جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL
كلية العلوم والتقنيات
FACULTÉ DES SCIENCES ET TECHNIQUES

Prédiction de diabète

Rapport de mini projet

Réalisé par :
BENHA Bouchra

Encadré par :
Pr. Mohammed SABIRI

Table des matières

Introduction générale **1**

Etude de cas **1**

1. Problématique : 1
1. Solution proposée 1
2. Outils utilisés 2
3. Langage 2
4. Les librairies utilisées 2

Définition des données et des variables utilisées **3**

Réalisation **3**

1. Lecture des données 3
2. Visualisation des données 6
3. Matrice de corrélation 10

Conclusion : **11**

Introduction générale

Actuellement, on a découvert que les approches traditionnelles de la statistique ont des limites avec de grosses bases de données, car en présence de milliers ou de millions d'individus et de centaines ou de milliers de variables, on trouvera forcément un niveau élevé de redondance parmi ces variables. La fouille de données joue désormais un rôle important dans la prédiction des maladies dans l'industrie des soins de santé. La fouille de données est le processus de sélection, d'exploration et de modélisation de grandes quantités de données pour découvrir des modèles inconnus oudes relations utiles à l'analyste de données. Le récent rapport de L'Organisation mondiale de la santé (OMS) montre une progression remarquable le nombre de patients diabétiques et ce sera dans le même tendance dans les décennies à venir également. Identification précoce de diabète sucré est un défi important. L'exploration de données a joué un rôle important dans la recherche sur le diabète.La fouille de données serait un atout précieux pour les recherches sur le diabète car il peut déterrer les connaissances cachées d'une énorme quantité de diabète données connexes. Diverses techniques d'exploration de données aident la recherche de diabète et finalement améliorer la qualité des soins de santé pour les patients diabétiques.

Etude de cas

1. Problématique :

Le diabète est un problème de santé de plus en plus croissant en raison de notre mode de vie inactif. Aussi, Le diabète est l'une des maladies les plus meurtrières au monde. Les patients doivent se rendre dans un centre de diagnostic, consulter leur médecin et attendre un jour ou plus pour obtenir leur résultat. De plus, chaque fois qu'ils veulent obtenir leur rapport de diagnostic, ils doivent gaspiller leur argent en vain. Mais s'il on peut détecter à temps, un traitement médical approprié Les effets indésirables peuvent être évités. Pour aider à la détection précoce, la technologie peut être utilisée de manière très fiable et efficace.

1. Solution proposée

Grâce aux data mining, nous avons pu trouver une solution à ce problème en utilisant l'exploration de données. Il joue un rôle important dans la recherche sur le diabète car il a la capacité d'extraire des connaissances cachées d'une énorme quantité de données liées au diabète. Le but de cette recherche est de développer un système permettant de prédire si le patient est diabétique ou non. De plus, la prédiction précoce de la maladie conduit au traitement des patients avant qu'elle ne devienne critique.

2. Outils utilisés

Google Colab, un environnement de développement basé sur le cloud, peut être utilisé pour des tâches de data mining. Voici quelques utilisations courantes de Colab en data mining :

- ✓ Importation et prétraitement des données ;
- ✓ Exploration des données ;
- ✓ Modélisation et apprentissage automatique ;
- ✓ Analyse de texte ;
- ✓ Visualisation des résultats.

En résumé, Colab offre une plateforme flexible et puissante pour effectuer diverses tâches de data mining, que ce soit pour l'exploration, la modélisation, l'apprentissage automatique ou la visualisation des résultats.

3. Langage

Python est un langage de script de haut niveau, structuré et open source. Il est multi-usage. Il est unlangage de programmation très puissant utilisé en Data Mining pour faire de l'analyse statistique, la classification, le clustering et l'analyse prédictive.

4. Les librairies utilisées

Pandas

Est une autre bibliothèque Python utilisée pour la manipulation et l'analyse des données, le point fort de cette bibliothèque est qu'elle possède une fonctionnalité importante appelée nettoyage des données qui résout le problème du temps passé à nettoyer les données dans un projet d'apprentissage automatique.

