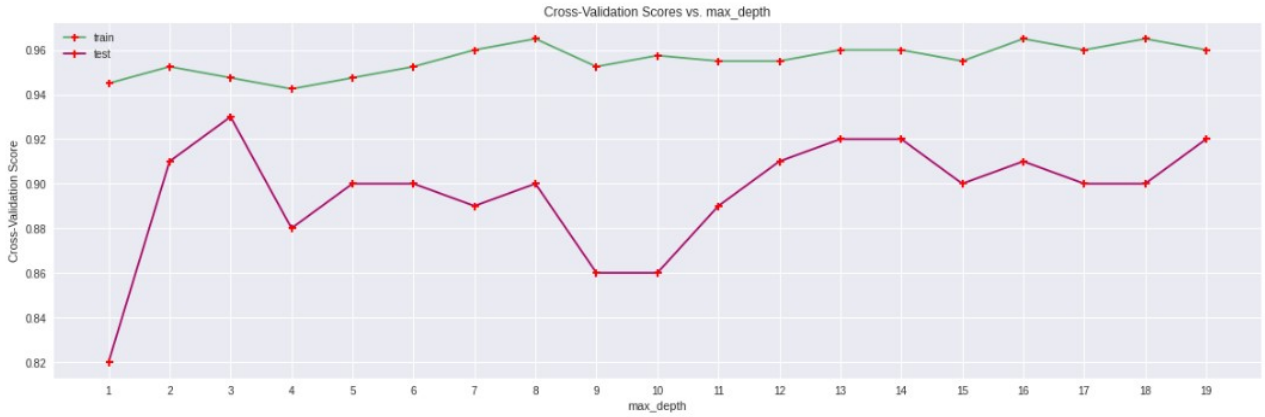


Figure 24 : Cross-validation: evaluating estimator performance

La courbe (ROC AUC)

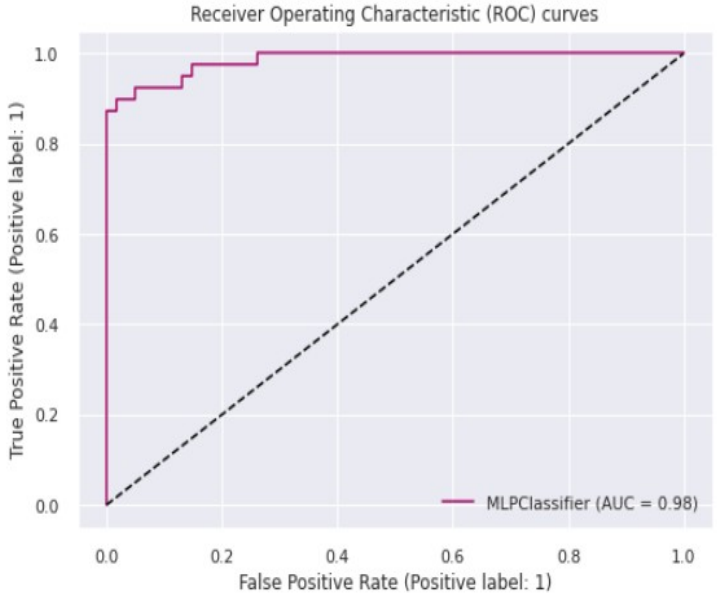


Figure 25 : La courbe (ROC AUC)

2.2.5 La méthode Support Vector Machine ou SVM

Les machines à vecteurs de support ou SVM sont des modèles d'apprentissage automatique supervisés, c'est-à-dire qu'elles utilisent des ensembles de données étiquetés pour entraîner les algorithmes. SVM peut résoudre des problèmes à la fois linéaires et non linéaires, et par la notion de marge, il classe entre différentes classes. Cependant, il est essentiellement utilisé pour les problèmes de classification dans l'apprentissage automatique. L'objectif de

l'algorithme est de trouver la ligne la plus fine ou la limite de décision permettant de séparer l'espace à n dimensions en classes de manière à pouvoir placer les nouveaux points de données dans la bonne classe à l'avenir. Cette limite de décision est appelée un hyperplan. Dans la plupart des cas, les SVM ont une précision supérieure à celle des arbres de décision, des KNN, des classificateurs Naive Bayes, des régressions logistiques, etc. En plus de cela, les SVM sont bien connus pour surpasser les réseaux neuronaux à quelques reprises. Les SVM sont fortement recommandés en raison de leur mise en œuvre plus facile et de leur plus grande précision avec moins de calculs.

