Intelligence Artificielle: Sujet 9 Détection du Cancer

Ce rapport présentera une des utilisations possible du Machine Learning pour la détection du cancer du sein chez la femme.

Nous avons choisi de travailler sur la détection du cancer, plutôt que sur tout autre sujet proposé sur le lien DropBox de notre professeur Mr Boige. Tout d'abord ce choix s'explique par le contexte actuel, l'épidémie de COVID-19, nous a montré les effets dévastateurs que peuvent avoir la maladie sur notre société, jusqu'à la paralysie totale de notre économie. Nous avons alors pensé que, découvrir une nouvelle manière d'apporter une solution concrète à une cause aussi répandue que celle du cancer en manipulant le Machine Learning, était une façon pour nous futurs ingénieurs de se consacrer à la tâche commune de rendre la société et la vie de nos concitoyens meilleure en apportant des solutions innovantes aux caractéristiques scientifiques et mathématiques poussées. Ainsi, c'est un chemin possible dans l'accomplissement du devoir de l'ingénieur et aussi celui d'étudiant à la recherche de challenge et de moyens de mettre à profit ses compétences techniques. Par ailleurs, le Machine Learning est un domaine que nous avons abordé pour la première fois cette année, c'était alors une occasion rêvée pour nous, Edouard et moi-même de monter en compétences dans ce domaine à la pointe de l'innovation.

Le projet était de recréer, avec comme support et point de départ l'ensemble des documents présentés dans le DropBox (voir lien dans l'annexe), sous l'onglet 9 titré Utilisation du Machine Learning pour la Détection Cancer. Nous avons choisi de répartir les tâches de la manière suivante, au vu du planning chargé en examens et rendu de la période de fin d'année d'ING 4.

Semaine 1: étude et analyse des documents mis à disposition en profondeur/ choix du cancer sur lequel nous souhaitions travailler/ analyse et pronostic de réalisation du projet.

Semaine 2: Recherche annexe chacun de notre côté afin d'obtenir un plus large choix d'informations à piocher et de possibilités d'implémentation.

Semaine 3: Choix porté sur la détection cancer car c'est un cancer très répandu sur le globe et qui permet une grande chance de survie si la détection est faite assez tôt, tout comme précisé dans l'introduction, nous ne faisons

qu'apporter notre pierre à l'édifice de la société qui tend à être plus sûr et saine pour tous.

Semaine 4: Nous avons tous deux décidé de mettre en pause le projet à l'arrivée des différents examens que nous avions, les plannings de révisions étant assez lourds et chargés, il a été plus judicieux au vu de notre avancée de mettre le projet en suspens.

Semaine 5: Implémentation du code en session de Live Coding au lendemain des derniers cours/ rédaction du rapport final en co Writing sur Google Docs/ finalisation sur les derniers jours avec création des schémas et captures d'écrans à intégrer.

Nous avons utilisé, principalement, pour ce projet Python et ses librairies Panda, Numpy, Matplotlib et Seaborn. En plus de Python, nous avons utilisé JupyterLab à des fins plus pratiques que le python. Cela nous a permis de conforter nos compétences en Python déjà correctes au vu de notre enseignement à l'ECE et nos compétences en Machine Learning dans un aspect beaucoup plus pratique que ce que nous avons vu en début d'année.

Présentation et explication de notre utilisation du Machine Learning afin de détecter les cellules cancéreuses

Pour résoudre ce projet, nous utilisons Sklearn une bibliothèque libre Python destinée à l'apprentissage automatique
Visualisation ;
M veut dire Malignant qui signifie que la cellule est cancéreuse
B veut dire Benign qui signifie que la cellule est non cancéreuse.