Matplotlib

Est une bibliothèque complète pour créer des visualisations statiques, Animées et interactives en Python.

- Numpy

Est une extension du langage de programmation Python, destinée à manipuler des tableaux multidimensionnels.

- Seaborn

Est une bibliothèque de visualisation de données Python basée sur matplotlib. Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attrayants et informatifs

Définition des données et des variables utilisées

La base de données qu'on va utiliser provient de la plateforme web Kaggle. Cet ensemble de données provient à l'origine de l'Institut national du diabète et des maladies digestives et rénales L'objectif de l'ensemble de données est de prédire avec un diagnostique si un patient est atteint du diabète. Plusieurs contraintes ont été placées sur la sélection de ces instances dans une base de données plus grande. En particulier, tous les patients ici sont des femmes âgées d'au moins 21 ans.

Les ensembles de données comprennent plusieurs variables prédictives médicales et une variable cible, « Outcome ». Les variables prédictives comprennent le nombre de grossesses que le patient a eues, son IMC, son taux d'insuline, son âge, etc.

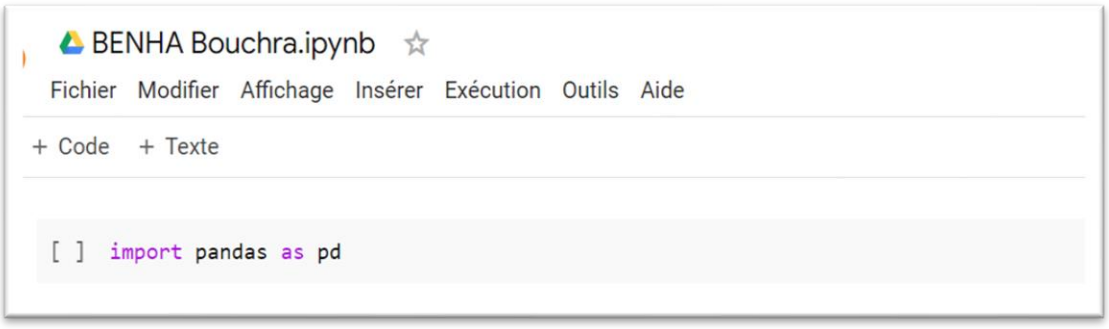
Chaque ligne représente un patient et les colonnes sont :

- ✓ Grossesses: nombre de fois enceintes
- ✓ Glucose: Concentration en glucose plasmatique 2 heures dans un test de tolérance au glucose par voie orale
- ✓ BloodPressure: pression artérielle diastolique (mm Hg)
- ✓ SkinThickness: Épaisseur du pli cutané des triceps (mm)
- ✓ Insuline: insuline sérique de 2 heures (mu U / ml)
- ✓ IMC: indice de masse corporelle (poids en kg / (taille en m) ^ 2)
- ✓ DiabetesPedigreeFunction: Fonction pedigree du diabète
- ✓ Age: Age (ans)
- ✓ Résultat: Variable de classe (0 ou 1).

Réalisation

1. Lecture des données

Grace à la LIB : 'pandas' on peut manipuler notre jeu de données : lire, écrire, stocker, visualiser, supprimer des lignes, des colonnes, créer des dataframes, ...etc.



- Après l'importation de base de données et la lecture des données

```
[ ]  # read the csv-formatted data file into a pandas dataframe
     df=pd.read_csv('diabetes-1.csv')
     # get shape of data frame
     print('Shape (n_rows,n_columns) of dataframe:',df.shape)
     # print top 5 rows of data frame
     df.head()
```

- On obtient :

```
Shape (n_rows,n_columns) of dataframe: (768, 9)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

- Sélection d'une (ou plusieurs) colonne(s) comme un nouveau dataframe et afficher uniquement les 5 premières lignes :

```
df2=df[['Pregnancies','BMI','Outcome']].head()
df2
```

	Pregnancies	BMI	Outcome
0	6	33.6	1
1	1	26.6	0
2	8	23.3	1
3	1	28.1	0
4	0	43.1	1

```
df3=df[['Insulin']].head()
df3
```

	Insulin
0	0
1	0
2	0
3	94
4	168

- Sélection des données qui satisfont une condition, et calculer leur quantité.

