République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tunis EL Manar

Rapport de Projet soumis dans le cadre du projet Fouille De Données

Réalisé par

Mustapha Zouari
Amine Miladi

Réalisation d'un classifieur à base de réseau de neurones (MLPClassifier de sklearn).

Introduction

L'objectif de ce projet est de créer et entraîner un classifieur à base de réseau de neurones pour le jeu de données BrestCancer.

Ce jeu de données contient des mesures de caractéristiques de cellules prélevées sur des patients atteints ou non d'un cancer du sein.
L'objectif de la tâche est de prédire si les cellules sont malignes ou bénignes en utilisant les caractéristiques mesurées.

Table des matières

Introduction **1**

Table des matières **2**

Table des figures **3**

1 Données étudiées **4**
1.1 Description des données 4
1.1.1 Qu'est ce que Breast Cancer dataset ? 4
1.1.2 Attributs Breast Cancer 5
1.2 Exploration des Données 6

2 Modèle utilisé et Fonctionnement **9**
2.1 Définition de MLPCLassifier : 9
2.2 Fonctionnement et concepts du MLP Classifier 9
2.2.1 Fonctionnement : 9
2.2.2 Concepts liés au MLPClassifier 9

3 Indicateurs de performances et Résultats **12**

4 Optimisation du modèle **14**

Optimisation **14**

Conclusion **15**

Annexes **16**

A Code python pour résoudre la problématique **16**
A.1 Exploration des données 16
A.2 Code Python pour les modèles 17
A.3 Code Python pour l'optimisation 17
A.4 Librairies utilisées 18

Table des figures

1.1 Breast Cancer 4
1.2 Breast Cancer Attributs 5
1.3 Exploration des données 1 6
1.4 Exploration des Données 2 7
1.5 Correlation Breast Cancer 7
1.6 Graphe correlation Breast Cancer 8

2.1 MLP Classifier Demo 10

3.1 Matrice de confusion pour notre modèle 12
3.2 Rapport de classification 13

4.1 liste des hyperparamètres pour GridsearchCV 14
4.2 :Résultats de l'optimisation 14

Chapitre 1

Données étudiées

1.1 Description des données

1.1.1 Qu'est ce que Breast Cancer dataset ?

Dans cette partie on vous parlera de la base de données utilisées pour notre modèle Breast Cancer. La base de données breast cancer est une base de données médicale utilisée pour la classification binaire de tumeurs mammaires comme bénignes ou malignes. Elle est souvent utilisée comme un exemple d'application d'apprentissage automatique pour la classification binaire dans le domaine médical.
La base de données contient 569 exemples de tumeurs mammaires, chacune décrite par 30 attributs numériques. Les attributs incluent des caractéristiques des noyaux des cellules présentes dans les images numérisées des tumeurs, telles que la taille, la forme, la texture et la densité.
Chaque exemple est étiqueté comme bénin (357 exemples) ou malin (212 exemples). L'objectif de la classification est de prédire si une nouvelle tumeur est bénigne ou maligne en fonction de ses attributs.
La base de données breast cancer est souvent utilisée comme exemple pour l'analyse de données et l'apprentissage automatique, en particulier pour les techniques de classification binaire. Elle a été introduite en 1995 par William H. Wolberg, W. Nick Street et Olvi L. Mangasarian de l'Université du Wisconsin-Madison. Ceci est démontré dans la figure ci -dessous :