Etape 1 : Analyse du data

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#Chargement du data
df = pd.read_csv('/Users/edouardbotherel/Desktop/data.csv')
```

[1]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	tex
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.08089	...	
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.1127	0.07400	...	

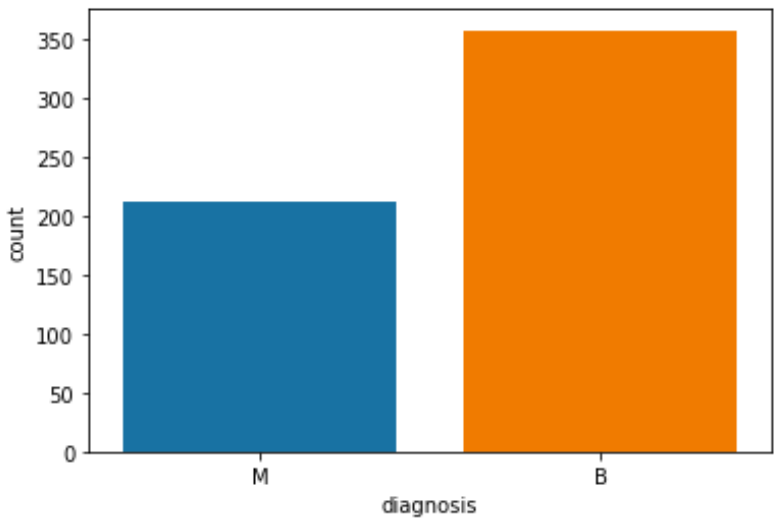
a. Comptage des cellules M & B parmis les patients :

[5]:
```
#Compte des M & B
df['diagnosis'].value_counts()
```

[5]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

[6]:
```
#Visualisation du compte
sns.countplot(df['diagnosis'],label="Count")
```

[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x103c75220>
```



Nous disposons de 357 patients dont les cellules ont été diagnostiquées comme cancéreuses.

b. **Encodage catégorique des M&B :**

Afin de faciliter la manipulation du data des clients, nous allons encoder les données, 1 = M et 0 =B

```
[8]: #Encodage catégorique des M&B (
     from sklearn.preprocessing import LabelEncoder
     labelencoder_Y = LabelEncoder()
     df.iloc[:,1]= labelencoder_Y.fit_transform(df.iloc[:,1].values)
     print(labelencoder_Y.fit_transform(df.iloc[:,1].values))
```

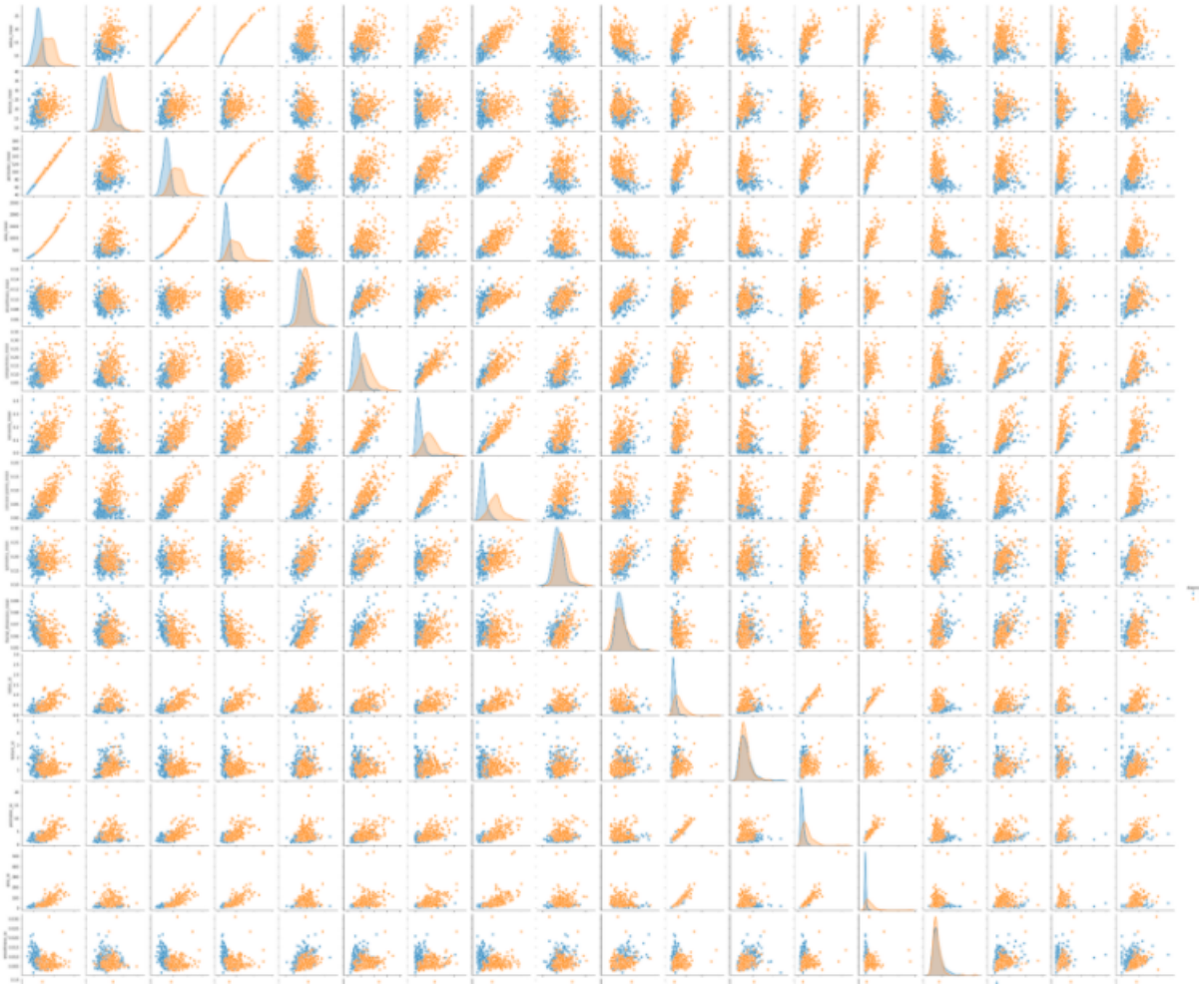
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0]
```

c. **Visualisation :**

Nous pouvons visualiser la répartition des cellules cancéreuses en fonction de chacun des 31 paramètres. Ceci nous aide dans la prochaine étape à détecter les corrélations entre les paramètres et le diagnostic.