```
# Le premier élément est le nombre de lignes, le second est celui de colonnes:
print(df[df.BMI>30].shape)
# le premier élément (nbr lignes) a pour index 0, le second (nbr colonnes) 1.
print('The number of rows where BMI>30 = ',df[df.BMI>30].shape[0])
df[df.BMI<10].head()
```

```
(465, 9)
The number of rows where BMI>30 =  465
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
9	8	125	96	0	0	0.0	0.232	54	1
49	7	105	0	0	0	0.0	0.305	24	0
60	2	84	0	0	0	0.0	0.304	21	0
81	2	74	0	0	0	0.0	0.102	22	0
145	0	102	75	23	0	0.0	0.572	21	0

- Savoir si notre jeu de données possède des champs vides (non renseignés) ?

```
[8]  df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
df.notnull().sum()
```

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

- Afficher la liste des colonnes et le type de données de colonnes :

```
[10]  df.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

```
df.dtypes
```

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

- Avoir un résumé statistique de notre data frame :

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

- La moyenne de "SkinThickness" lorsque le outcome est égal à 1 :

```
df[df.Outcome==1].SkinThickness.mean()
```

```
22.16417910447761
```

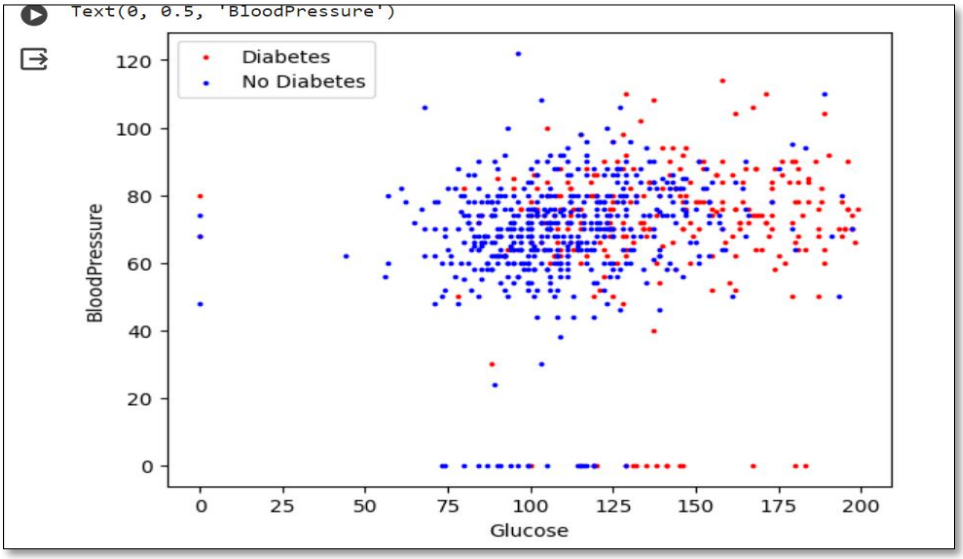
2. Visualisation des données

- Installation de la bibliothèque matplotlib

```
[14] # get a plotting library
     import matplotlib.pyplot as plt
     # make it interactive in the notebook
     # (Only for matplot online)
     %matplotlib inline
```

