UNIVERSITE IBN ZOHR FACULTE

DES SCIENCES D’AGADIR

**Centre d’Excellence**

**Filière : Ingénierie informatique et système embarqué**

**Rapport Project Machine Learning**

**Groupe de TP : 3**

**Réaliser par :**

**OUBOURHAIL CHAYMA**

**ZGAR LATIFA**

**Classification du Cancer du Sein à l'aide d'Algorithmes de Machine Learning**

**Résumé**

Ce projet porte sur l’utilisation d’algorithmes de machine learning pour prédire la nature d’une tumeur mammaire à partir de données cliniques. Le dataset utilisé est le Breast Cancer Wisconsin (Diagnostic) dataset. L’objectif est de développer et comparer plusieurs modèles de classification pour déterminer le plus performant dans la distinction entre tumeurs bénignes et malignes.

**1. Introduction**

Le cancer du sein est l’une des principales causes de mortalité chez les femmes. La précision du diagnostic est donc essentielle pour garantir une prise en charge rapide et adaptée. Le machine learning peut apporter un soutien déterminant dans la prédiction des diagnostics médicaux. Ce projet vise à évaluer plusieurs modèles de classification appliqués aux données de biopsies de tumeurs mammaires.

**2. Problématique**

Face à un volume croissant de données médicales, les méthodes classiques de diagnostic montrent leurs limites. Comment prédire efficacement si une tumeur est bénigne ou maligne à partir de mesures cliniques extraites de biopsies ?

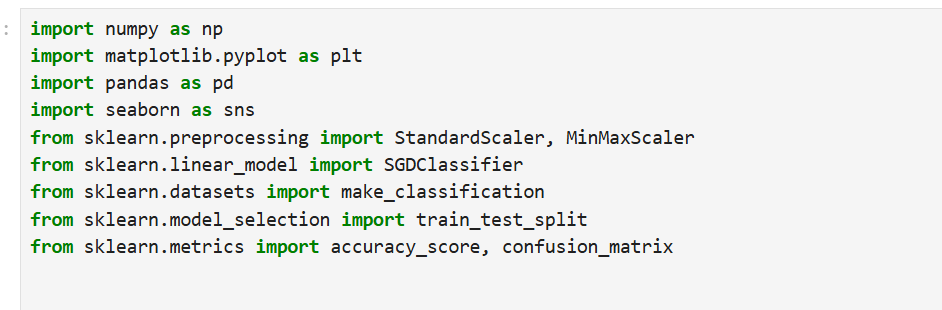
**3. Description du Dataset**

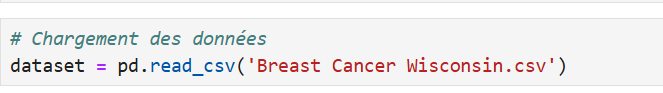
* **Nom :** Breast Cancer Wisconsin (Diagnostic)
* **Taille :** 569 exemples
* **Variables :** 30 caractéristiques (mesures statistiques des noyaux cellulaires) avec un cible ('diagnosis')
* **Classes :** M (maligne), B (bénigne)

**4. Les modèles utilisées :**

**La régression Logistique**

*isconsin* pour prédire si une tumeur est maligne (M) ou bénigne (B).La régression logistique est un algorithme de classification supervisée utilisé pour prédire des variables catégorielles binaires Dans ce rapport, nous appliquons une régression logistique via SGDClassifier sur le dataset *Breast Cancer W*

****

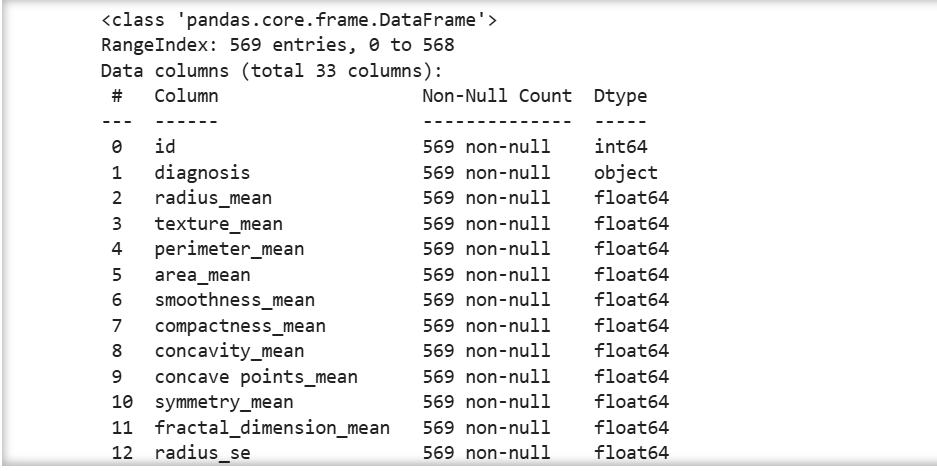
****

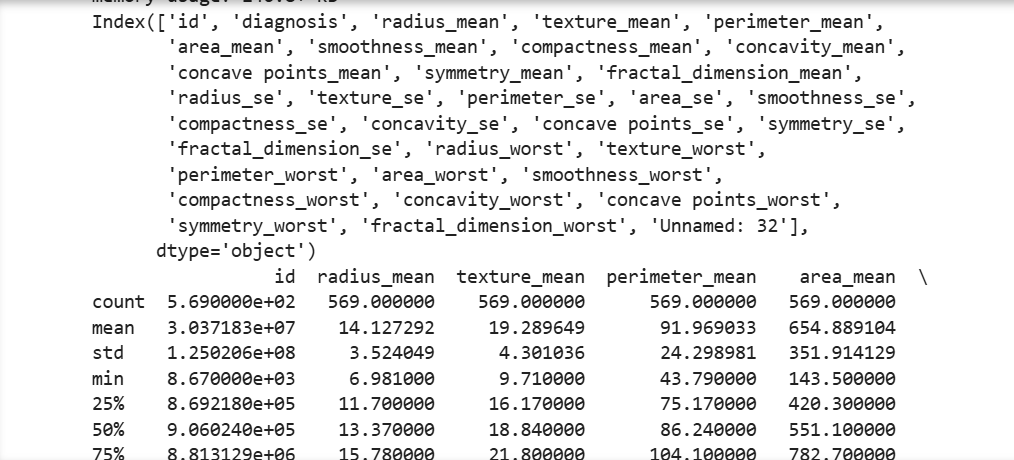
La commande permet de charge les données depuis un fichier CSV.

