Royaume du Maroc
Université Mohammed Premier
École Supérieure De Technologie-Oujda
Département : Informatique
Filière : LP Informatique Décisionnelle

Analyse De Données

Analyse et Classification des Emails : Détection de Spams / Non-Spams

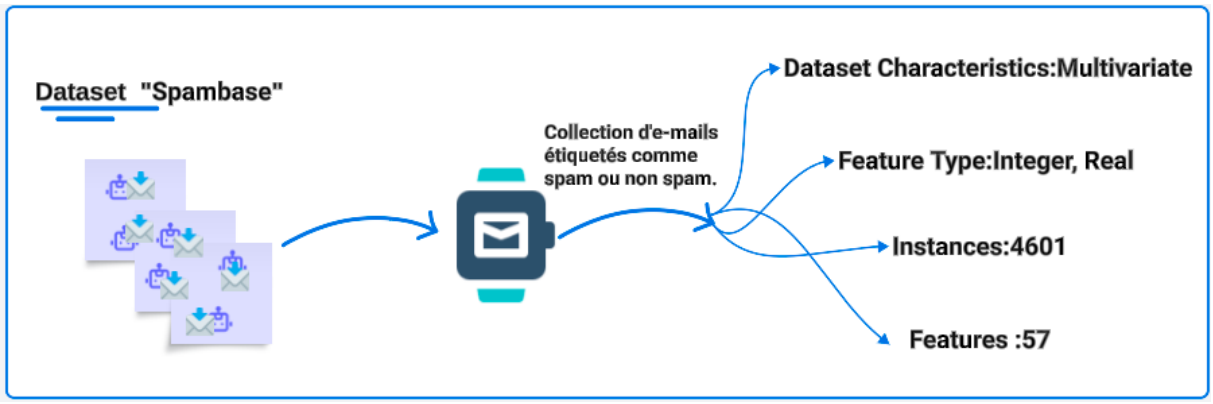
Réalisé par : **HNIOUA Abdessamad**

Encadré par : **M. Mounir Grari**

Année Universitaire : 2023/2024

La classification supervisée : est souvent un choix approprié pour la détection de spam en raison de la disponibilité des données étiquetées, de sa performance élevée, de son interprétabilité, de sa flexibilité et de son évolutivité. Cependant, il est important de choisir le bon algorithme et de bien prétraiter les données pour obtenir les meilleurs résultats possibles.

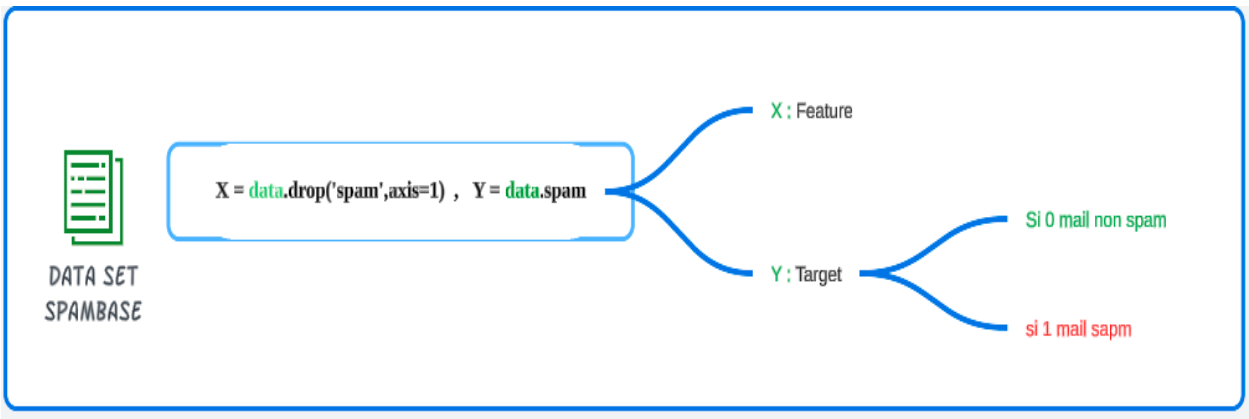
1. Dataset 'SpamBase:



Ce dataset représente un défi intéressant pour la construction de modèles de classification efficaces pour la détection de spam.