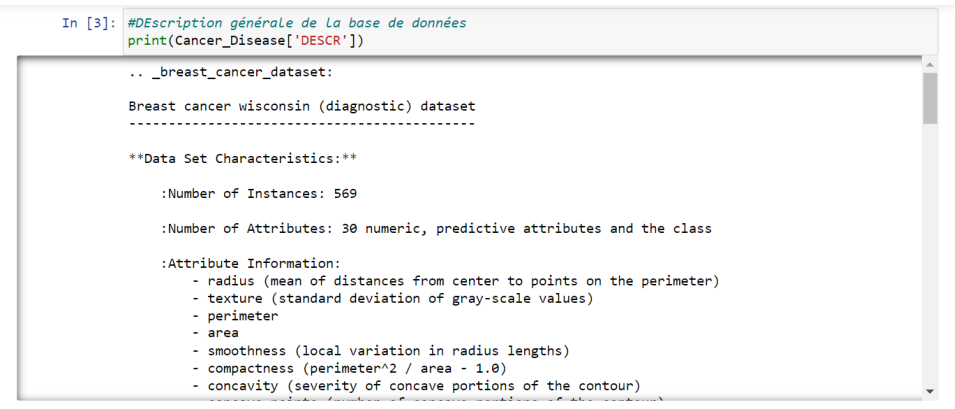

FIGURE 1.1 –

1.1.2 Attributs Breast Cancer

```
In [4]: #Nom classes à prédire
        Cancer_Disease.target_names

Out[4]: array(['malignant', 'benign'], dtype='<U9')

In [5]: #Les attributs du Dataframe
        Cancer_Disease.feature_names

Out[5]: array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
               'mean smoothness', 'mean compactness', 'mean concavity',
               'mean concave points', 'mean symmetry', 'mean fractal dimension',
               'radius error', 'texture error', 'perimeter error', 'area error',
               'smoothness error', 'compactness error', 'concavity error',
               'concave points error', 'symmetry error',
               'fractal dimension error', 'worst radius', 'worst texture',
               'worst perimeter', 'worst area', 'worst smoothness',
               'worst compactness', 'worst concavity', 'worst concave points',
               'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

FIGURE 1.2 –

Dans cette partie on essaiera d'expliquer les différents attributs du jeu de données.

- **Rayon (mean of distances from center to points on the perimeter)**
  Cette caractéristique mesure la distance moyenne entre le centre de la tumeur et les points situés sur le bord de la tumeur.

- **Texture (standard deviation of gray-scale values)**
  Cette caractéristique mesure l'écart-type des valeurs d'échelle de gris dans l'image de la tumeur. Elle reflète la variation de la couleur ou de la texture des cellules dans la région de la tumeur.

- **Périmètre**
  Cette caractéristique mesure la longueur totale de la ligne de la tumeur.

- **Surface**
  Cette caractéristique mesure la surface de la tumeur.

- **Smoothness (local variation in radius lengths)**
  Cette caractéristique mesure la variation locale de la longueur du rayon dans la tumeur. Elle reflète la régularité ou l'irrégularité de la forme de la tumeur.

- **Compacité (perimeter^2/$areaminus1.0$)**
  Cette caractéristique mesure la compacité de la tumeur. Elle est calculée en prenant le rapport entre le carré du périmètre et la surface de la tumeur, moins 1.0.

- **Concavité (severity of concave portions of the contour)**
  Cette caractéristique mesure la gravité des parties concaves du contour de la tumeur.

- **Points concaves (number of concave portions of the contour)**
  Cette caractéristique mesure le nombre de parties concaves du contour de la tumeur.

- **Symétrie**
  Cette caractéristique mesure la symétrie de la tumeur par rapport à son centre.

- **Dimension fractale ("coastline approximation" - 1)**
  Cette caractéristique mesure l'approximation de la tumeur à une ligne côtière fractale. Elle est calculée en prenant le rapport entre la longueur de la ligne côtière approximative et la longueur maximale possible pour une région donnée.

- **11-20. .**
  Les attributs 11 à 20 mesurent les mêmes caractéristiques que les attributs 1 à 10, mais pour les noyaux des cellules identifiés comme les plus grands dans l'image

- **21-30.**
  Les attributs 21 à 30 mesurent les mêmes caractéristiques que les attributs 1 à 10, mais pour les noyaux des cellules identifiés comme les plus petits dans l'image.

1.2 Exploration des Données

On cherchera à explorer les données en essayant de voir les différents types de ces données et s'il existe des données qui manquent. Dans notre cas et comme montré dans la figure ci-dessous les données sont numériques (Réels pour tous les attributs et entier pour le target) : // La prochaine étape est