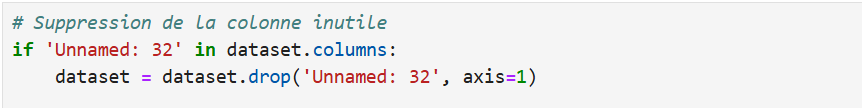
****

Ces commandes permettent d'explorer rapidement la structure et le contenu du dataset. Et chaque commande permet d’affiche les informations suivants :

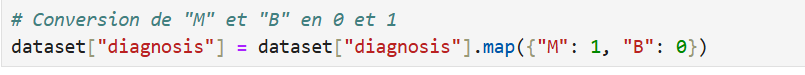
* les types de colonnes,
* les noms de colonnes,
* des statistiques descriptives,
* les dimensions (lignes,colonnes),
* 5 exemples aléatoires.

****

****

****

Supprime une colonne vide si elle existe (Unnamed: 32 souvent générée automatiquement).

****

Et la commande suivante permet d’ Encoder la variable cible :

* **M (malin) → 1**
* **B (bénin) → 0**

****

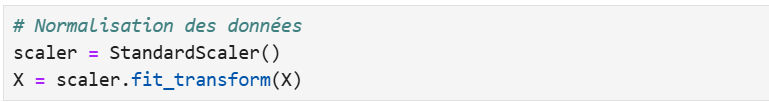
La commande Affiche le nombre de valeurs manquantes par colonne.

****

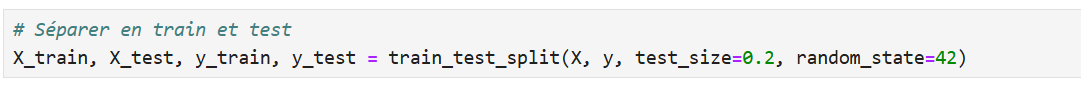
* Sélectionne toutes les colonnes à partir de l'indice 3
* Values convertit en tableau NumPy.

y = dataset["diagnosis"].values

Récupère la colonne diagnosis et la convertit en tableau NumPy.

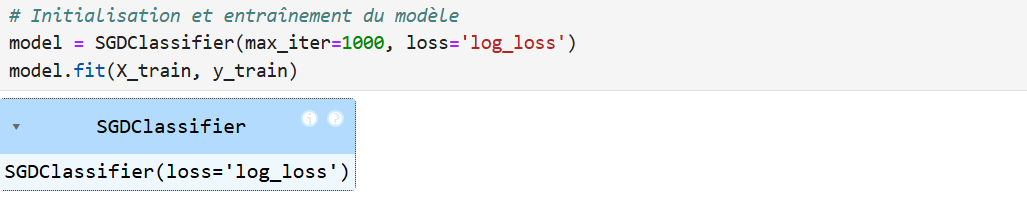
****

On normalise les données pour améliorer la convergence du modèle. Chaque variable est centrée réduite (moyenne 0, écart-type 1).****

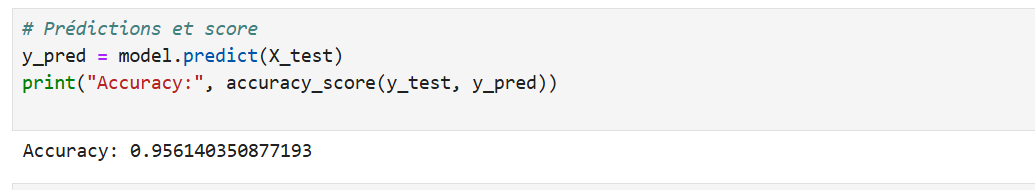
On aplatit y pour obtenir un vecteur à une dimension, ce qui est nécessaire pour certains modèles.****

On divise le dataset en deux :

* 80 % pour entraîner le modèle (X\_train, y\_train)
* 20 % pour l’évaluer (X\_test, y\_test)

****

On initialise un classificateur utilisant la régression logistique (fonction de perte log\_loss), entraîné via descente de gradient stochastique, puis on l’entraîne sur les données.

****

On utilise la méthode .predict() pour générer des prédictions à partir du modèle entraîné sur les données de test.

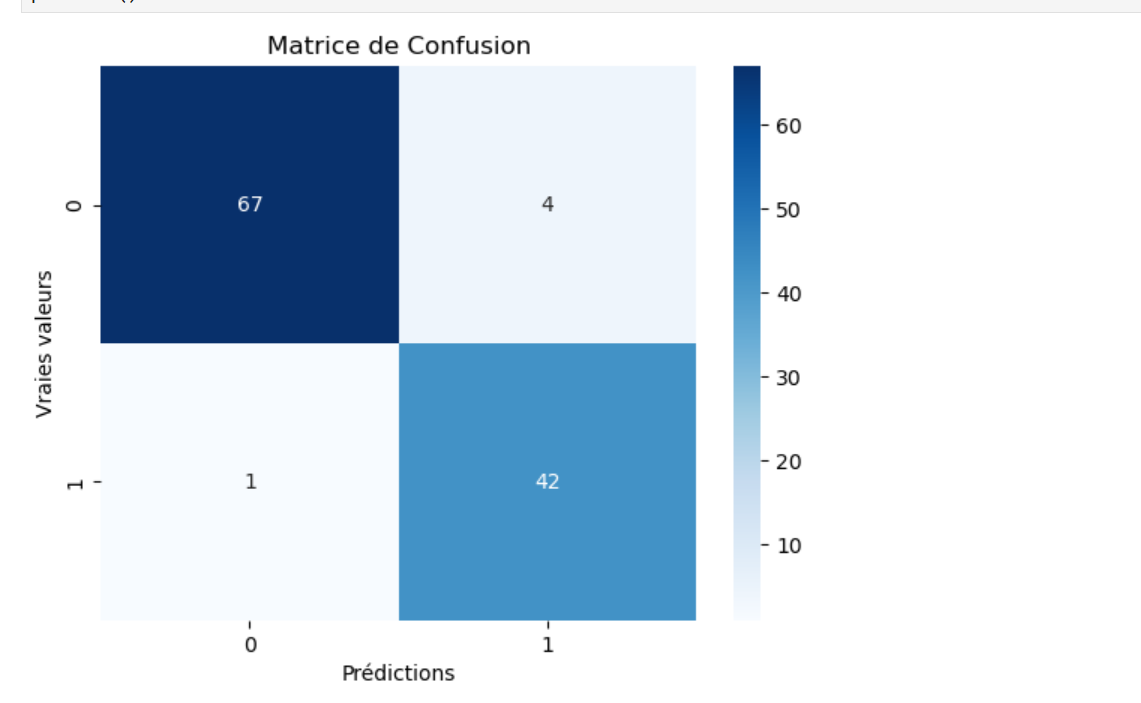
Ensuite, on mesure la précision (accuracy) du modèle, c’est-à-dire la proportion de bonnes prédictions parmi l’ensemble des prédictions réalisées.