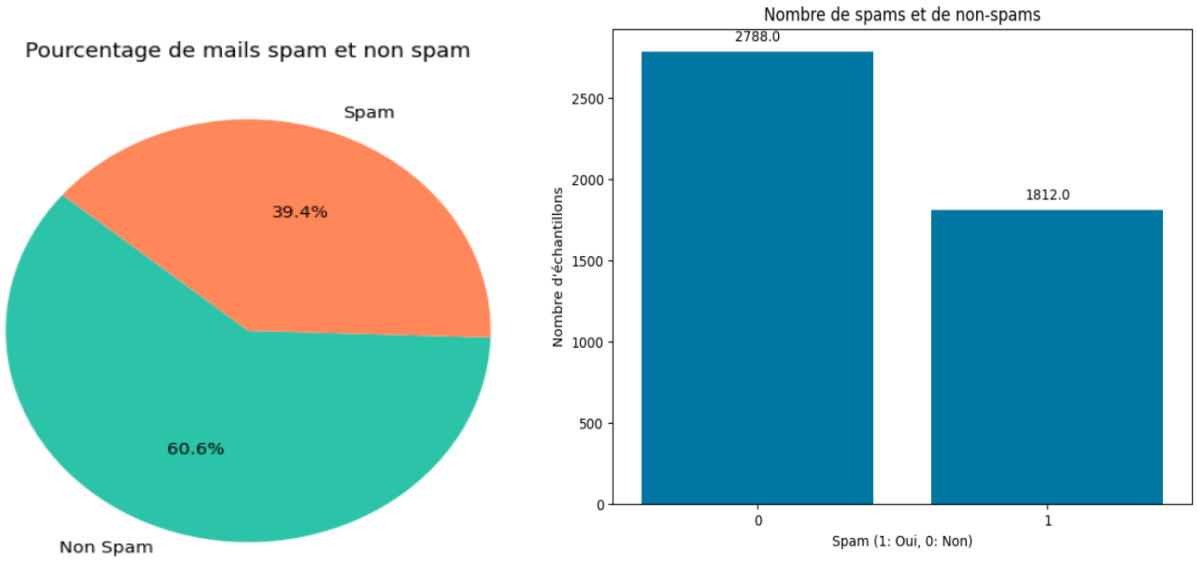
```
data.columns = ['X' + str(i) for i in range(0, len(data.columns) - 1)]
+ ['spam'] #spambase.data
```

Renommer les colonnes de vos caractéristiques en 'X1', 'X2', ..., 'X57' et la dernière colonne en 'spam'.

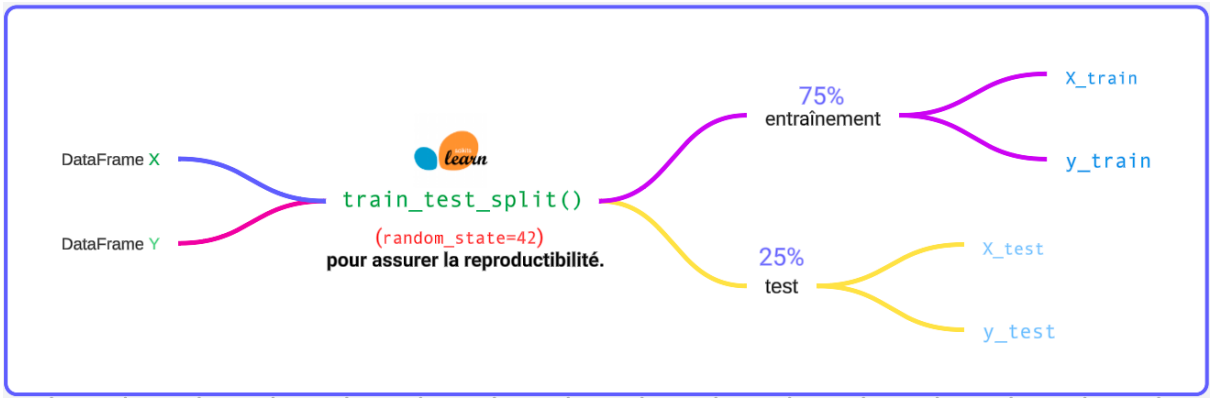


Cela crée un nouveau dataframe X qui contient uniquement les caractéristiques (facteurs).