```
In [6]: #Etude des différents types de données
        #on remarque que la majorité des données est de type réel alors que pour notre target le type est entie
        Cancer_Disease.frame.info()
        #pas de valeurs nulles
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
```

FIGURE 1.3 –

de vérifier les statistiques de chaque attributs(moyenne,mode,mediane ...) Ensuite nous optons pour la verification de la corrélation entre les différentes variables. Voir figure : Puis nous vérifions la corrélation entre la valuer dépendante(target) et toutes les autres variables à travers ce graphe :

```
In [7]: #statistiques pour chaque colonne
        Cancer_Disease.frame.describe()
```

Out[7]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fr dimei
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.06
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.00
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.04
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.05
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.06
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.06
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.09

8 rows × 31 columns

FIGURE 1.4 –

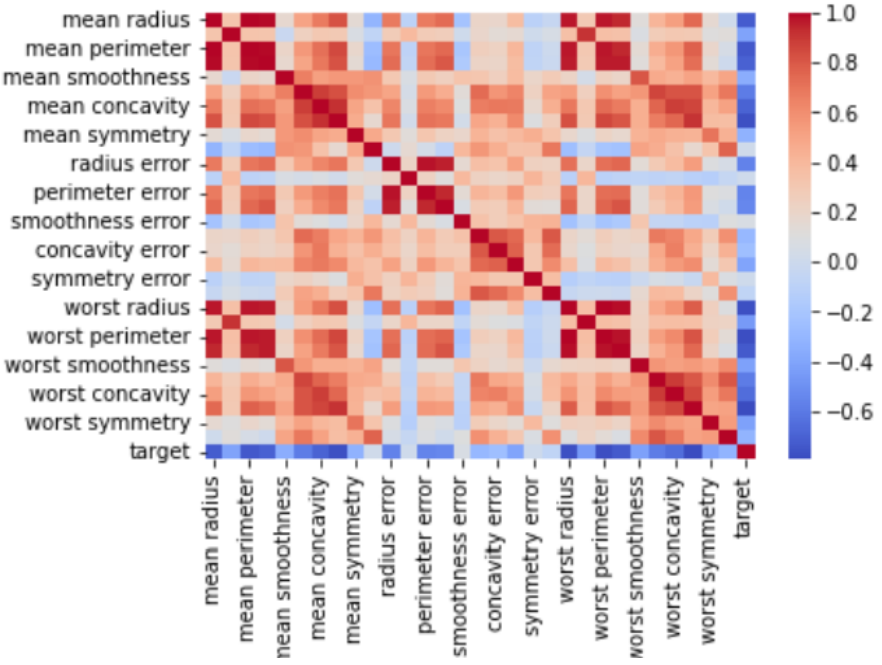

FIGURE 1.5 –

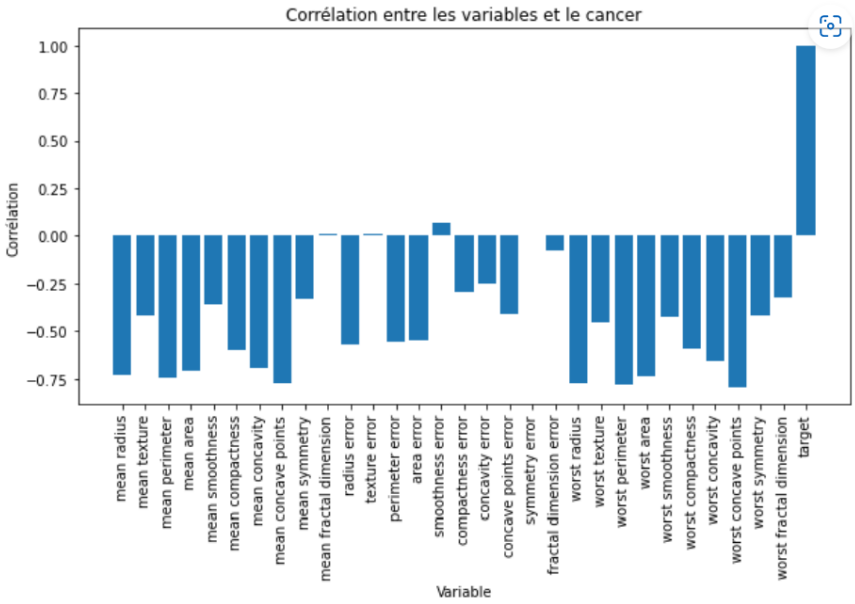
Corrélation entre les variables et le cancer
Corrélation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension
target
Variable

FIGURE 1.6 –

Chapitre 2

Modèle utilisé et Fonctionnement

2.1 Définition de MLPCLassifier :

Un MLP (Multi-Layer Perceptron) Classifier est un type de réseau de neurones artificiels qui est utilisé pour la classification supervisée. Il se compose de plusieurs couches de neurones qui sont connectées de manière dense, c'est-à-dire que chaque neurone dans une couche est connecté à tous les neurones dans la couche suivante. Le MLP Classifier est capable d'apprendre des relations non linéaires complexes entre les entrées et les sorties en ajustant les poids de connexion entre les neurones lors de l'apprentissage.
