

Détection de la maladie de Parkinson

BOUKHOBZA El Mehdi
Numéro d'inscription: 10938
Thème: Santé, prévention

Sommaire

01

Problématique

02

Introduction

03

**L'analyse des données
(Data analysis)**

04

**K plus proches voisins
(KNN)**

- Fondement mathématique
- Hyperparamètres
- Implémentation Python

05

**Naïve Bayes
(NB)**

- Fondement mathématique
- Hyperparamètres
- Implémentation Python

06

**Comparaison entre
KNN et NB**

07

**Construction d'un modèle plus
performant**

08

Conclusion

09

Annexes



01

Problématique

Déterminer les Hyperparamètres de KNN et NB en étudiant leurs fondements mathématiques et les appliquer dans le cas de la maladie de Parkinson.



02

Introduction

10 000 000 de malades





03

L'analyse des données (Data analysis)

Informations sur les attributs ⁶

MDVP:F0(Hz)	Fréquence vocale fondamentale moyenne
MDVP:Fhi(Hz)	Fréquence vocale fondamentale maximale
MDVP:Flo(Hz)	Fréquence minimale de base vocale
MDVP:Jitter(%) MDVP:Jitter(Abs) MDVP:RAP5 MDVP:PPQ Jitter:DDP	Plusieurs mesures de variation de la fréquence fondamentale
MDVP:Shimmer MDVP:Shimmer(dB) Shimmer:APQ3 Shimmer:APQ5 MDVP:APQ Shimmer:DDA	Plusieurs mesures de variation d'amplitude
NHR HNR	Deux mesures du rapport entre le bruit et les composantes tonales de la voix
RPDE D2	Deux mesures dynamiques non linéaires de la complexité
DFA	Exposant d'échelle fractale de signal
spread1 spread2 PPE	Trois mesures non linéaires de variation de fréquence fondamentale
name	Chaîne de caractère (unique pour chaque instance)

Nettoyage de données(*Data Cleaning*)