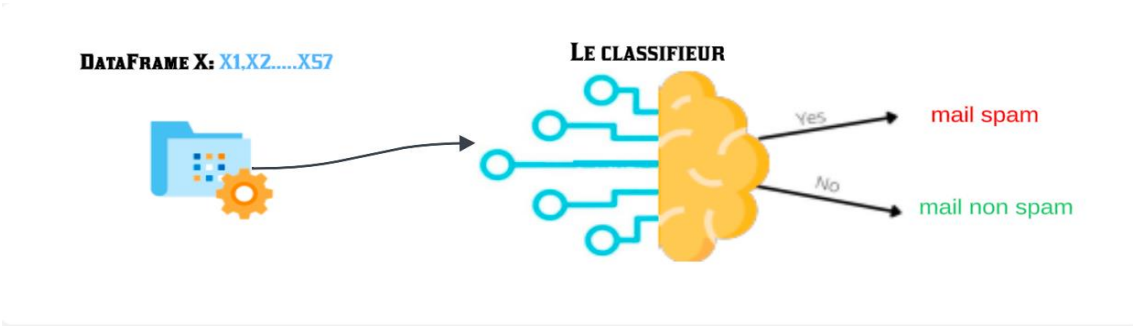
Nous attribuons la colonne 'spam' du dataframe data à la variable y, créant ainsi un dataframe y qui contient la cible.



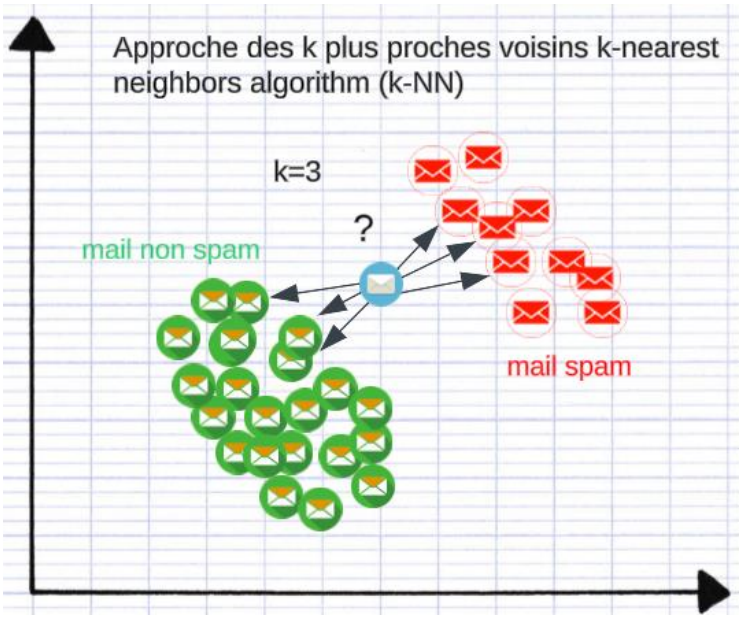
La visualisation représente un histogramme des données de spam, distinguant les échantillons de courrier électronique en fonction de s'ils sont classés comme spam (1) ou non-spam (0). La colonne des courriels non spam est plus dominante, avec une valeur de 2788 échantillons, tandis que la colonne des courriels spam présente une valeur de 1812 échantillons. Cela suggère que la majorité des échantillons de courrier électronique dans l'ensemble de données ne sont pas classés comme du spam.



Le schéma illustre le processus de division des données en ensembles d'entraînement et de test à l'aide de la fonction `train_test_split` de scikit-learn. Dans cette configuration, 75% des données sont réservées pour l'ensemble d'entraînement et 25% pour l'ensemble de test. La graine aléatoire est fixée à 42 pour assurer que la division des données est reproductible.