Plus précisément, un MLP Classifier est composé d'une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque couche est constituée de plusieurs neurones, qui effectuent des calculs sur les entrées qu'ils reçoivent et transmettent ensuite les résultats à la couche suivante. Les neurones dans les couches cachées utilisent une fonction d'activation non linéaire pour introduire de la non-linéarité dans les relations apprises par le modèle.

2.2 Fonctionnement et concepts du MLP Classifier

2.2.1 Fonctionnement :

un MLP Classifier est composé d'une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque couche est constituée de plusieurs neurones, qui effectuent des calculs sur les entrées qu'ils reçoivent et transmettent ensuite les résultats à la couche suivante. Les neurones dans les couches cachées utilisent une fonction d'activation non linéaire pour introduire de la non-linéarité dans les relations apprises par le modèle.

Lors de l'apprentissage, le MLP Classifier ajuste les poids de connexion entre les neurones en utilisant un algorithme d'optimisation pour minimiser une fonction de perte qui mesure la différence entre les sorties prédites par le modèle et les sorties réelles. Une fois l'apprentissage terminé, le MLP Classifier peut être utilisé pour prédire la classe d'une nouvelle entrée en passant l'entrée à travers le réseau de neurones et en retournant la sortie de la couche de sortie.

2.2.2 Concepts liés au MLPClassifier

Définitions des concepts liés au MLP Classifier :

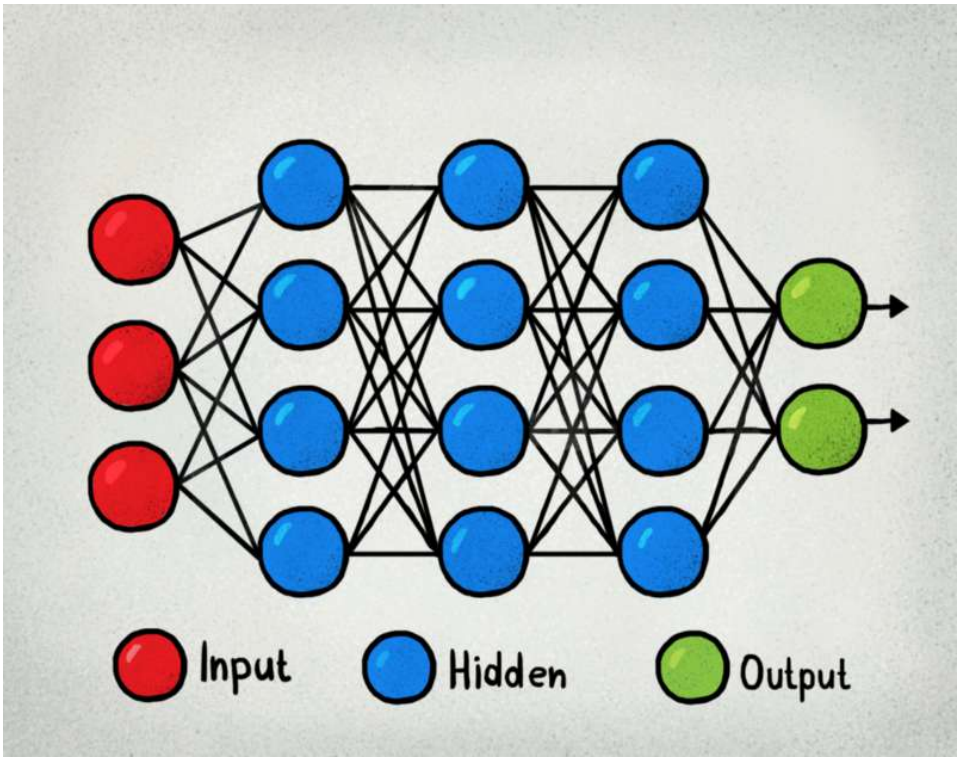

FIGURE 2.1 –

- **Fonction d'activation**
  Les fonctions d'activation sont utilisées dans les neurones pour introduire de la non-linéarité dans le modèle. Les fonctions d'activation couramment utilisées comprennent la fonction sigmoïde, la fonction ReLU, la fonction tangente hyperbolique, etc.

- **Rétropropagation**
  La rétropropagation est l'algorithme d'optimisation utilisé pour ajuster les poids de connexion entre les neurones lors de l'apprentissage. Il fonctionne en calculant le gradient de la fonction de perte par rapport aux poids de connexion et en utilisant ce gradient pour mettre à jour les poids dans le sens qui minimise la perte.

- **Couche d'entrée**
  La couche d'entrée est la première couche du MLP Classifier, où les données d'entrée sont introduites dans le réseau.