```
# plot Glucose vs BloodPressure and color points according to Outcome
plt.figure()
plt.scatter(df[df.Outcome==1].Glucose,df[df.Outcome==1].BloodPressure,label='Diabetes',color='r',s=3)
plt.scatter(df[df.Outcome==0].Glucose,df[df.Outcome==0].BloodPressure,label='No Diabetes',color='b',s=3)
plt.legend()
plt.xlabel('Glucose')
plt.ylabel('BloodPressure')
```

- On obtient le diagramme suivant :



→Remarquez qu'on peut constater que plus le taux de glucose est élevé, plus l'enregistrement est associé à un diabète (résultat = 1, points rouges), tandis que plus il est bas, et plus il est associé à l'absence du diabète (résultat = 0, points bleus).

→Notez également qu'il y a un ensemble de points de valeur 0 pour Glucose et un autre de valeur 0 pour BloodPressure. Cela n'a pas de sens physiquement. Il semble que ces données ont été remplies avec 0 alors que la valeur aurait dû être NULL. Vérifions combien de zéros apparaissent dans chaque colonne.

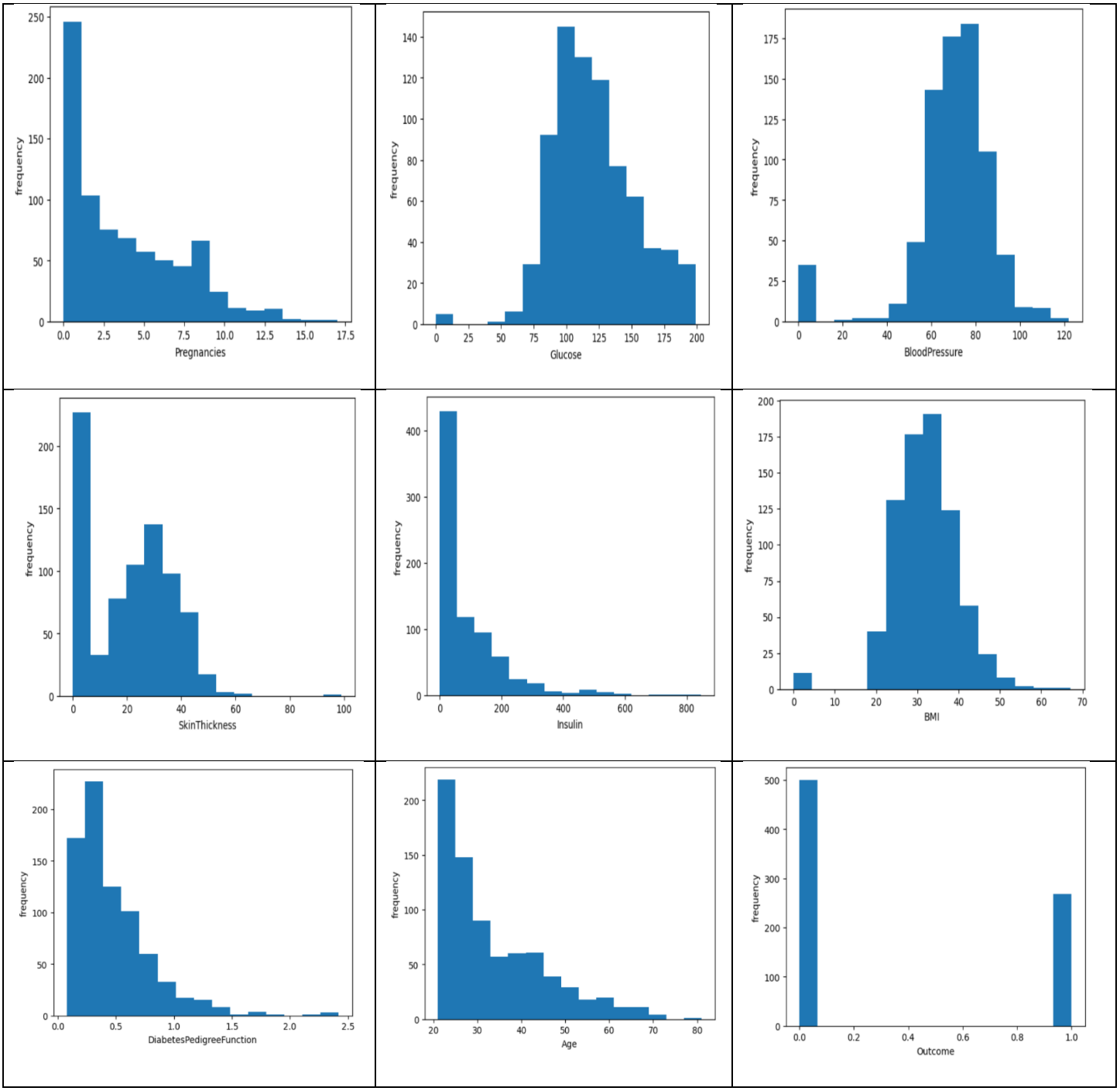
```
for c in df.columns:
    print('For column',c,' there are',df[df[c]==0][c].count(),'zero values.')
```