```
sns.pairplot(df.iloc[:,1:20], hue='diagnosis')
```

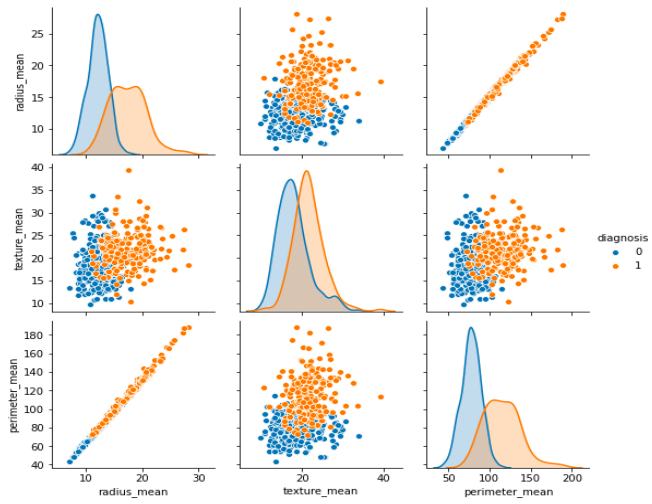
```
<seaborn.axisgrid.PairGrid at 0x14b00ed30>
```



Détail :

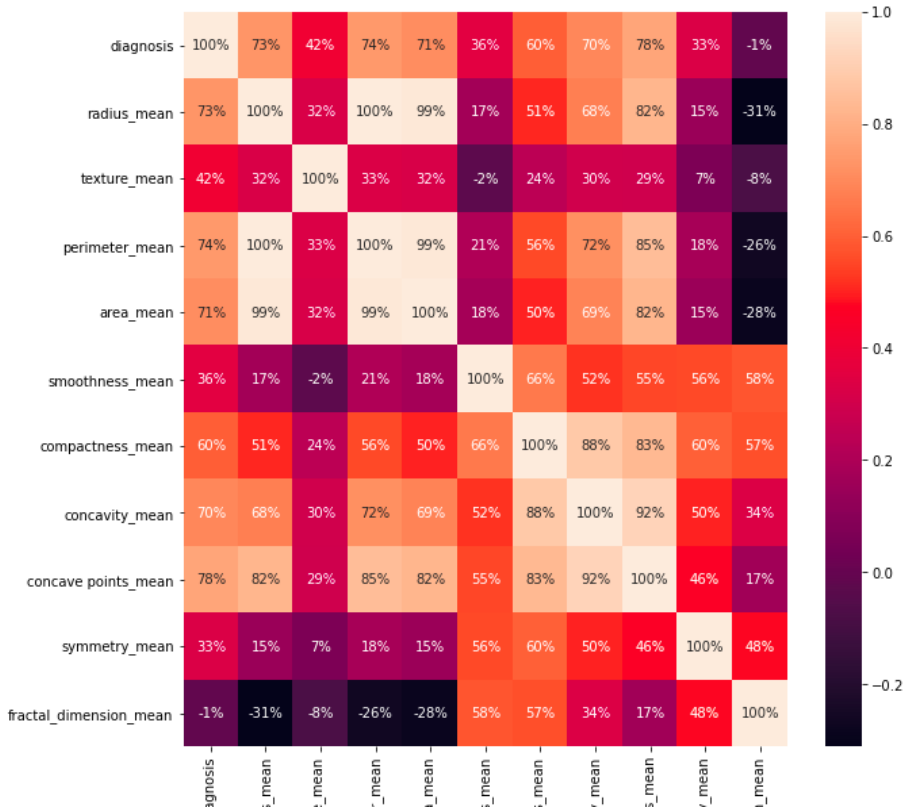
```
[12]: sns.pairplot(df.iloc[:,1:5], hue='diagnosis')
```