- **Couches cachées**
  Les couches cachées sont situées entre la couche d'entrée et la couche de sortie et sont responsables de l'apprentissage des relations non linéaires entre les entrées et les sorties.

- **Couche de sortie**
  L La couche de sortie est la dernière couche du MLP Classifier, où la sortie finale du modèle est générée. Le nombre de neurones dans cette couche dépend du nombre de classes dans le problème de classification.

- **Fonction de perte**

La fonction de perte mesure la différence entre les sorties prédites par le modèle et les sorties réelles, et est utilisée pour guider l'optimisation des poids du modèle.

- **Algorithme d'optimisation**
  L'algorithme d'optimisation est utilisé pour ajuster les poids de connexion entre les neurones du modèle en minimisant la fonction de perte. Des exemples d'algorithmes d'optimisation couramment utilisés comprennent la descente de gradient stochastique (SGD), l'algorithme ADAM, etc.

Chapitre 3

Indicateurs de performances et Résultats

Pour Evaluer notre modèle plsieurs méthodes sont utilisées. Parmi les méthodes utilisées dans notre cas :

- **La matrice de confusion**
  La matrice de confusion est une matrice qui est utilisée pour évaluer les performances d'un modèle de classification en comparant les prédictions du modèle avec les véritables labels de la classe. La matrice de confusion est souvent utilisée pour des problèmes de classification binaire (deux classes) mais peut également être étendue pour des problèmes de classification multi-classes (plus de deux classes). La matrice de confusion est une matrice carrée où chaque ligne représente la classe réelle et chaque colonne représente la classe prédite par le modèle. Les éléments sur la diagonale de la matrice représentent le nombre de prédictions correctes pour chaque classe, tandis que les éléments hors diagonale représentent le nombre de prédictions incorrectes.
  Vous trouverez dans l'image suivante La matrice de confusion pour notre modèle MLPCLassifier :

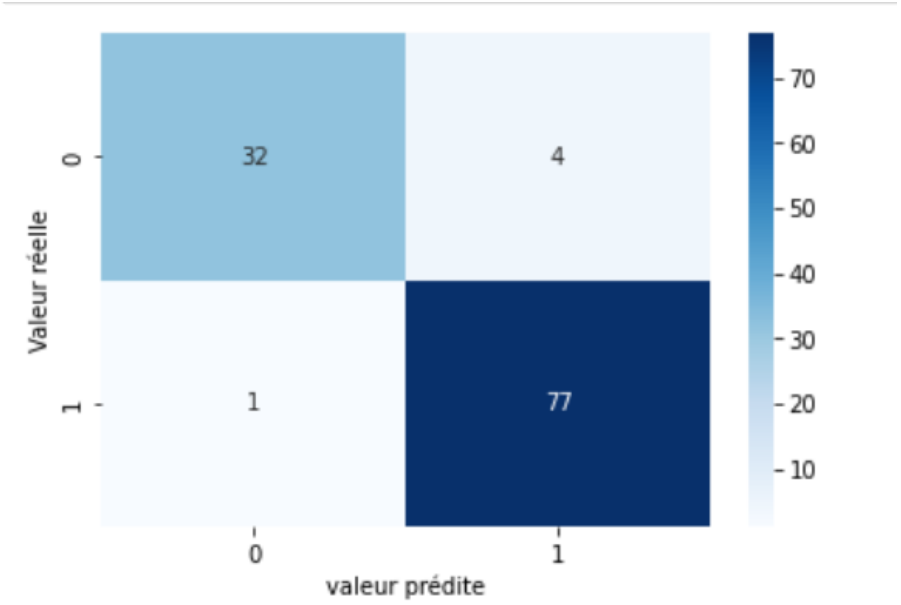

Figure 3.1 –

- **Le rapport de classification**