```
For column Pregnancies  there are 111 zero values.
For column Glucose  there are 5 zero values.
For column BloodPressure  there are 35 zero values.
For column SkinThickness  there are 227 zero values.
For column Insulin  there are 374 zero values.
For column BMI  there are 11 zero values.
For column DiabetesPedigreeFunction  there are 0 zero values.
For column Age  there are 0 zero values.
For column Outcome  there are 500 zero values.
```

→Pour certaines de ces colonnes, zéro a un sens, comme pour le nombre de grossesses (Pregnancies) et le résultat (Outcome). Mais pour d'autres colonnes, comme BloodPressure ou BMI, zéro n'a absolument aucun sens. Examinons de plus près les données en dessinant un histogramme des valeurs des données pour chaque colonne.

```
for c in df.columns:
    plt.figure()
    plt.hist(df[c],bins=15)
    plt.xlabel(c)
    plt.ylabel('frequency')
    plt.show()
```

- Après l'exécution de code ci-dessus on obtient les histogrammes suivants :



→ À partir de ces histogrammes, il semble que bon nombre de valeurs nulles sont en effet des données manquantes qui auraient dû être étiquetées NULL, et devront être prises en compte avant que nous formions un modèle pour classer les données.

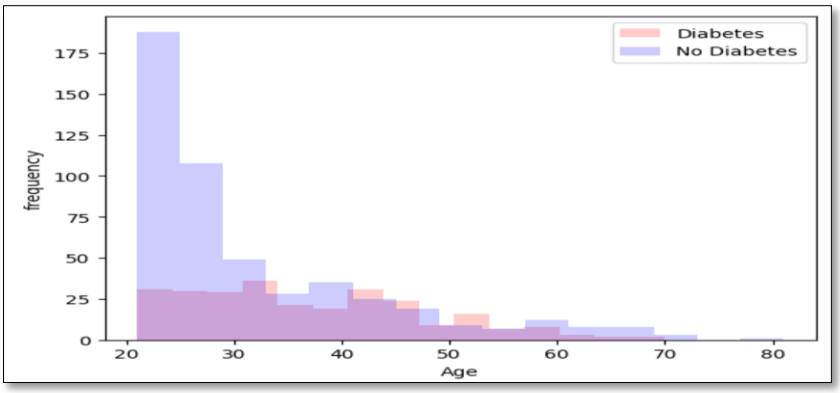
→ De plus, nous pouvons remarquer que la colonne Insuline n'a que 374 valeurs renseignées (sur un total de 768 lignes), soit presque 50% des valeurs sont des zéros.

→ Lors de la construction d'un modèle, nous allons d'abord supprimer la colonne d'insuline, car beaucoup de valeurs sont manquantes. Ensuite, nous supprimerons (remplacer) les zéros dans les colonnes où zéro n'a pas de sens. Nous ferons le choix d'utiliser la moyenne (moyenne) des valeurs non nulles dans chaque colonne pour remplacer les valeurs nulles. Nous traiterons dans l'étape 3 ces cas d'inconsistance de ces données.

- Chercher visuellement un attribut qui permetterait de séparer correctement les valeurs de Outcome (la colonne "target" ou "label"). Quelle valeur de cet attribut vous permet alors de découper le dataframe en diabète et non-diabète ?