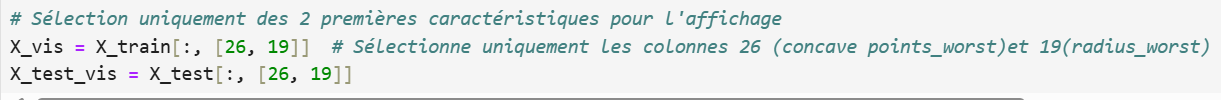
Accuracy = 0.956140350877193

Cela signifie que le modèle a correctement prédit environ 95,61 % des cas dans les données de test.

****

On génère une matrice de confusion pour visualiser les performances du modèle :

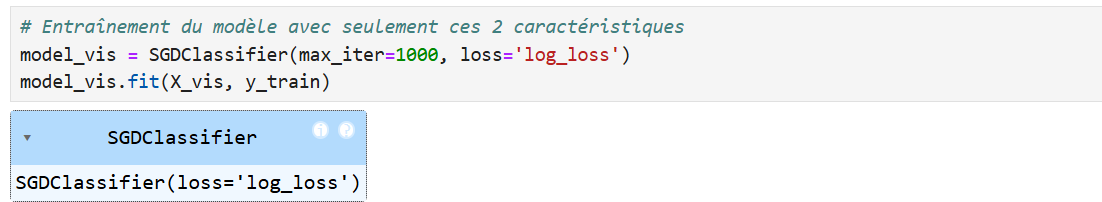
****

****

On sélectionne deux variables très discriminantes :

concave points\_worst (index 26)

radius\_worst (index 19)  
Cela nous permet de visualiser la frontière de décision en 2D.

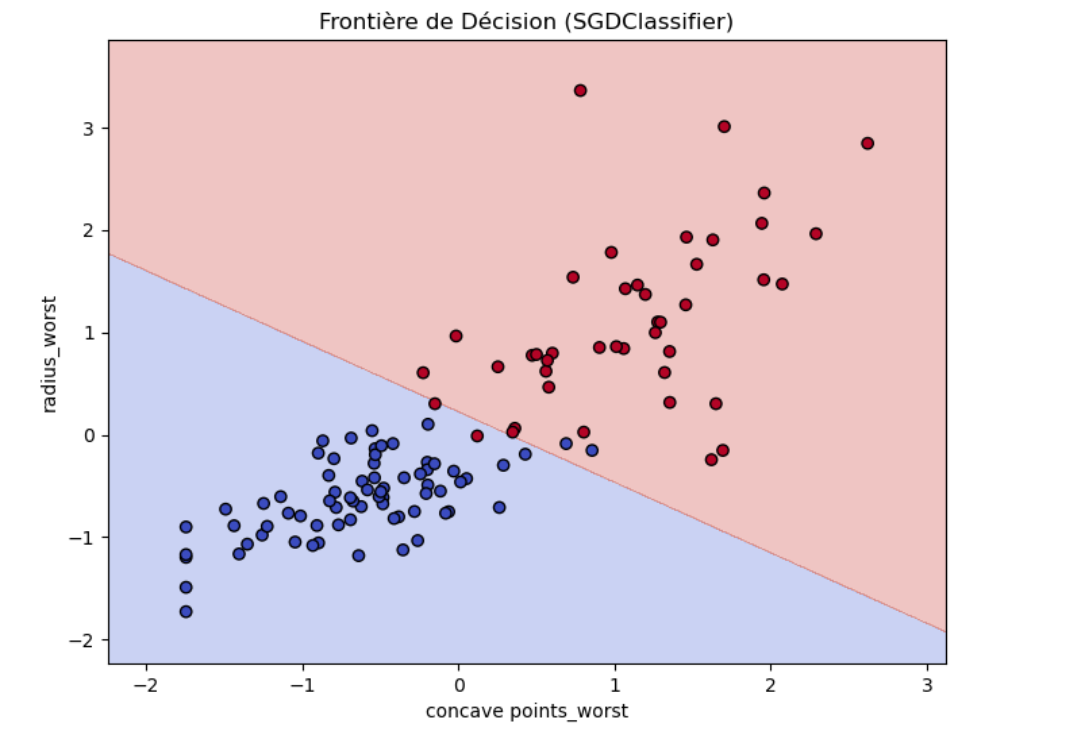
****

On réentraîne un modèle en utilisant uniquement ces deux variables.

****

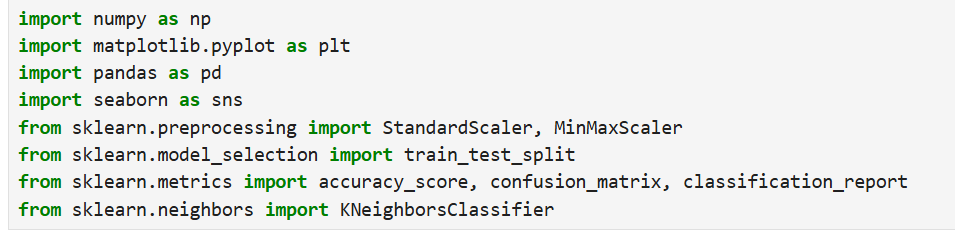
Dans le cadre de notre étude sur la classification des tumeurs cancéreuses,nous avons implémenté une fonction plot\_decision\_boundary() pour visualiser comment notre modèle sépare les cas bénins des cas malins.

* On génère des milliers de points espacés de 0.01 unité pour couvrir tout le graphique.
* Le modèle prédit la classe (0 ou 1) pour chaque point de la grille.
* contourf() : Colorie les zones selon la classe prédite (bleu pour Bénin, orange pour Malin).
* scatter() : Superpose les **vraies données** (points) pour comparer avec les prédictions.
* Et le résultats montre Les points bleus (vrais Bénins) sont dans la zone bleue.Et les points orange (vrais Malins) sont dans la zone orange.