  Le rapport de classification est une mesure qui fournit des informations détaillées sur la précision, le rappel, le score F1 et le support pour chaque classe dans un problème de classification multi-classes. Le rapport de classification est souvent utilisé en conjonction avec la matrice de confusion pour évaluer les performances d'un modèle de classification.
  Le rapport de classification calcule les métriques de performance pour chaque classe en considérant cette classe comme la classe positive et toutes les autres classes comme la classe négative. Ensuite, les métriques sont moyennées sur toutes les classes pour obtenir une mesure agrégée.

  Voici une explication des métriques mesurées :

  Précision (precision) : La précision mesure la proportion de vrais positifs parmi toutes les prédictions positives. Elle est calculée comme VP / (VP + FP).
  Rappel (recall) : Le rappel mesure la proportion de vrais positifs parmi tous les exemples positifs. Il est calculé comme VP / (VP + FN).
  Score F1 (F1-score) : Le score F1 est la moyenne harmonique de la précision et du rappel et est une mesure globale de la performance du modèle. Il est calculé comme 2 * (precision * recall) / (precision + recall).
  Support : Le support est le nombre total d'exemples dans chaque classe.

  Voici le rapport de classification de notre modèle :

	precision	recall	f1-score	support
0	0.97	0.89	0.93	36
1	0.95	0.99	0.97	78
accuracy			0.96	114
macro avg	0.96	0.94	0.95	114
weighted avg	0.96	0.96	0.96	114

FIGURE 3.2 –

Chapitre 4

Optimisation du modèle

Dans cette partie on a essayé d'optimiser les hyperparamètres de notre mlp classifier(la fonction d'activation,nombre de couches cachées,l'algorithme d'optimisation) qu'on a expliqué durent le Chapitre 2 pour obtenir le meilleur score avec le module GridsearchCV.

GridSearchCV est une méthode de recherche d'hyperparamètres pour les modèles d'apprentissage automatique. L'objectif de la recherche d'hyperparamètres est de trouver les valeurs des paramètres qui donnent les meilleures performances de modèle pour un ensemble de données donné. La figure suivante montre la liste des hyperparamètres avec lesquels on va utiliser GridsearchCV ;