```
[12]: <seaborn.axisgrid.PairGrid at 0x1420b90d0>
```



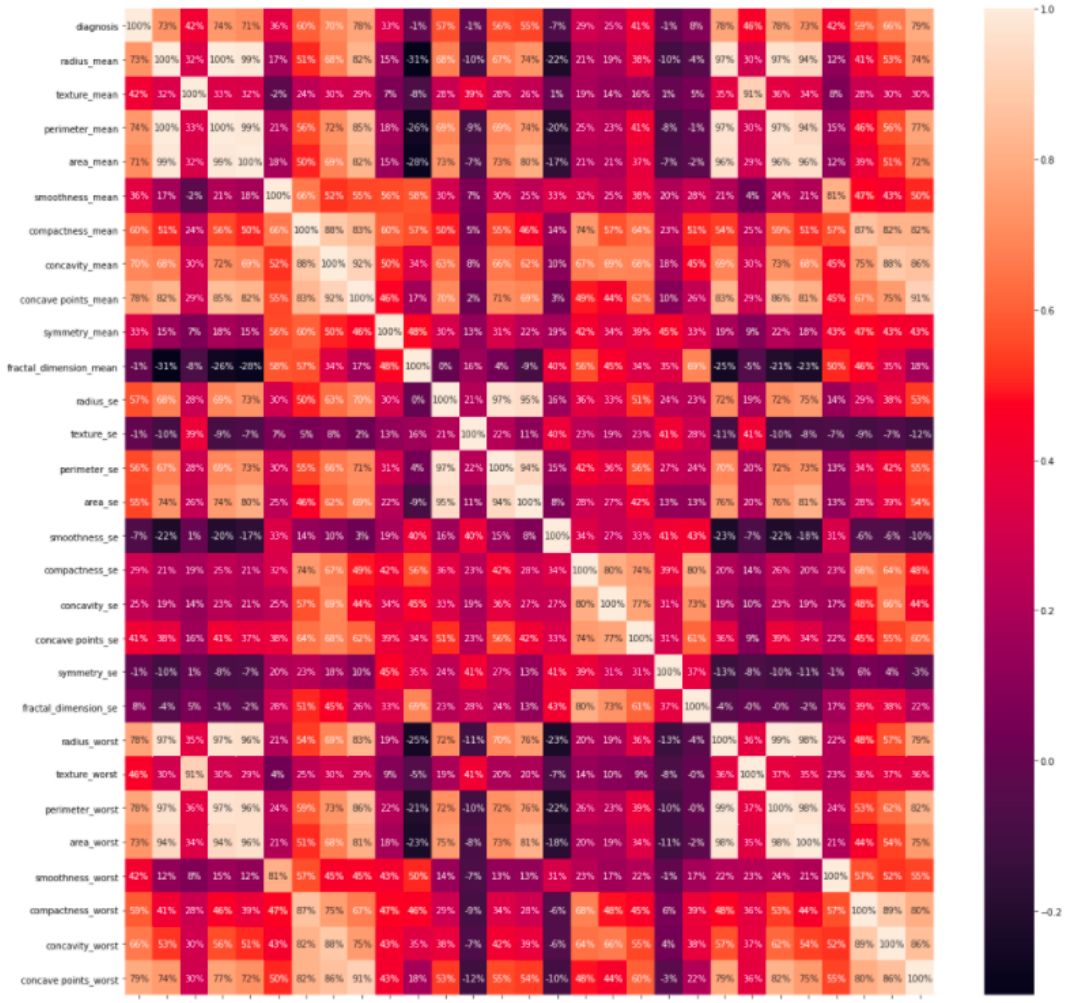
```
[19]: #Visualisation des corrélations
      plt.figure(figsize=(10,10))
      sns.heatmap(df.iloc[:,1:12].corr(),annot=True, fmt='.0%')