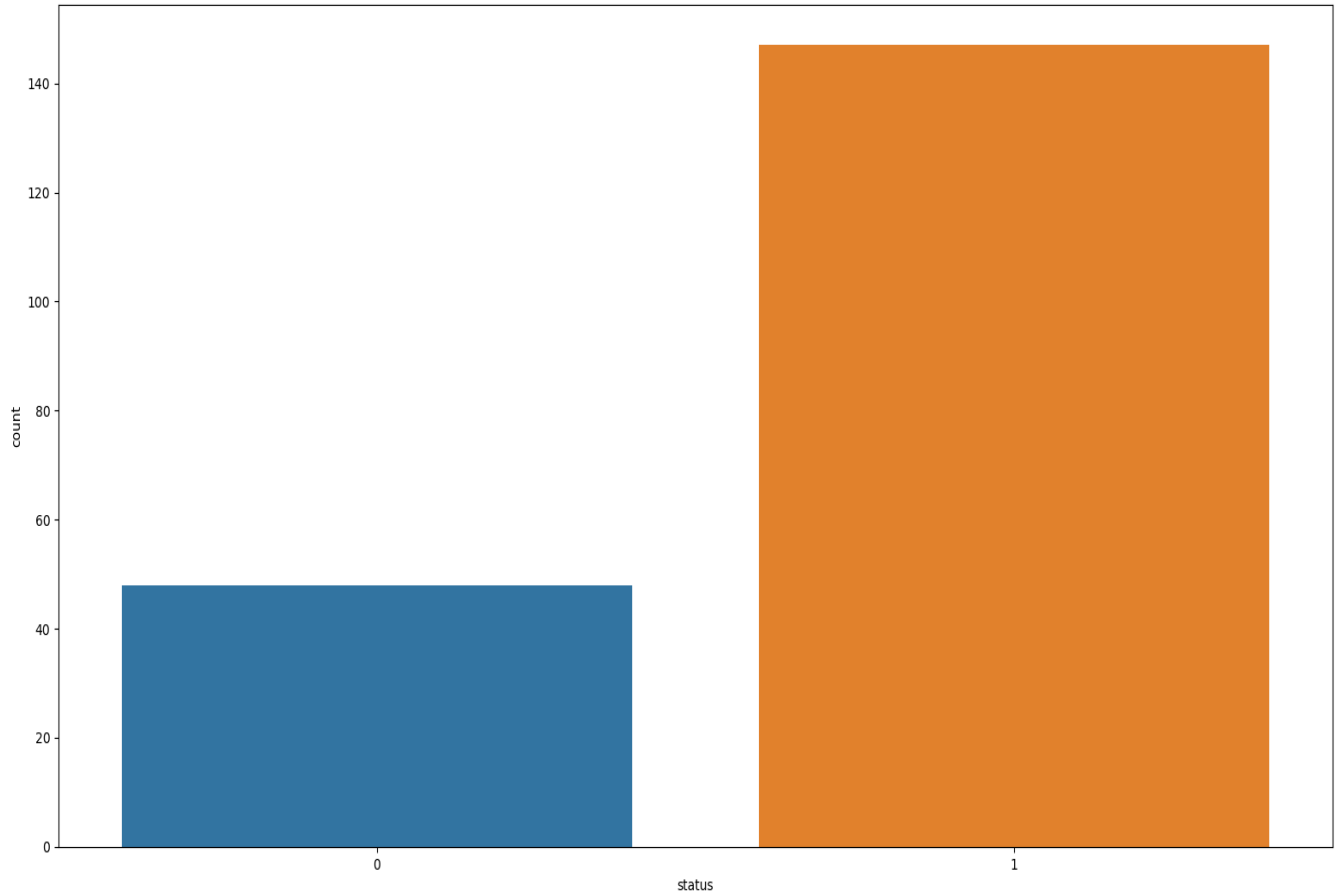
les enregistrements dupliqués

Le nombre de doublons dans cette base de donnée est 0

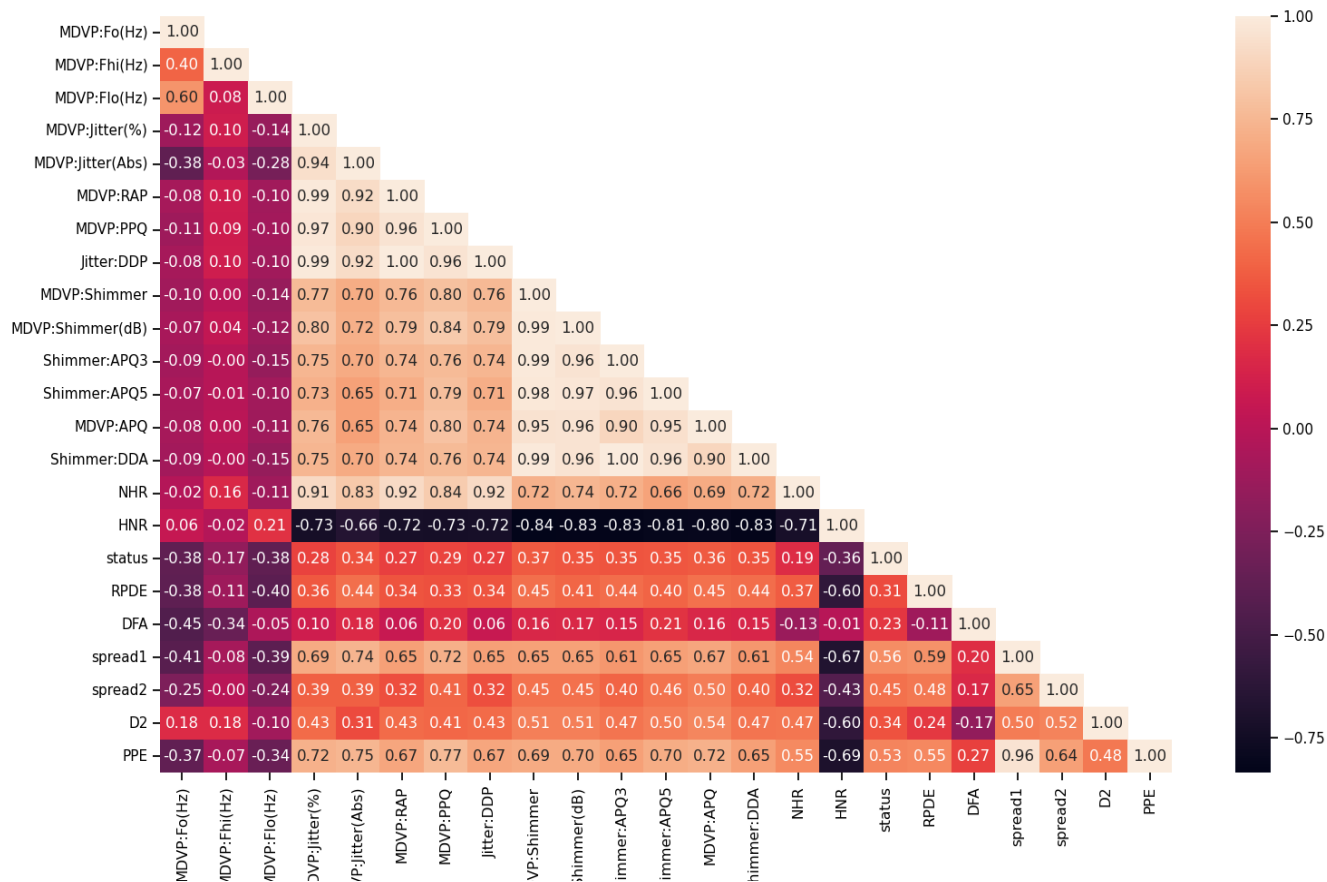
les enregistrements inconnus ou manquants