I. Approche des k plus proches voisins k-nearest neighbors algorithm (k-NN)



L'algorithme des k plus proches voisins (k-nearest neighbors ou k-NN) est une méthode de classification simple mais puissante. Dans le contexte d'un projet de filtrage de spam, cette approche fonctionne en calculant la similarité entre les e-mails en se basant sur des caractéristiques sélectionnées, puis en attribuant une étiquette (spam ou non-spam) en fonction du vote majoritaire parmi les k voisins les plus proches. Bien que facile à comprendre et à mettre en œuvre, il est essentiel de choisir judicieusement le nombre k et **la mesure de distance** pour obtenir des performances optimales.

1. Sélectionne le modèle d'estimateur des k plus proches voisins avec k=3 en utilisant la classe KNeighborsClassifier de scikit-learn.

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)
```

2. Entraîne le modèle des k plus proches voisins avec k=3 sur les données fournies dans les DataFrames X_train (caractéristiques) et y_train (étiquettes de classe).

```
clf.fit(X_train, y_train)
```

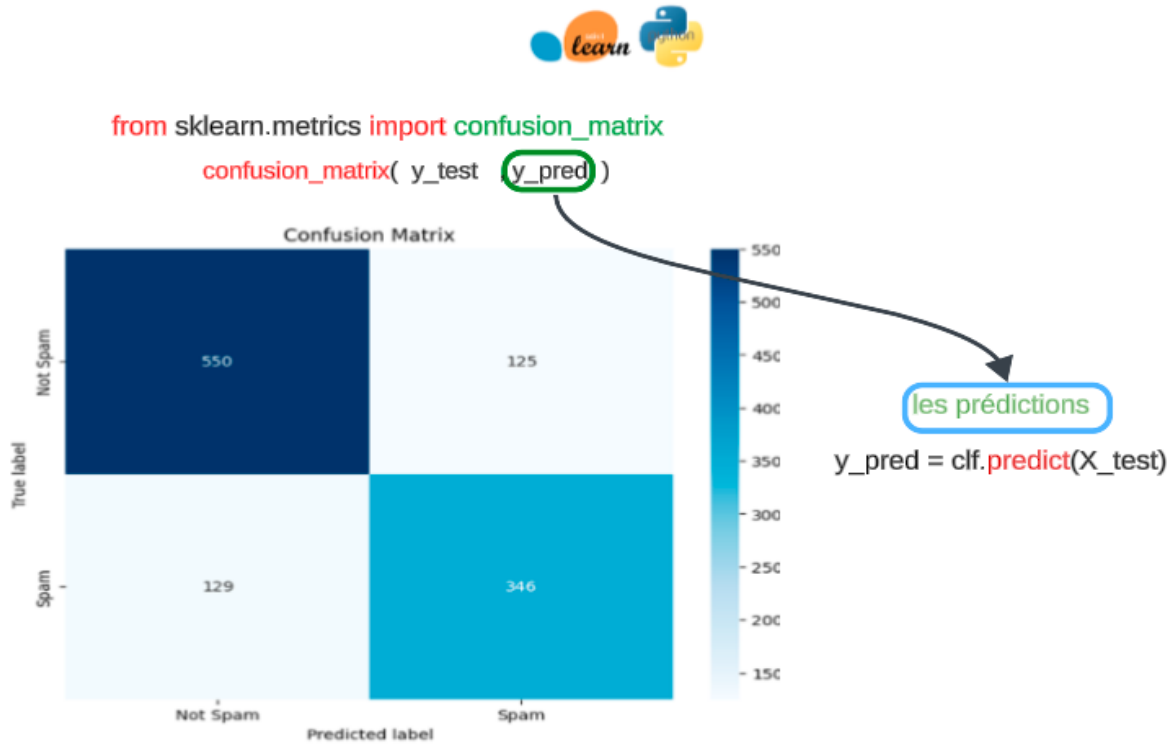
3. Le modèle a été évalué sur les données d'entraînement, obtenant un score de 90% de précision. Cela signifie que le modèle prédit correctement la classe des échantillons dans 90% des cas.

```
print(clf.score(X_train, y_train))
```

4. Utilise le modèle entraîné pour effectuer des prédictions sur les données de test X_test, stockant les prédictions dans la variable pred.