[19]: <matplotlib.axes._subplots.AxesSubplot at 0x103c75250>
```



d. Visualisation des corrélations :

Nous pouvons observer les corrélations entre chacun des paramètres et le diagnostic, en clair les plus corrélés jusqu'au sombre les moins corrélés.


diagnosis
radius_mean
texture_mean
perimeter_mean
area_mean
smoothness_mean
compactness_mean
concavity_mean
concave points_mean
symmetry_mean
fractal_dimension_mean
radius_se
texture_se
perimeter_se
area_se
smoothness_se
compactness_se
concavity_se
concave points_se
symmetry_se
fractal_dimension_se
radius_worst
texture_worst
perimeter_worst
area_worst
smoothness_worst
compactness_worst
concavity_worst
concave points_worst
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Etape 2 : Création du Modèle :

a. Séparation du Data en 2 matrices

```
[24]: #Séparation du data en 2 matrices X et Y
      #Contient tous les paramètres à part le diagnosis (paramètres détectant si patient a le cancer ou non )
      X=df.iloc[:,2:31].values
      #Diagnosis (Cancer ou pas)
      Y=df.iloc[:,1].values
```

```
[25]: #75% Training et 25% Testing
      from sklearn.model_selection import train_test_split
      X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.25,random_state=0)
```

```
[27]: from sklearn.preprocessing import StandardScaler
      sc= StandardScaler()
      X_train=sc.fit_transform(X_train)
      X_test=sc.fit_transform(X_test)

      X_train
```

```
[27]: array([[-0.65079907, -0.43057322, -0.68024847, ..., -0.69592933,
              -0.36433881,  0.32349851],
             [-0.82835341,  0.15226547, -0.82773762, ..., -1.29277423,
              -1.45036679,  0.62563098],
             [ 1.68277234,  2.18977235,  1.60009756, ...,  0.26255563,
               0.72504581, -0.51329768],
             ...,
             [-1.33114223, -0.22172269, -1.3242844 , ..., -0.78274313,
              -0.98806491, -0.69995543],
             [-1.25110186, -0.24600763, -1.28700242, ..., -1.36015587,
              -1.75887319, -1.56206114],
             [-0.74662205,  1.14066273, -0.72203706, ...,  0.47201917,
              -0.2860679 , -1.24094654]])
```

b. Modèles

```
[31]: def models(X_train,Y_train):

        #Using Logistic Regression (Model 1)
          from sklearn.linear_model import LogisticRegression
          log = LogisticRegression(random_state = 0)
          log.fit(X_train, Y_train) #trains model

        #Using KNeighborsClassifier (Model 2)
          from sklearn.neighbors import KNeighborsClassifier
          knn = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
          knn.fit(X_train, Y_train)

        #Using SVC linear (Model 3)
          from sklearn.svm import SVC
          svc_lin = SVC(kernel = 'linear', random_state = 0)
          svc_lin.fit(X_train, Y_train)

        #Using SVC rbf (Model 4)
          from sklearn.svm import SVC
          svc_rbf = SVC(kernel = 'rbf', random_state = 0)
          svc_rbf.fit(X_train, Y_train)

        #Using GaussianNB (Model 5)
          from sklearn.naive_bayes import GaussianNB
          gauss = GaussianNB()
          gauss.fit(X_train, Y_train)
```

Afin d'appliquer une méthode de machine learning, il est nécessaire de diviser le data en deux sets de matrices,

- l'une fonctionnant comme un booléen répondant oui ou non : le diagnostic
- La deuxième contenant toutes les variables pouvant être corrélées au diagnostic

Nous séparons ces deux matrices en 4 pour former nos 4 matrices de notre modèle :

75% : X_train, Y_train permettant d'entraîner le modèle
25% : X_test, Y_test permettant de conclure des résultats

Nous créons une fonction pour contenir de nombreux modèles différents (régression logistique, classificateur d'arbre de décision, classificateur de forêt aléatoire) pour effectuer la classification. Ce sont les modèles qui permettent de détecter si un patient a un cancer ou non.
Nous nous servons de méthodes de classement déjà implémentées dans la bibliothèque sklearn et cela nous permet de mesurer la vitesse et la précision de chaque modèle.

Résultats des précisions de chaque modèle :