name	0	Jitter:DDP	0	NHR	0
MDVP:F0(Hz)	0	MDVP:Shimmer	0	HNR	0
MDVP:F1(Hz)	0	MDVP:Shimmer(dB)	0	status	0
MDVP:F2(Hz)	0	Shimmer:APQ3	0	RPDE	0
MDVP:F3(Hz)	0	Shimmer:APQ5	0	DFA	0
MDVP:F0F2(Hz)	0	MDVP:APQ	0	spread1	0
MDVP:F0F2(Hz)	0	Shimmer:DDA	0	spread2	0
MDVP:F0F2(Hz)	0		0	D2	0
MDVP:F0F2(Hz)	0		0	PPE	0

Distribution de l'attribut « Status »



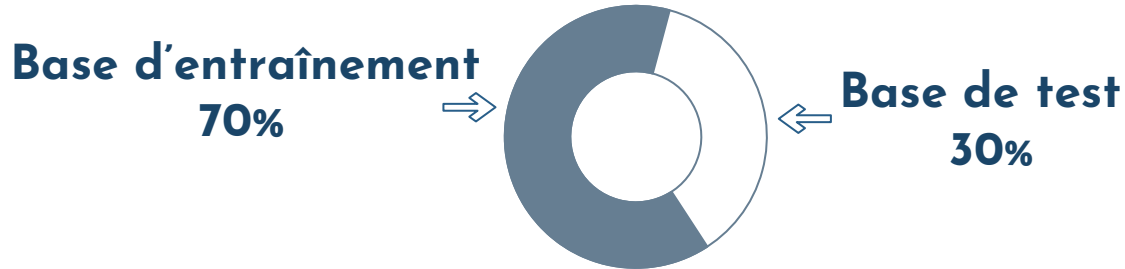
Calcul de corrélation



Coefficients de corrélations entre « status » et les autres attributs

status	1.000000	Shimmer:APQ3	0.347617	MDVP:RAP	0.266668
spread1	0.564838	Shimmer:DDA	0.347608	Jitter:DDP	0.266646
PPE	0.531039	D2	0.340232	DFA	0.231739
spread2	0.454842	MDVP:Jitter(Abs)	0.338653	NHR	0.189429
MDVP:F0(Hz)	0.383535	RPDE	0.308567	MDVP:Fhi(Hz)	0.166136
MDVP:FLo(Hz)	0.380200	MDVP:PPQ	0.288698		
MDVP:Shimmer	0.367430	MDVP:Jitter(%)	0.278220		
MDVP:APQ	0.364316	MDVP:RAP	0.266668		
HNR	0.361515	Jitter:DDP	0.266646		

Répartir la base de donnée



Standardiser les attributs

$$X_{standard} = \frac{X - \mu}{\sigma}$$

Variable centrée réduite associée à X



04

K plus proches voisins (KNN)

- Fondement mathématique
- Hyperparamètres
- Implémentation Python

Soit $i \in \llbracket 1, n \rrbracket$, on note X_i un point dans la base d'entraînement, y_i la variable cible

$$\psi = (X_1, X_2, X_3, \dots, X_n) \quad \text{et} \quad \gamma = (y_1, y_2, y_3, \dots, y_n)$$

$$D = \{(X_1, y_1), (X_2, y_2), (X_3, y_3), \dots, (X_n, y_n)\} \in \psi \times \gamma : \text{Base d'entraînement}$$