****

**K plus proches voisins (KNN)**

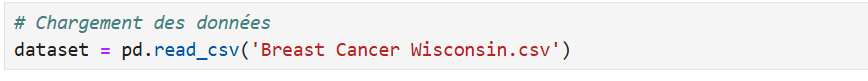
l’algorithme KNN (K-Nearest Neighbors) est un modèle de classification qu’on utilise pour prédire si une tumeur est bénigne ou maligne.  
Le principe est simple : on compare une nouvelle tumeur aux K tumeurs les plus proches d’elle dans le dataset (en se basant sur des caractéristiques.  
Ensuite, on regarde la majorité des classes parmi ces voisins : si la majorité sont malignes, le modèle prédit « maligne » ; sinon, il prédit « bénigne ».

****

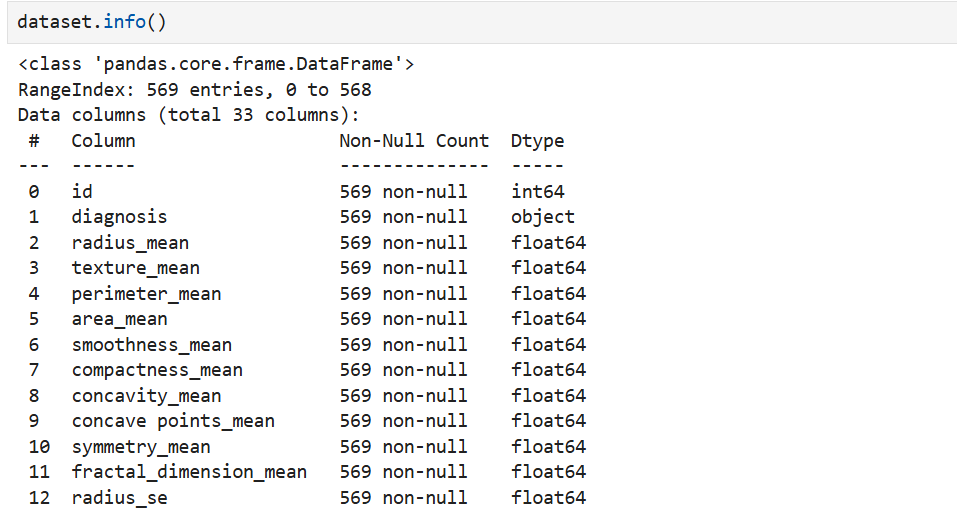
* On importe numpy pour les opérations mathématiques.
* matplotlib.pyplot et seaborn servent à afficher des graphiques (comme la matrice de confusion ou les erreurs).
* pandas permet de lire et manipuler le dataset.
* On utilise StandardScaler et MinMaxScaler pour normaliser les données.

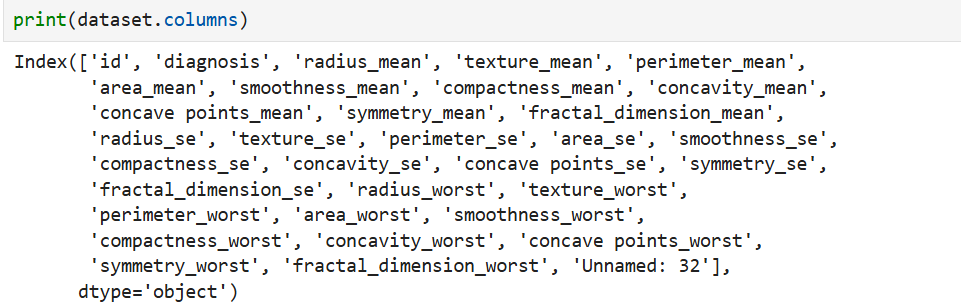
train\_test\_split permet de séparer les données en train/test.

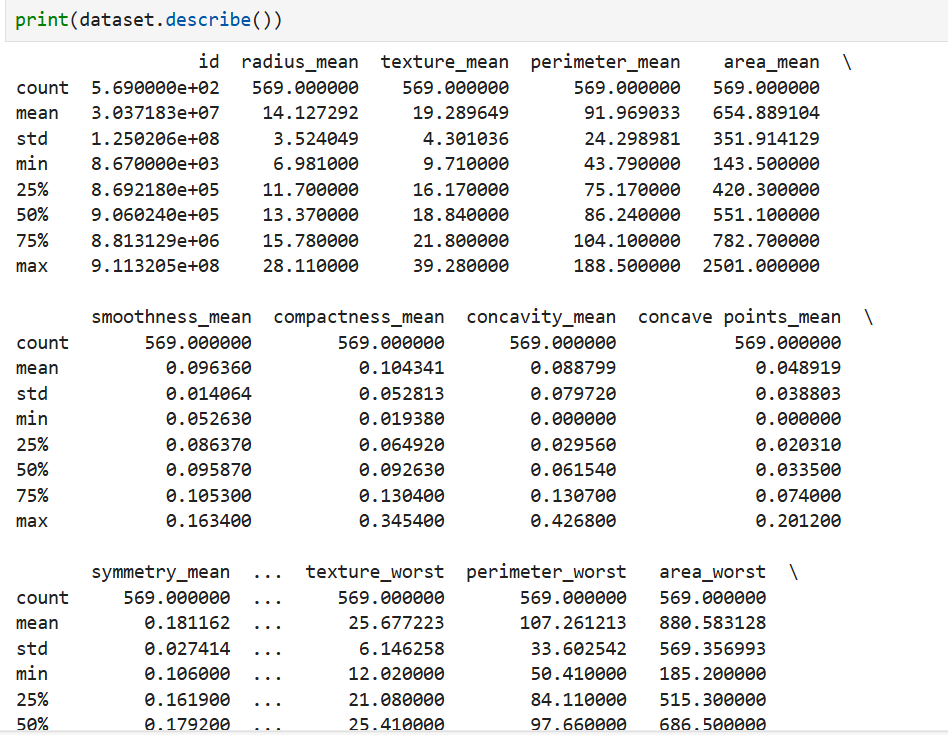
* accuracy\_score, confusion\_matrix, et classification\_report permettent d’évaluer les performances du modèle.
* Enfin, KNeighborsClassifier est le modèle KNN qu'on va entraîner.