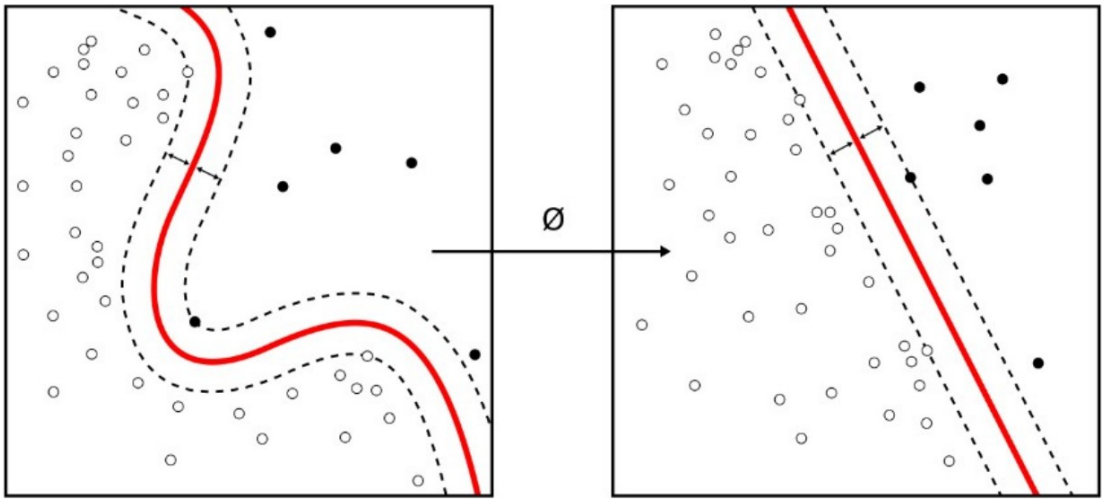


Figure 26 : La méthode Support Vector Machine ou SVM

Comprendre l'architecture du SVM

Marge :

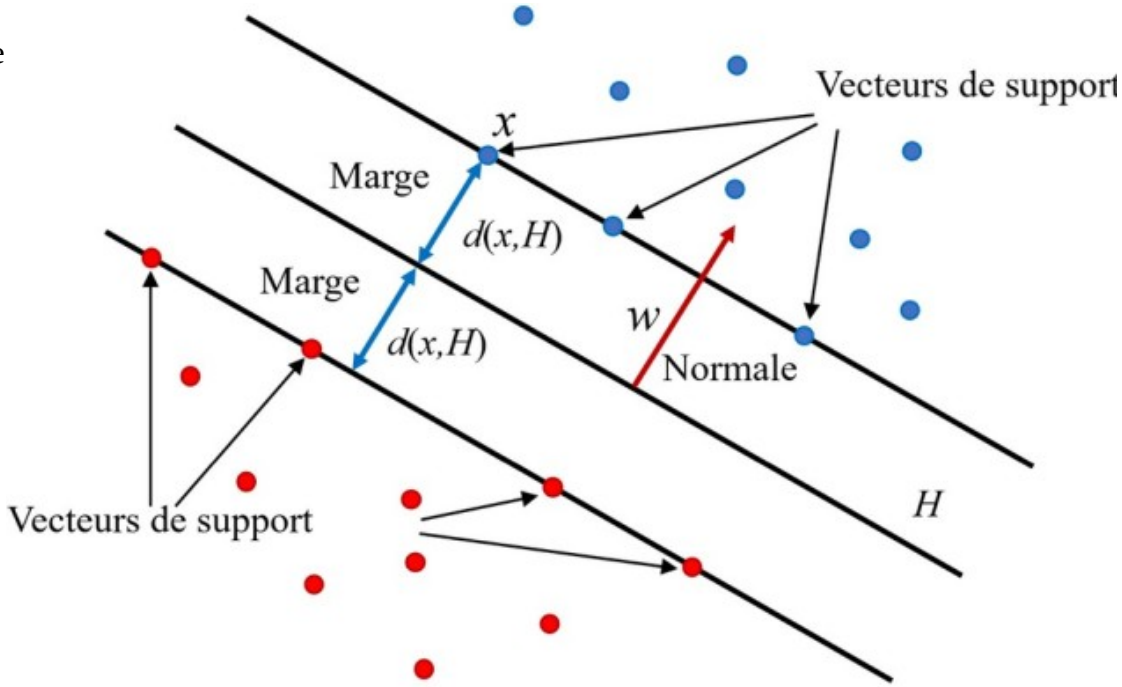


Figure 27 : Comprendre l'architecture du SVM

Distance entre le plus proche exemple d'apprentissage et la surface de séparation.