Soit X un point dans la base de test,

S_X : l'ensemble des K plus proches voisins de X , $S_X \subseteq D, |S_X| = K$

$$\text{et } \forall (X', y) \in D/S_X : \text{dist}(X, X') \geq \max_{(X, X'') \in S_X} \text{dist}(X, X'')$$

Distance Euclidienne:

; Distance de Manhattan:

$$\text{dist}(X, X') = \left(\sum_{i=1}^n (x_i - x'_i)^2 \right)^{1/2}$$

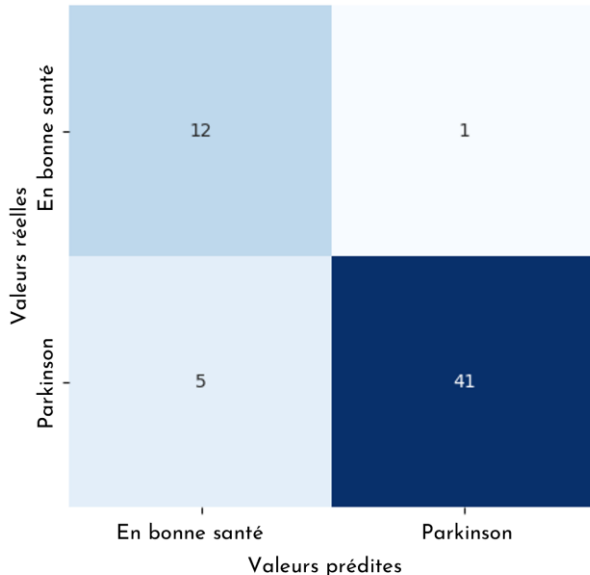
$$\text{dist}(X, X') = \sum_{i=1}^n |x_i - x'_i|$$

$\varphi : D \rightarrow \gamma$ Détermine la valeur de y (0 ou 1)
 $X \mapsto y$ la plus fréquente dans S_X
et l'assigne à la variable X

K : entier impair (3)

Implémentation Python K plus proches voisins

Matrice de confusion



Rapport de classification du modèle
K plus proches voisins

$$precision = \frac{12 + 41}{12 + 5 + 41 + 1}$$

$$rappel = \frac{12}{12 + 5}$$

$$specificite = \frac{41}{41 + 1}$$

précision est : 0.8983050847457628

rappel est : 0.7058823529411765

spécificité est : 0.9761904761904762

04 Naïve Bayes (NB)

- Fondement mathématique
 - Hyperparamètres
- Implémentation Python

Soit $X = (x_1, x_2, x_3, \dots, x_n)$ un point dans la base de donnée avec x_i les attributs de ce point, et y la variable cible.

Théorème de Bayes:
$$P_X(y) = \frac{P_y(X) * P(y)}{P(X)}$$

Calcul:

$$\begin{aligned} P_y(X) &= P_y(x_1, x_2, x_3, \dots, x_n) \\ &= \frac{P(y) * P_y(x_1) * P_{y \cap x_1}(x_2) * P_{y \cap x_1 \cap x_2}(x_3) * \dots * P_{y \cap x_1 \cap x_2 \cap x_3 \cap \dots \cap x_{n-1}}(x_n)}{P(y)} \\ &= P(y) * P_y(x_1) * P_{y \cap x_1}(x_2) * P_{y \cap x_1 \cap x_2}(x_3) * \dots * P_{y \cap x_1 \cap x_2 \cap x_3 \cap x_4 \cap \dots \cap x_{n-1}}(x_n) \end{aligned}$$

D'après l'indépendance des attributs:

$$P_X(y) = \frac{P(y) * P_y(x_1) * P_y(x_2) * P_y(x_3) * \dots * P_y(x_n)}{P(x_1) * P(x_2) * P(x_3) * \dots * P(x_n)}$$
$$P_X(y) \propto P(y) * \prod_{i=1}^n P_y(x_i)$$

$P(y) * \prod_{i=1}^n P_y(x_i)$ Pour quelle valeur de y ce terme est maximal ?