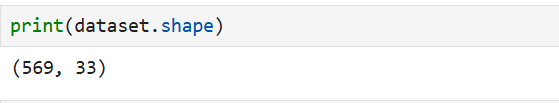
****

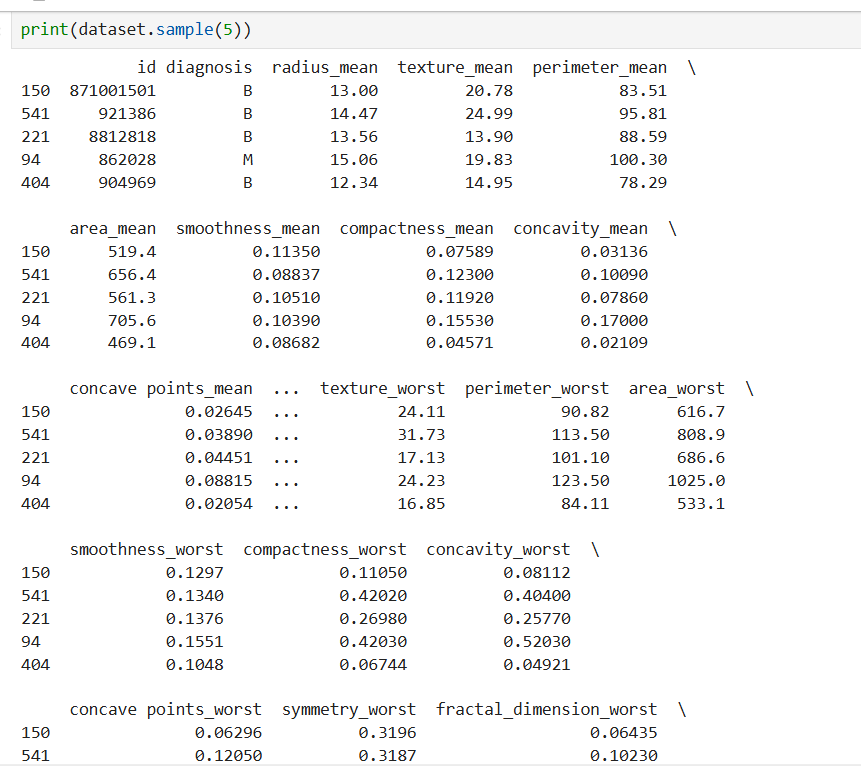
On charge le fichier CSV contenant les données du cancer du sein dans un DataFrame nommé dataset.

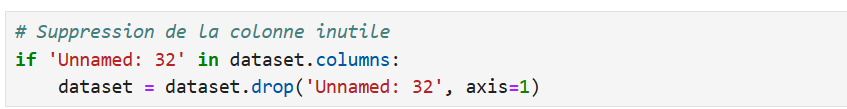




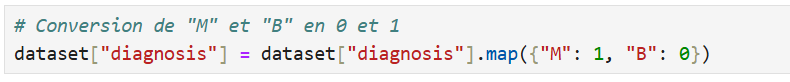








On supprime la colonne "Unnamed: 32" si elle est présente, car c’est une colonne vide inutile.



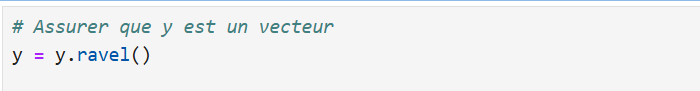
On remplace "M" (maligne) par 1 et "B" (bénigne) par 0 pour rendre les classes **numériques**, ce qui est nécessaire pour l’algorithme.



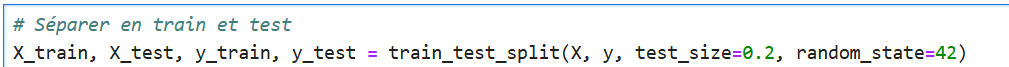
On vérifie qu’il n’y a **pas de valeurs manquantes** dans le dataset.Et la sortie indique que **toutes les colonnes contiennent 0 valeurs manquantes**.



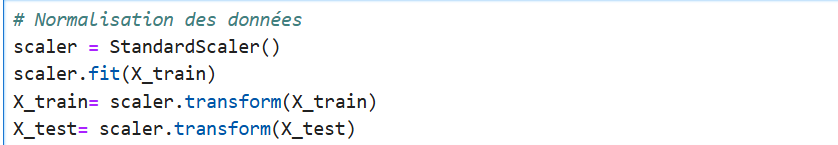
* On sélectionne les **caractéristiques (features)** à partir de la colonne 3 jusqu’à la fin (en excluant l’ID et le nom).
* y contient la **variable cible** (0 ou 1).



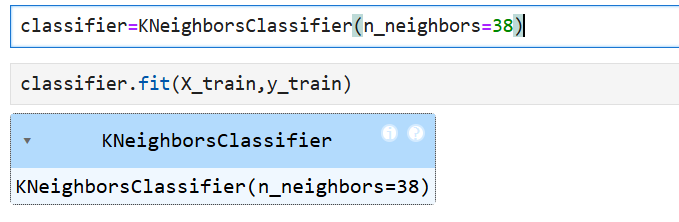
ravel() transforme y en un tableau 1D (nécessaire pour l'entraînement).



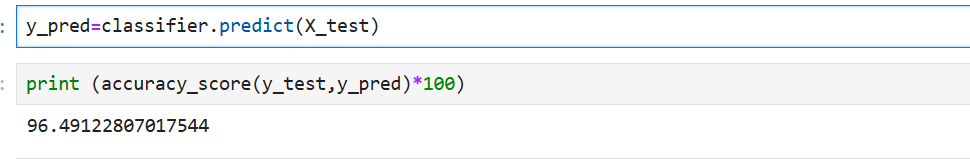
* On divise les données en **80 % pour l’entraînement** et **20 % pour le test**.
* random\_state=42 permet d’avoir des résultats reproductibles.



Avant de normaliser les données avec StandardScaler, on doit d'abord entraîner le scaler en appelant .fit() sur les données d'entraînement. Ensuite, on peut appliquer .transform() pour normaliser les valeurs. Cette étape est imporante pour éviter l’erreur NotFittedError.



* On crée un modèle KNN avec **38 voisins** .
* On entraîne le modèle avec les données d’entraînement.