```
pred = clf.predict(X_test)
```

La matrice de confusion : résume la performance d'un modèle de classification en révélant ses prédictions correctes et incorrectes par rapport aux vraies étiquettes de classe, essentiellement en identifiant les vrais positifs, les vrais négatifs, les faux positifs et les faux négatifs.



L'accuracy évalue la proportion de prédictions correctes du modèle de classification sur l'ensemble des échantillons, ce qui est essentiel pour mesurer sa capacité à distinguer efficacement les mails spam des non-spam.

GridSearchCV de scikit-learn pour rechercher la meilleure valeur du paramètre n_neighbors (nombre de voisins) pour le modèle kNN. Il configure une grille de valeurs de n_neighbors à tester, puis utilise GridSearchCV pour évaluer les performances du modèle kNN avec chaque valeur de n_neighbors et sélectionner celle qui donne les meilleures performances sur les données d'entraînement à l'aide d'une validation croisée. Enfin, il renvoie les paramètres du modèle kNN optimal.

La partie d'optimisation par GridSearchCV a permis de déterminer que le meilleur nombre de voisins à utiliser dans l'algorithme kNN est 5, en se basant sur les performances du modèle évalué sur les données d'entraînement à l'aide d'une validation croisée. Cette démarche a abouti à la conclusion que le modèle kNN avec 5 voisins offre la meilleure performance parmi les valeurs de `n_neighbors` testées, ce qui permet d'optimiser la précision du modèle pour la classification des données.

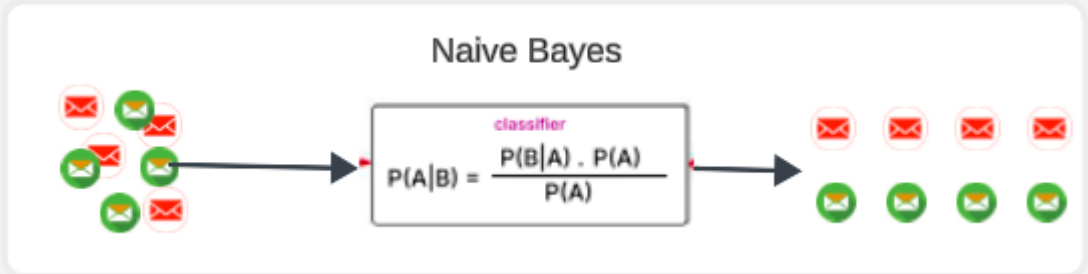
```
y_pred = best_knn.predict(X_test)
from sklearn.metrics import accuracy_score
# Calcul de l'accuracy
accuracy = accuracy_score(y_pred , y_test)

# Affichage de l'accuracy
print("Accuracy:", accuracy*100,"%")
```

Après l'exécution de l'algorithme d'optimisation GridSearchCV, l'accuracy du modèle a légèrement augmenté, passant de 76.52% à 76.78%. Cette amélioration peut être attribuée à la recherche du meilleur paramètre `n_neighbors` effectuée par GridSearchCV, ce qui a permis de sélectionner une configuration plus adaptée du modèle kNN pour les données d'entraînement.

II. Naive Bayes

Les classificateurs **Naive Bayes** sont une famille de classificateurs probabilistes simples basés sur l'application du théorème de Bayes, avec une hypothèse d'indépendance parmi les prédicteurs.



1. Sélectionne le modèle d'estimateur Naive bayes en utilisant la classe aussianNB de scikit-learn.

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB().fit(X_train, y_train)
```

2. Entraîne le modèle sur les données fournies dans les DataFrames X_train (caractéristiques) et y_train (étiquettes de classe).

```
nb.fit(X_train, y_train)
```

3. Le modèle a été évalué sur les données d'entraînement, obtenant un score de 83% de précision.

```
print(nb.score(X_test, y_test))
```

4. Utilise le modèle entraîné pour effectuer des prédictions sur les données de test X_test, stockant les prédictions dans la variable pred.

```
pred = nb.predict(X_test)
```

Obtenant dans la prédictions un score de 81% de précision.