$$y = \operatorname{argmax}_y \left(P(y) * \prod_{i=1}^n P_y(x_i) \right)$$

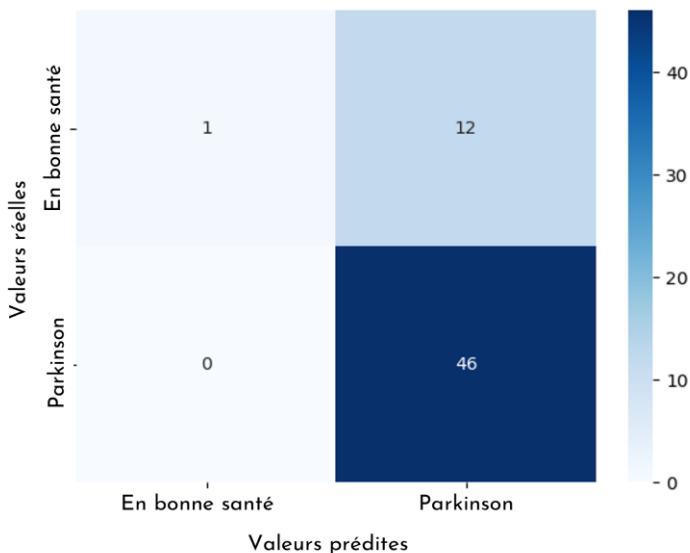
Multinomial

Gaussien

Bernoulli

Implémentation Python Naïve Bayes

Matrice de confusion



Rapport de classification du modèle
Naïve Bayes

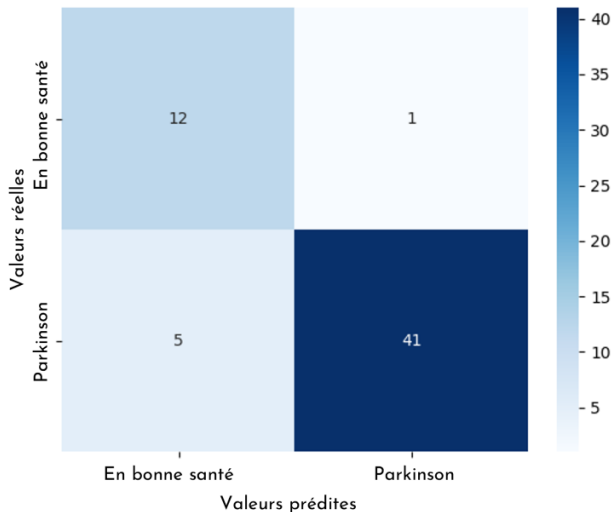
```
précision est : 0.7966101694915254  
rappel est : 1.0  
spécificité est : 0.7931034482758621
```



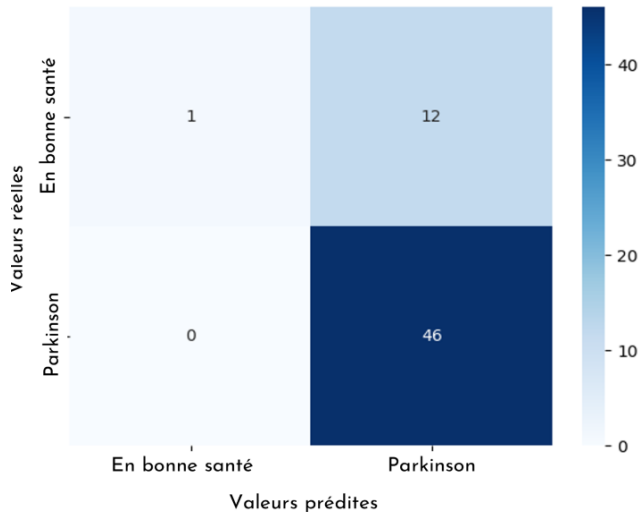
06

Comparaison entre KNN et NB

Matrice de confusion de KNN



Matrice de confusion de NB



Rapport de classification de KNN et NB

	précision	rappel	spécificité
KNN	0.898305	0.705882	0.976190
NB	0.796610	1.000000	0.793103



07

Construction d'un modèle plus performant

Bagging de KNN

Rapport de classification du modèle Bagging KNN

Estimators = 100

```
précision est : 0.9152542372881356  
rappel est : 0.75  
spécificité est : 0.9767441860465116
```