```
In [18]: param_grid = {
             'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 100),(30,30,30)],
             'activation': ['relu', 'tanh', 'logistic'],
             'solver': ['adam', 'sgd'],
             'learning_rate': ['constant', 'adaptive']
         }
```

FIGURE 4.1 –

et voici le résultat d'optimisation (les meilleurs paramètres avec leur score)

```
best_params = grid_search.best_params_
best_mlp = MLPClassifier(**best_params, random_state=42)
print(best_params)
```

```
{'activation': 'relu', 'hidden_layer_sizes': (50,), 'learning_rate': 'constant', 'solver': 'adam'}
```

```
best_mlp.fit(X_train, y_train)
accuracy = best_mlp.score(X_test, y_test)
print(accuracy)
```

```
0.9736842105263158
```

FIGURE 4.2 –

Conclusion et Perspectives

Dans ce projet, nous avons utilisé un MLP Classifier pour classer les cas de tissus mammaires en tumeurs bénignes ou malignes. Nous avons commencé par explorer les cas . Ensuite, nous avons entraîné le modèle sur les données d'entraînement et ajusté les hyperparamètres pour améliorer les performances.

Les résultats obtenus montrent que notre modèle a réussi à classifier les tumeurs bénignes et malignes avec une précision de 97 %, ce qui est un résultat très satisfaisant. Les métriques de précision, de rappel et de score F1 ont également été calculées pour chaque classe, et ont montré que le modèle est performant dans la prédiction de chaque classe.

En conclusion, nous avons réussi à construire un modèle de classification précis pour le cancer du sein en utilisant un MLP Classifier. Ce modèle peut être utilisé pour aider les professionnels de la santé à diagnostiquer les tumeurs mammaires et à prendre des décisions éclairées sur les traitements. Cependant, d'autres recherches sont nécessaires pour explorer les limites de notre modèle et pour améliorer sa performance sur des ensembles de données plus diversifiés.

Annexe A

Code python pour résoudre la problématique

A.1 Exploration des données

```
 import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.metrics import confusion_matrix,classification_report
import seaborn as sns
import matplotlib.pyplot as plt
Cancer_Disease=load_breast_cancer(as_frame=True)
#DEscription générale de la base de données
print(Cancer_Disease['DESCR'])
#Nom classes à prédire
Cancer_Disease.target_names
#les attributs du Dataframe
Cancer_Disease.feature_names
#Etude des différents types de données
#on remarque que la majorité des données est de type réel alors que pour notre target le t
Cancer_Disease.frame.info()
#pas de valeurs nulles
#statistiques pour chaque colonne
Cancer_Disease.frame.describe()
#Etablissement d'une matrice de corrélation entre les attributs
Corrmatrix=Cancer_Disease.frame.corr()
correlations = Cancer_Disease.frame.corrwith(Cancer_Disease.frame.target)
sns.heatmap(Corrmatrix, cmap="coolwarm",)
plt.figure(figsize=(10, 5))
plt.bar(correlations.index, correlations)
plt.xticks(rotation=90)
plt.xlabel('Variable')
plt.ylabel('Corrélation')
plt.title('Corrélation entre les variables et le cancer')
plt.show()
```

A.2 Code Python pour les modèles

```
 from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
X=Cancer_Disease.data
y=Cancer_Disease.target
#préparation des ensembles train et test
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
#normalisation des données pour obtenir un meilleur score après l'entrainement
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
Scaler.fit(X_train)
X_train=Scaler.transform(X_train)
X_test=Scaler.transform(X_test)
#Entrainement du modèle
MLP=MLPClassifier(hidden_layer_sizes=(40,40,40))
MLP.fit(X_train,y_train)
#Evaluation du modèle avec la matrice de confusion et le rapport de classification
Y_pred=MLP.predict(X_test)
cm=confusion_matrix(Y_pred,y_test)
sns.heatmap(cm, annot=True, cmap="Blues")
plt.xlabel('valeur prédite')
plt.ylabel('Valeur réelle')
plt.show()
print(classification_report(Y_pred,y_test))
```

A.3 Code Python pour l'optimisation

```
    from sklearn.model_selection import GridSearchCV
    param_grid = {
    'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 100),(30,30,30)],
    'activation': ['relu', 'tanh', 'logistic'],
    'solver': ['adam', 'sgd'],
    'learning_rate': ['constant', 'adaptive']
}
mlp = MLPClassifier(random_state=42)
grid_search = GridSearchCV(mlp, param_grid, cv=5, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
best_mlp = MLPClassifier(**best_params, random_state=42)
print(best_params)
```

```
best_mlp.fit(X_train, y_train)
accuracy = best_mlp.score(X_test, y_test)
print(accuracy)
```

A.4 Librairies utilisées

pandas (importé sous le nom "pd")
scikit-learn (importé sous le nom sklearn)
seaborn (importé sous le nom "sns")
matplotlib.pyplot (importé sous le nom "plt")