Bas d'apprentissage : $f(x)=wT\,x+b=0$

Paramètres :

w est la normale à l'hyperplan,

b est le décalage par rapport à l'origine

Les paramètres w et b ne sont pas uniques. kw et kb donnent la même surface de

séparation : $kw^T x+kb=k(w^T x+b)=0$

Quelle fonction de décision choisir : $f(x)=w^T x+b=0$

Solution : celle qui maximise la marge

Si x_s est un support vecteur, et $H=x/W^T X+b=0$ alors la marge est :

$$marge=d(x,H)=\frac{|w^T x\,s+b|}{\|W\|}$$

On impose la condition de normalisation $|w^T X_s+b|=1$ pour les vecteurs de support X_S :

$$marge=\frac{1}{\|W\|}$$

Applications de SVM

```
# Create an instance of MLPClassifier

svm = svm.SVC(kernel='rbf',C=0.1)
# Train the model
svm.fit(x_train, y_train)
```

```
SVC
SVC(C=0.1)
```

```
from prettytable import PrettyTable

x = PrettyTable(["Model", "Train SCORE", "Test SCORE"])
z=str(int(svm.score(x_train,y_train)*100))+"%"
v=str(int(svm.score(x_test,y_test)*100))+"%"
x.add_row([" SVM",z,v])
print(x)
```

```
+-------+-------------+------------+
| Model | Train SCORE | Test SCORE |
+-------+-------------+------------+
|  SVM  |     94%     |    90%     |
+-------+-------------+------------+
```

Figure 28 : Applications de SVM

Confusion Matrices SVM

	precision	recall	f1-score	support
0	0.89	0.95	0.92	61
1	0.91	0.82	0.86	39
accuracy			0.90	100
macro avg	0.90	0.89	0.89	100
weighted avg	0.90	0.90	0.90	100

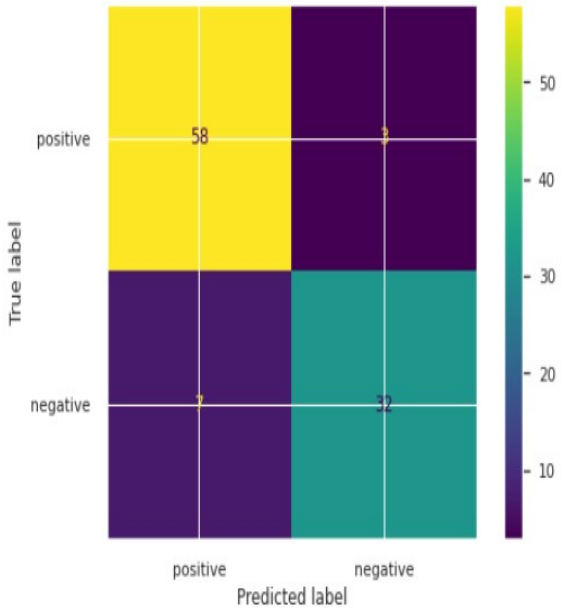
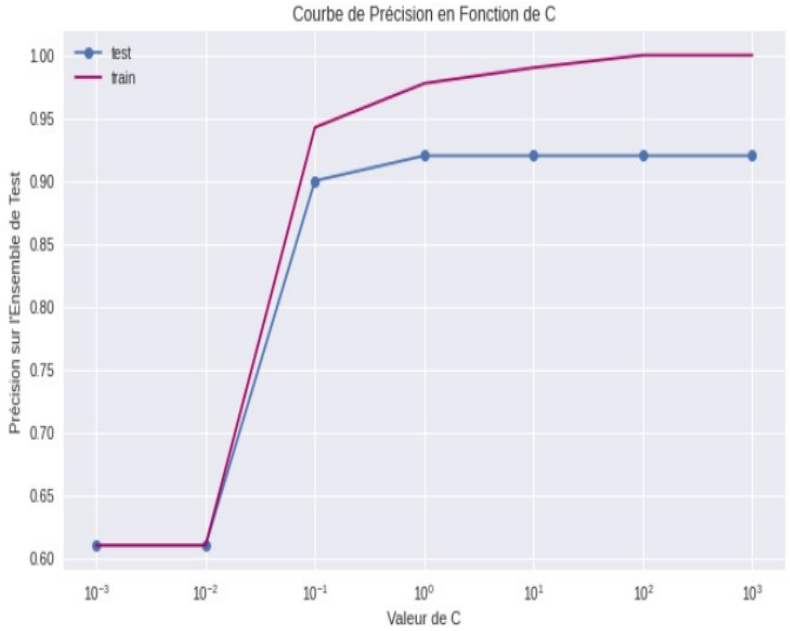