Comparaison entre KNN, NB et bagging de 100 versions de KNN

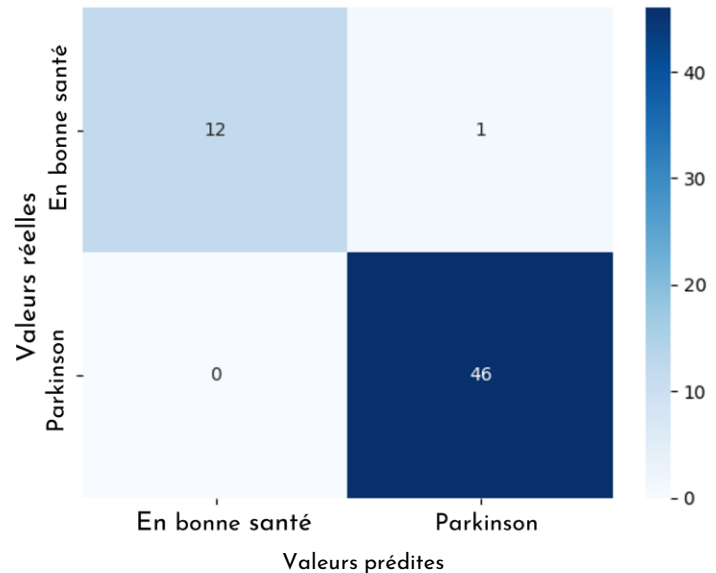
	précision	rappel	spécificité
KNN	0.898305	0.705882	0.976190
NB	0.796610	1.000000	0.793103
BaggKNN	0.915254	0.750000	0.976744

Méthode « Weighted Average Ensemble (ens) »

Coefficient de NB:
 $\text{accuracy de NB} \times \underline{0.2}$

Coefficient de Bagging KNN :
 $\text{accuracy de Bagging KNN} \times \underline{0.5}$

Matrice de confusion du modèle "ens"

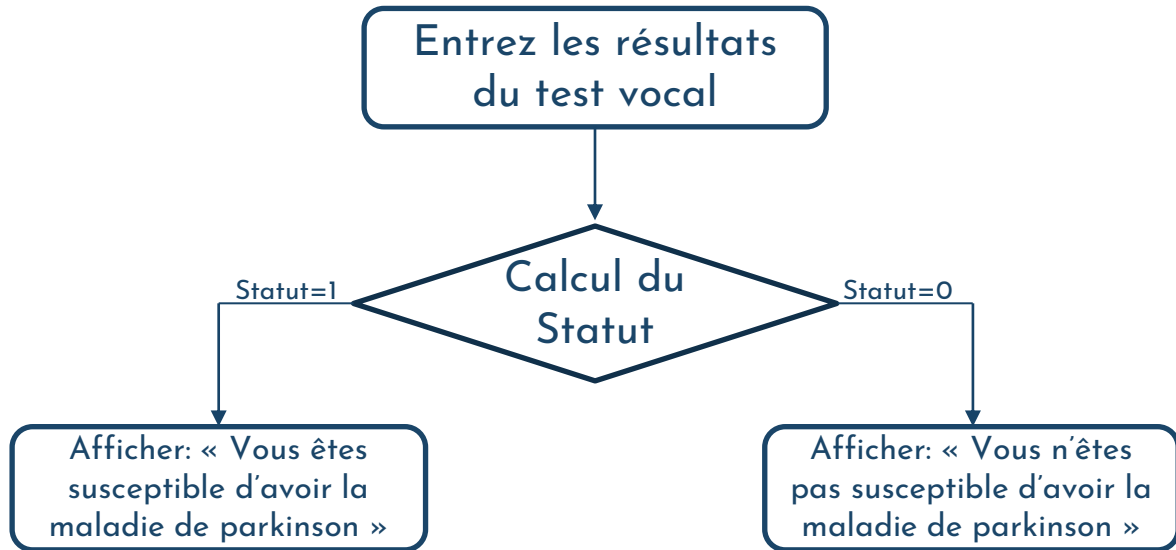


Résumé de tous les modèles

	précision	rappel	spécificité
KNN	0.898305	0.705882	0.976190
NB	0.796610	1.000000	0.793103
BaggKNN	0.915254	0.750000	0.976744
ens	0.983051	1.000000	0.978723

08 Conclusion

• Algorithme de la fonction « Parkinson_test »