```
[0]Logistic Regression Training Accuracy: 0.9906103286384976
[1]K Nearest Neighbor Training Accuracy: 0.9765258215962441
[2]Support Vector Machine (Linear Classifier) Training Accuracy: 0.9882629107981221
[3]Support Vector Machine (RBF Classifier) Training Accuracy: 0.9835680751173709
[4]Gaussian Naive Bayes Training Accuracy: 0.9507042253521126
[5]Decision Tree Classifier Training Accuracy: 1.0
[6]Random Forest Classifier Training Accuracy: 0.9953051643192489
```

Etape 3 : Test du modèle

```python
#Show other ways to get the classification accuracy & other metrics

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

for i in range(len(model)):
    print('Model ',i)
  #Check precision, recall, f1-score
    print( classification_report(Y_test, model[i].predict(X_test)) )
  #Another way to get the models accuracy on the test data
    print( accuracy_score(Y_test, model[i].predict(X_test)))
    print()#Print a new line
```

```
Model  0
              precision    recall  f1-score   support

           0       0.97      0.96      0.96        90
           1       0.93      0.94      0.93        53

    accuracy                           0.95       143
   macro avg       0.95      0.95      0.95       143
weighted avg       0.95      0.95      0.95       143

0.951048951048951

Model  1
              precision    recall  f1-score   support

           0       0.94      0.99      0.96        90
           1       0.98      0.89      0.93        53

    accuracy                           0.95       143
   macro avg       0.96      0.94      0.95       143
weighted avg       0.95      0.95      0.95       143

0.951048951048951

Model  2
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        90
           1       0.93      0.96      0.94        53

    accuracy                           0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143

0.958041958041958

Model  3
              precision    recall  f1-score   support

           0       0.97      0.98      0.97        90
           1       0.96      0.94      0.95        53

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.96      0.97      0.96       143

0.965034965034965

Model  4
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        90
           1       0.94      0.91      0.92        53

    accuracy                           0.94       143
   macro avg       0.94      0.94      0.94       143
weighted avg       0.94      0.94      0.94       143

0.9440559440559441

Model  5
              precision    recall  f1-score   support

           0       0.98      0.92      0.95        90
           1       0.88      0.96      0.92        53

    accuracy                           0.94       143
   macro avg       0.93      0.94      0.93       143
weighted avg       0.94      0.94      0.94       143
```

Le test du modèle s'effectue soit par l'utilisation des méthodes déjà implémentées ou en injectant une matrice "confuse" pour tester le modèle.

La matrice de confusion nous indique combien de patients chaque modèle a mal diagnostiqué (nombre de patients atteints de cancer qui ont été diagnostiqués à tort comme n'ayant pas de cancer aka faux négatif, et le nombre de patients qui n'avaient pas de cancer qui ont été mal diagnostiqués avec un cancer aka faux positif) et le nombre de diagnostics corrects, les vrais positifs et les vrais négatifs.

[59]:
```
from sklearn.metrics import confusion_matrix
cs=confusion_matrix(Y_test,model[0].predict(X_test))

TP=cs[0][0] #True positive
TN=cs[1][1] #True negative
FN=cs[1][0] #False negative
FP=cs[0][1] #False positive

print(cs)
print('Testing Accuracy', (TP+TN)/(TP+TN+FN+FP))
```

Conclusion :

Nous avons apprécié travailler sur ce projet car il nous a permis de mieux saisir les capacités d'analyse que nous offre l'intelligence artificielle ainsi que son étendue sur des domaines variés.

Au terme de notre projet, nous avons réussi à développer un programme permettant une classification et une détection d'anomalie en fonction de récurrences dans les corrélations.

Du fait que l'intelligence artificielle peut permettre une reconnaissance de "patterns" dans un échantillon, son application au domaine médical peut avoir un impact très prometteur. Elle permet de restreindre les risques et prévenir davantage en amont sur des corrélations détectant une anomalie.

Bibliographie

Machine learning applications in cancer prognosis and prediction - ScienceDirect
Understanding Cancer using Machine Learning | by Pier Paolo Ippolito | Towards Data Science
Cancers | Free Full-Text | Cancer Diagnosis Using Deep Learning: A Bibliographic Review | HTML (mdpi.com)
Data Science Bowl 2017 | Kaggle
AI-Immersion-Workshop/cntk_lightgbm_data_science_bowl.ipynb at master · microsoft/AI-Immersion-Workshop (github.com)
cbailes/awesome-ai-cancer: Awesome artificial intelligence in cancer diagnostics and oncology (github.com)