Figure 29 : Confusion Matrices SVM

Cross-validation: evaluating estimator performance




```
Meilleur paramètre C : 1.0
```

Figure 30 : Cross-validation: evaluating estimator performance

la courbe (ROC AUC)

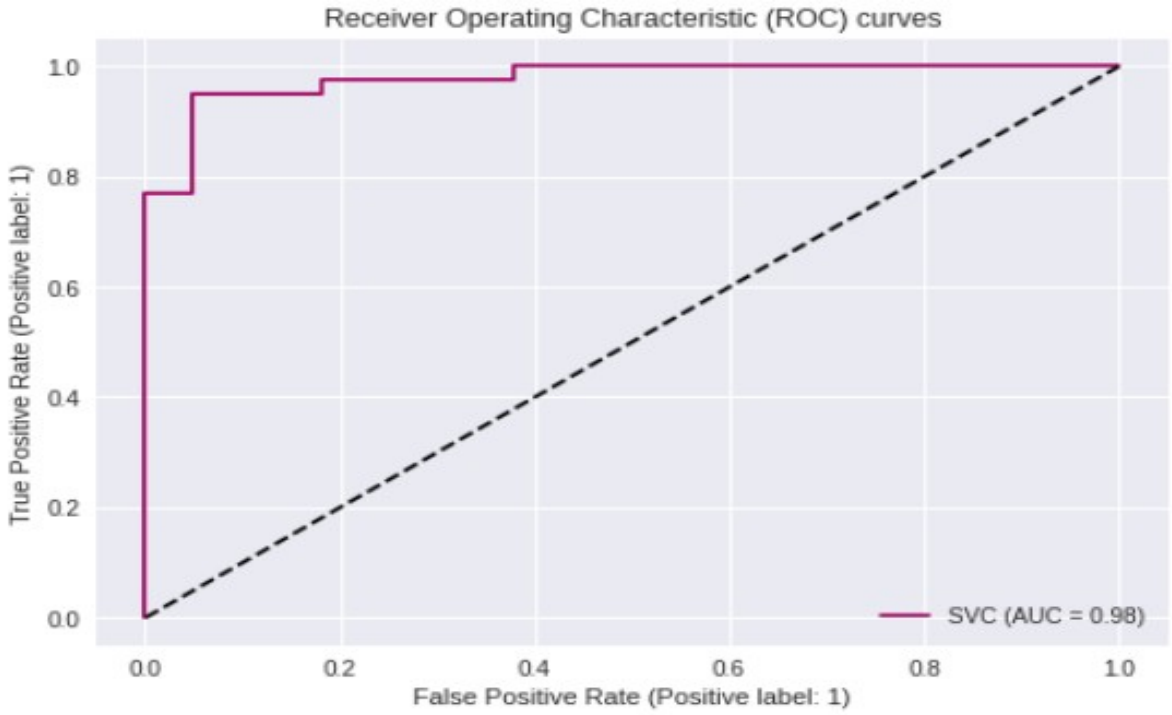


Figure 31 : la courbe (ROC AUC)