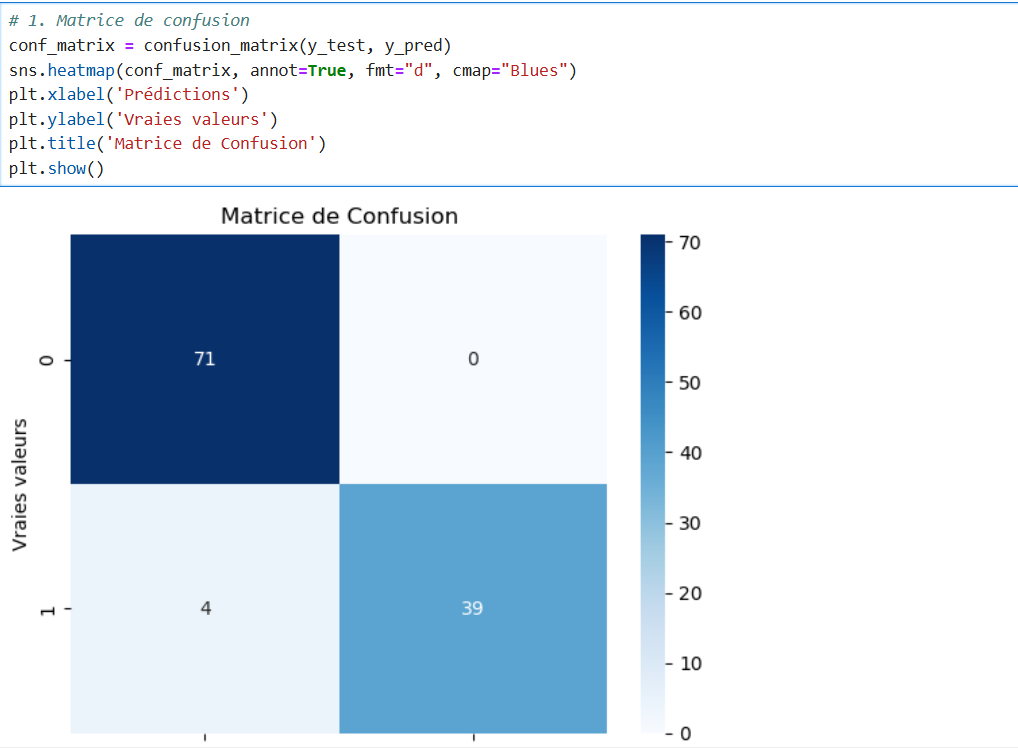


* Dans la première ligne, on utilise notre **modèle KNN** qui estdéjà entraîné sur l’ensemble X\_train avec classifier.fit pour faire des **prédictions** sur l'ensemble de test X\_test.
* Avec classifier.predict c’est ici qu’on demande au modèle de **deviner** à quelle classe chaque observation de X\_test appartient cancer malin ou bénin.
* Le résultat est un **vecteur y\_pred** contenant les prédictions du modèle pour chaque échantillon.

Dans la deuxième ligne on évalue à quel point les prédictions du modèle sont correctes en comparant :

* y\_test : les vraies classes de l’ensemble de test,
* y\_pred : les classes prédites par le modèle.

Et la sortie de la commande 96.49 % signifie que le modèle a bien prédit le diagnostic pour environ **96 patients sur 100**.



Interprétation des Résultats

**1.Vrais Négatifs (TN = 71)**:

* 71 tumeurs **bénignes** correctement identifiées comme bénignes
* Cas où le modèle dit sain et le patient est vraiment sain

**2.Faux Positifs (FP = 0)**:

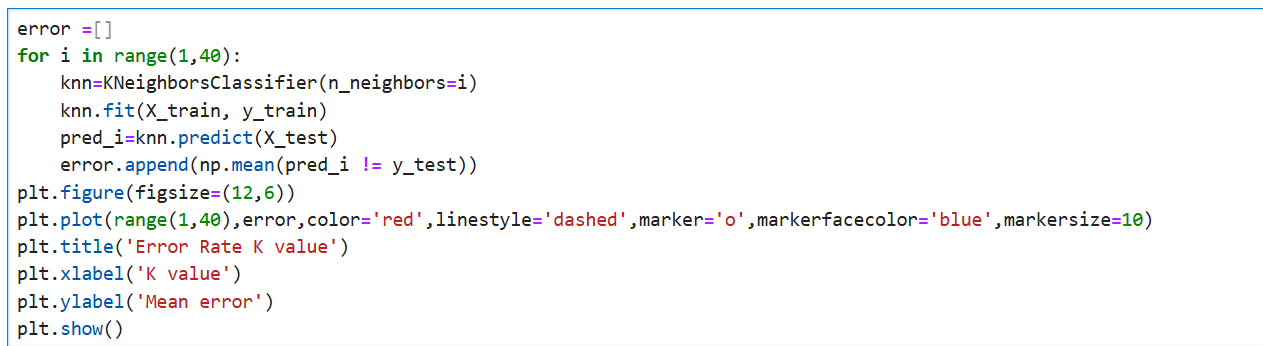
* **Aucun** cas où une tumeur bénigne a été faussement classée comme maligne
* Très important car éviter les faux positifs prévient les traitements inutiles et le stress chez les patients