**Merci de Votre
Attention**

09

Annexes

Code de la fonction Parkinson_test

```
def Parkinson_test():  
    L=[]  
    print('Test de Parkinson\nVeuillez entrer les resultats de votre test:')  
    MDVP_Fo=float(input('MDVP_Fo='))  
    L.append(MDVP_Fo)  
    MDVP_Fhi=float(input('MDVP_Fhi='))  
    L.append(MDVP_Fhi)  
    MDVP_Flo=float(input('MDVP_Flo='))  
    L.append(MDVP_Flo)  
    MDVP_Jitter=float(input('MDVP_Jitter='))  
    L.append(MDVP_Jitter)  
    MDVP_RAP=float(input('MDVP_RAP='))  
    L.append(MDVP_RAP)  
    MDVP_PPQ=float(input('MDVP_PPQ='))  
    L.append(MDVP_PPQ)  
    Jitter_DDP=float(input('Jitter_DDP='))  
    L.append(Jitter_DDP)  
    MDVP_Shimmer=float(input('MDVP_Shimmer='))  
    L.append(MDVP_Shimmer)
```

```

Shimmer_APQ3=float(input('Shimmer_APQ3='))
L.append(Shimmer_APQ3)
Shimmer_APQ5=float(input('Shimmer_APQ5='))
L.append(Shimmer_APQ5)
MDVP_APQ=float(input('MDVP_APQ='))
L.append(MDVP_APQ)
Shimmer_DDA=float(input('Shimmer_DDA='))
L.append(Shimmer_DDA)
NHR=float(input('NHR='))
L.append(NHR)
HNR=float(input('HNR='))
L.append(HNR)
RPDE=float(input('RPDE='))
L.append(RPDE)
DFA=float(input('DFA='))
L.append(DFA)
spread1=float(input('spread1='))
L.append(spread1)
spread2=float(input('spread2='))
L.append(spread2)
Statut=ens.predict([L])
if Statut==1:
    print("Vous etes susceptible d'avoir la maladie de parkinson\n"
          "Veuillez contacter un medecin au plus tot")
else:
    print("Vous n'etes pas susceptible d'avoir la maladie de parkinson ")

```

Code Python

```
import pandas as pd
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import BernoulliNB
# from sklearn.tree import export_graphviz
from sklearn.ensemble import VotingClassifier
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
```


```

#Lire la base de donnee
dataset = pd.read_csv("D:\\TIPE\\Data")
print(dataset.head())
print(dataset.dtypes)
duplique = dataset.duplicated()
print('Le nombre de doublons dans cette base de donnee est',sum(duplique))
print(dataset.isnull().sum())

#Graphe Variation de l'attribut «Status»
sns.countplot(x=dataset['status'])
plt.show()

#Correlation
corr = dataset.corr()
sns.set_context("notebook", font_scale=1.0, rc={"lines.linewidth ": 50})
plt.figure(figsize=(18,20))
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask, 1)] = True
a = sns.heatmap(corr,mask=mask, annot=True, fmt='.2f')
rotx = a.set_xticklabels(a.get_xticklabels(), rotation=90)
roty = a.set_yticklabels(a.get_yticklabels(), rotation=0)
plt.show()
correlation_values=dataset.corr()['status']
print(correlation_values.abs().sort_values(ascending=False))

```

```
#L'analyse des données
Y=dataset['status']
cols_to_drop =['NHR','MDVP:Fhi(Hz)','name','status']
X=dataset.drop(cols_to_drop,axis=1)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=None)
#Standardiser les attributs
scaler=StandardScaler().fit(X_train)
scaler_X_train=scaler.transform(X_train)
scaler=StandardScaler().fit(X_test)
scaler_X_test=scaler.transform(X_test)
```

```

#K plus proches voisins (KNN)
KNN = KNeighborsClassifier(n_neighbors=3, weights='distance', metric='euclidean')
KNN.fit(scaler_X_train, Y_train)
Y_Knn_pred = KNN.predict(scaler_X_test)
cfKN = confusion_matrix(Y_test, Y_Knn_pred)
ax = sns.heatmap(cfKN, annot=True, cmap='Blues')
ax.set_xlabel('\nValeurs prédites')
ax.set_ylabel('Valeurs réelles');
ax.xaxis.set_ticklabels(['En bonne santé', 'Parkinson'])
ax.yaxis.set_ticklabels(['En bonne santé', 'Parkinson'])
plt.plot()

accKN = (cfKN[0][0]+cfKN[1][1])/(cfKN[0][0]+cfKN[0][1]+cfKN[1][0]+cfKN[1][1])
RecallKN = cfKN[0][0]/(cfKN[0][0]+cfKN[1][0])
SpecificityKN = cfKN[1][1]/(cfKN[1][1]+cfKN[0][1])
print("précision est :", accKN)
print('rappel est :', RecallKN)
print('spécificité est :', SpecificityKN)
plt.show()