les 3 kernel

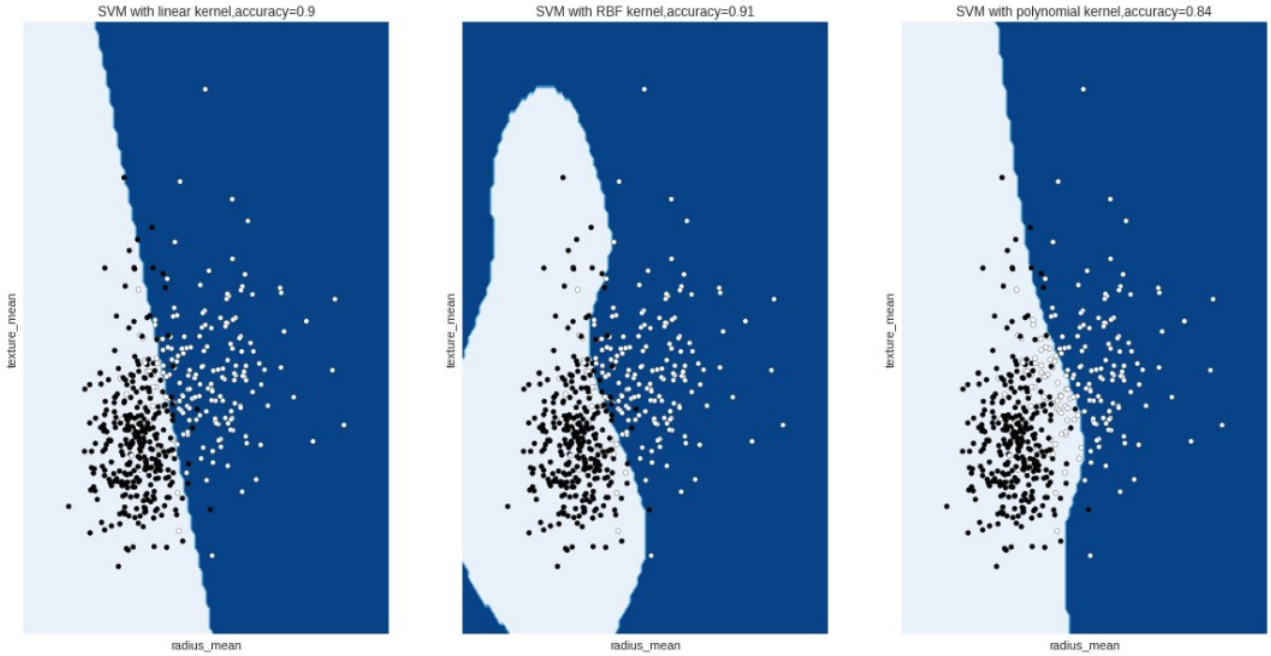
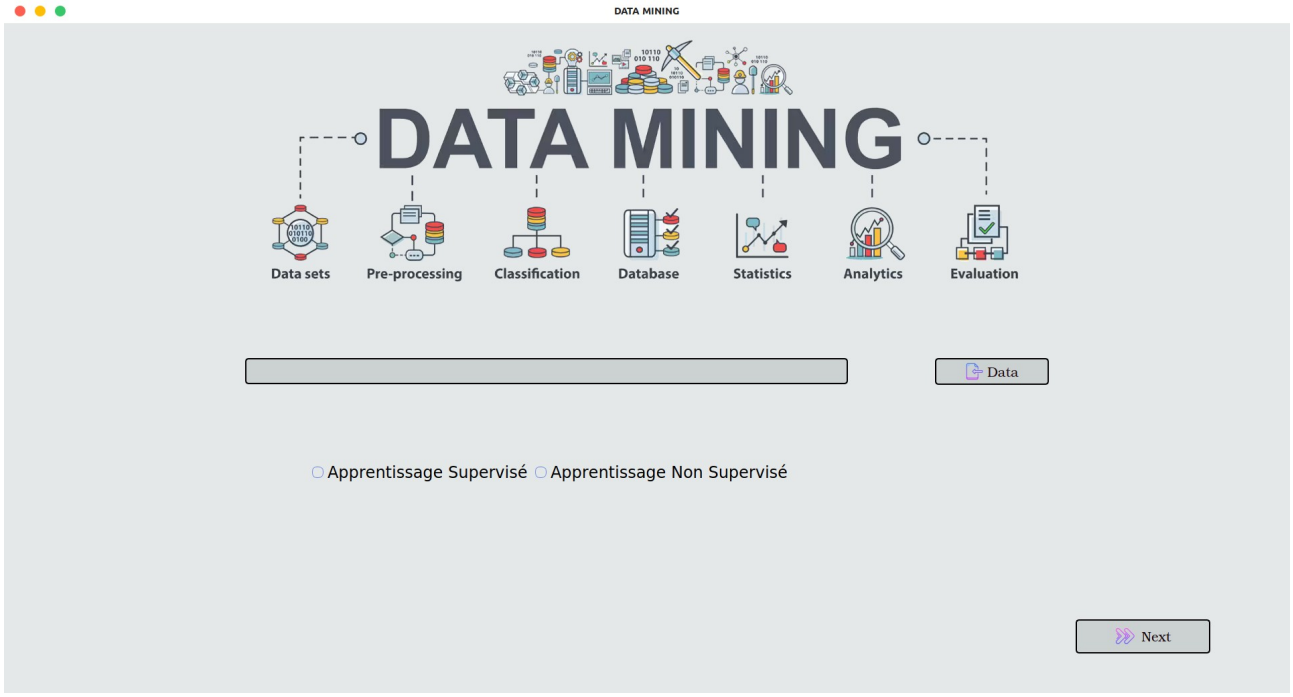
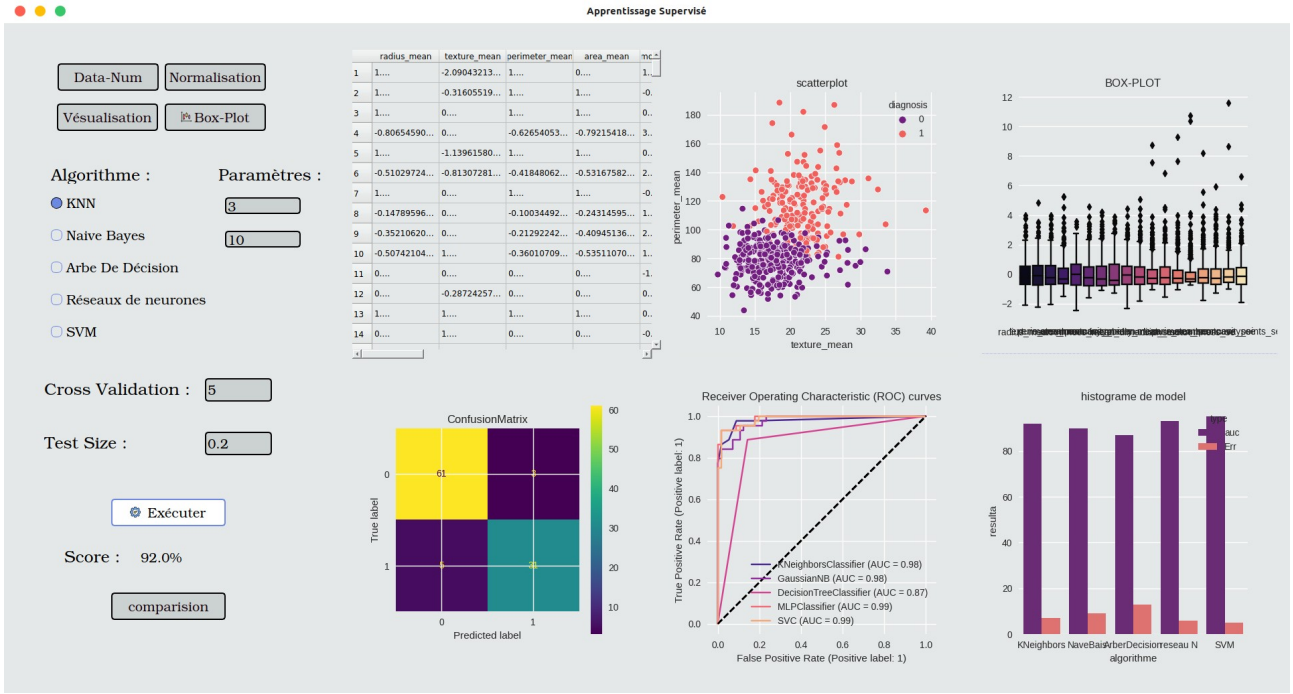


Figure 32 : les 3 kernel


DATA MINING
DATA MINING
Data sets
Pre-processing
Classification
Database
Statistics
Analytics
Evaluation
Data
Apprentissage Supervisé
Apprentissage Non Supervisé
Next


Apprentissage Supervisé
Data-Num
Normalisation
Vésualisation
Box-Plot
Algorithme :
Paramètres :
KNN
3
Naive Bayes
10
Arbe De Décision
Réseaux de neurones
SVM
Cross Validation :
5
Test Size :
0.2
Exécuter
Score :
92.0%
comparision
radius_mean
texture_mean
perimeter_mean
area_mean
scatterplot
diagnosis
texture_mean
perimeter_mean
BOX-PLOT
ConfusionMatrix
True label
Predicted label
Receiver Operating Characteristic (ROC) curves
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
KNeighborsClassifier (AUC = 0.98)
GaussianNB (AUC = 0.98)
DecisionTreeClassifier (AUC = 0.87)
MLPClassifier (AUC = 0.99)
SVC (AUC = 0.99)
histograme de model
resulta
algorithme
KNeighbors
SVM