**3.Faux Négatifs (FN = 4)**:

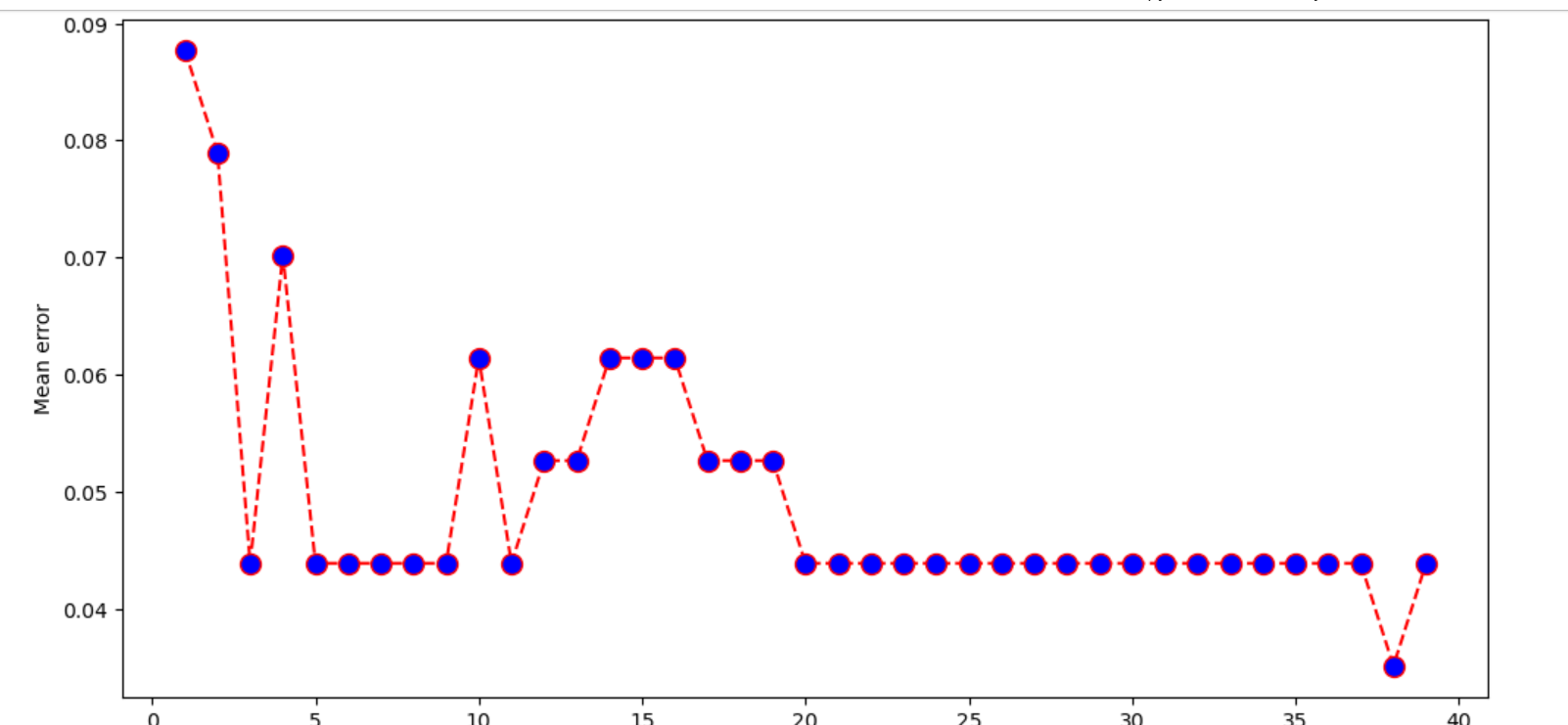
* 4 tumeurs **malignes** incorrectement classées comme bénignes
* Cas les plus dangereux en oncologie (faux sentiment de sécurité) car pourrait retarder des traitements nécessaires

**4.Vrais Positifs (TP = 39)**:

* 39 tumeurs malignes correctement détectées



Cette boucle for permet de Tester différentes valeurs de K entre 1 et 39 pour voir laquelle est la meilleure.



La courbe montre une **relation**  entre le taux d'erreur et la valeur de K avec :

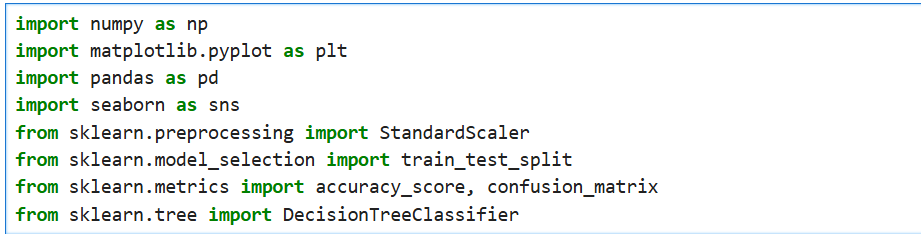
* **Erreur élevée** pour les petites valeurs de K (K=1 à 20)
* **Minimum d'erreur** autour de K>20

**Donc pour choisir un K fiable il faut le choisi dans l’intervalle qui est entre 35 et 40.**

**Classification du cancer du sein avec arbre de décision**

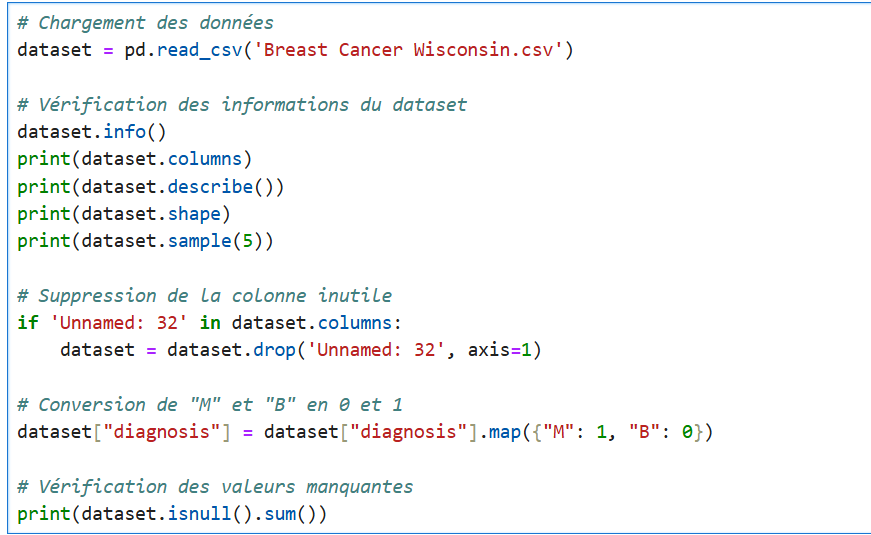
**Dans notre projet**, un **arbre de décision** est un modèle d’apprentissage supervisé qu’on utilise pour **classer automatiquement une tumeur**   
Il prend en compte différentes caractéristiques (comme la taille, la texture ou la concavité des cellules).

Grâce à sa structure simple en forme d’arbre (avec des nœuds, branches et feuilles), il permet de **prendre des décisions rapides** et **faciles à interpréter**, ce qui est utile pour **mieux comprendre les critères qui influencent le diagnostic**.

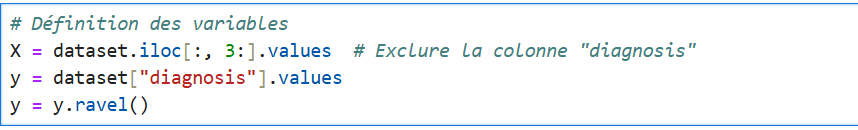


On a jouter la bibliothèque DecisionTreeClassifier.