```

```


#Naive Bayes
NB= BernoulliNB(alpha=3.0, binarize=2.0, fit_prior=True, class_prior=None)
NB.fit(scaler_X_train, Y_train)
Y_NB_pred = NB.predict(scaler_X_test)
print("confusion_matrix NB")
cfNB = confusion_matrix(Y_test, Y_NB_pred)
ax = sns.heatmap(cfNB, annot=True, cmap='Blues')
ax.set_xlabel('\nValeurs prédites')
ax.set_ylabel('Valeurs réelles');
ax.xaxis.set_ticklabels(['En bonne santé', 'Parkinson'])
ax.yaxis.set_ticklabels(['En bonne santé', 'Parkinson'])
accNB = (cfNB[0][0]+cfNB[1][1])/(cfNB[0][0]+cfNB[0][1]+cfNB[1][0]+cfNB[1][1])
RecallNB = cfNB[0][0]/(cfNB[0][0]+cfNB[1][0])
SpecificityNB = cfNB[1][1]/(cfNB[1][1]+cfNB[0][1])
print("précision est :", accNB)
print('rappel est :', RecallNB)
print('spécificité est :', SpecificityNB)
plt.show()

```

```

#Bagging de KNN
BaggKNN = BaggingClassifier(base_estimator=KNN,n_estimators=100)
BaggKNN.fit(scaler_X_train, Y_train)
Y_BaggKNN_pred=BaggKNN.predict(scaler_X_test)
cfBaggKNN = confusion_matrix(Y_test, Y_BaggKNN_pred)
accBaggKNN = (cfBaggKNN[0][0]+cfBaggKNN[1][1])/
              (cfBaggKNN[0][0]+cfBaggKNN[0][1]+cfBaggKNN[1][0]+cfBaggKNN[1][1])
RecallBaggKNN = cfBaggKNN[0][0]/(cfBaggKNN[0][0]+cfBaggKNN[1][0])
SpecificityBaggKNN = cfBaggKNN[1][1]/(cfBaggKNN[1][1]+cfBaggKNN[0][1])
print("précision est :",accBaggKNN)
print('rappel est :',RecallBaggKNN)
print('spécificité est :',SpecificityBaggKNN)
#Ensemble Weighted Method
models = [('Naive bayes',NB),('Bagging KNN ',BaggKNN)]
weights = [0.2*accNB,0.5*accBaggKNN]
ens = VotingClassifier(estimators=models, weights=weights, voting='soft')
ens.fit(scaler_X_train, Y_train)
Y_ens_pred = ens.predict(scaler_X_test)
cfens = confusion_matrix(Y_test, Y_ens_pred)
ax = sns.heatmap(cfens, annot=True, cmap='Blues')
ax.set_xlabel('\nValeurs prédites')
ax.set_ylabel('Valeurs réelles');
ax.xaxis.set_ticklabels(['En bonne santé','Parkinson'])
ax.yaxis.set_ticklabels(['En bonne santé','Parkinson'])
accBaggens = (cfens[0][0]+cfens[1][1])/(cfens[0][0]+cfens[0][1]+cfens[1][0]+cfens[1][1])
Recallens = cfens[0][0]/(cfens[0][0]+cfens[1][0])
Specificityens = cfens[1][1]/(cfens[1][1]+cfens[0][1])

```



```
#Résumé de tous les modèles
```

```
summary = {  
    'Accuracy': [accKN, accNB, accBaggKNN, accBaggens],  
    'Recall': [RecallKN, RecallNB, RecallBaggKNN, Recallens],  
    'Specificity': [SpecificityKN, SpecificityNB, SpecificityBaggKNN, Specificityens]  
}  
  
names = ['KNN', 'NB', 'BaggKNN', 'ens']  
sum_df = pd.DataFrame(summary, names)  
print(sum_df)  
plt.plot()  
plt.show()
```