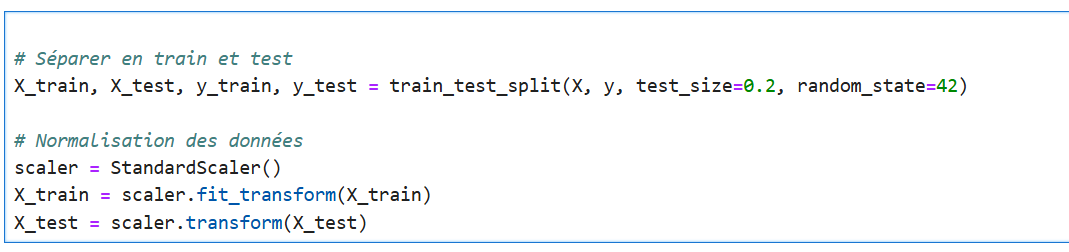
Le chargement et l’exploration du jeu de données :



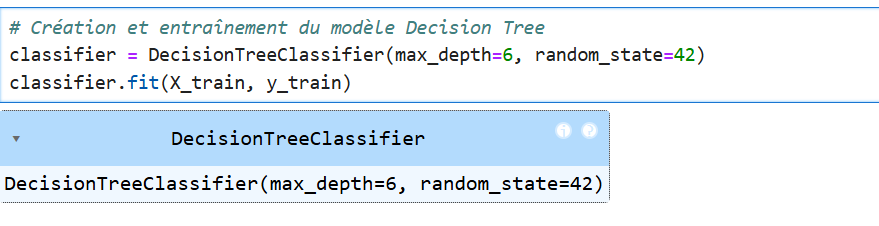
Définition des variables :



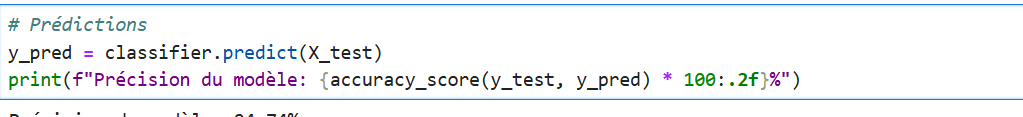
Séparation du dataset en donné de test et d’etrain et la normalisation des données :



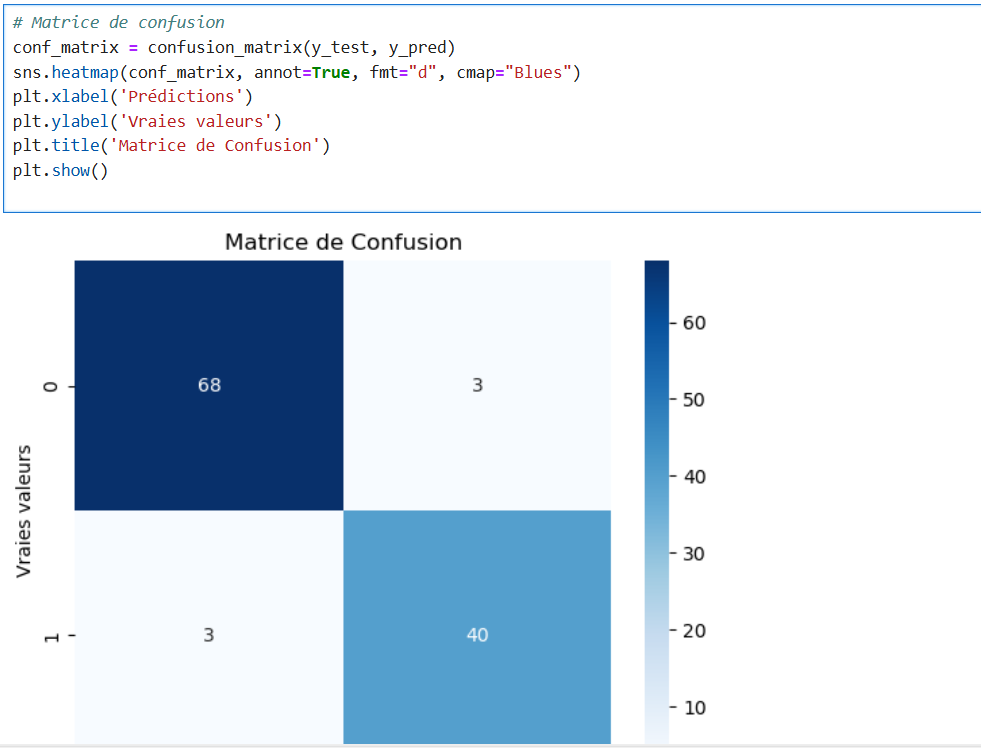
On a crée un **modèle d’arbre de décision**, avec une profondeur maximale de 6.  
On l’entraîne (fit) avec les données normalisées d’entraînement.



Prédiction sur les données de test



Matrice de confusion :



Explication de la matrice de confusion :

**Interprétation Clinique**

* **Vrais Négatifs (68)** :

**68 patients** étaient **sains** et le modèle l'a correctement identifié.

**Exemple** : Une biopsie confirme l'absence de cancer, et le modèle est d'accord.

* **Faux Positifs (3)** :

**3 patients** étaient **sains**, mais le modèle a **faussement suspecté un cancer**.

**Impact** :

Examens supplémentaires inutiles (IRM, biopsie).

Stress pour le patient.

* **Faux Négatifs (3)** :

**3 patients** avaient **un cancer**, mais le modèle les a **ratés**.

**Problème grave** :

Retard de traitement.

Risque de progression de la maladie.

* **Vrais Positifs (40)** :

**40 patients** avaient **un cancer**, et le modèle l'a correctement détecté.

**Bonne nouvelle** : Diagnostic précoce → traitement rapide.





Conclusion :

L'utilisation d'algorithmes de machine learning pour prédire la nature d'une tumeur mammaire à partir de données cliniques, comme le démontre l'analyse du Breast Cancer Wisconsin (Diagnostic) dataset, met en évidence l'efficacité de ces modèles dans l'aide au diagnostic médical. Parmi les modèles testés—régression logistique, K plus proches voisins (KNN) et arbre de décision—le KNN a obtenu la meilleure précision avec un taux d'accuracy de 96.49 %, ce qui suggère une capacité supérieure à différencier les tumeurs bénignes des tumeurs malignes. Ces résultats soulignent le potentiel du machine learning pour optimiser les outils de diagnostic, mais il reste essentiel de poursuivre l'amélioration des modèles, notamment en matière de robustesse et de généralisation, afin de garantir leur applicabilité dans des contextes cliniques réels