```
# example: plot histograms of Age for Outcome=1 and Outcome=0.
plt.figure()
plt.hist(df[df.Outcome==1]['Age'],bins=15,label='Diabetes',color='r',alpha=0.2)
plt.hist(df[df.Outcome==0]['Age'],bins=15,label='No Diabetes',color='b',alpha=0.2)
plt.xlabel('Age')
plt.ylabel('frequency')
plt.legend()
plt.show()
```

- On obtient :



- Choisir la colonne (l'attribut) qui, pour vous, maximise le taux de précision (accuracy). On fait l'hypothèse que l'âge est l'attribut discriminant, avec un seuil à 30 ans.

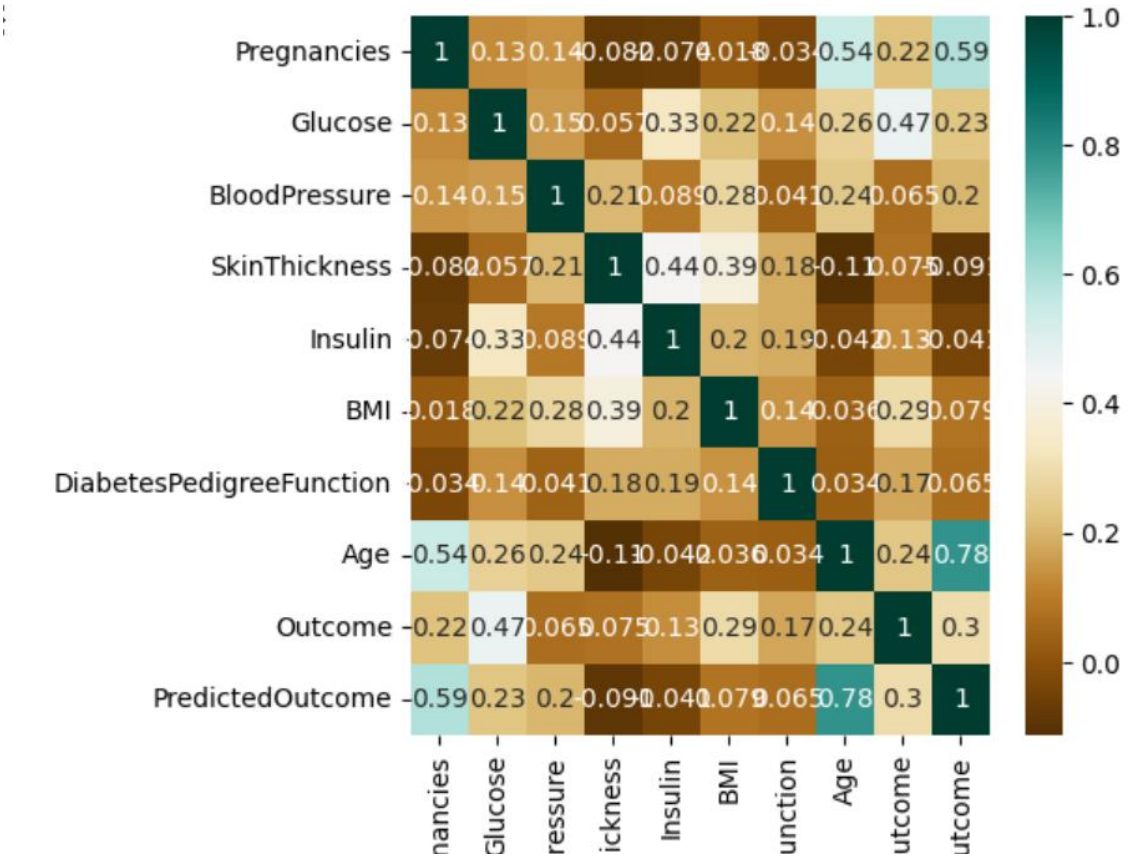
```
import numpy as np
# example
# create a new column in the data frame with the predicted outcome based on your split (here, Age<30 means outcome=0, otherwise outcome=1)
df['PredictedOutcome']=np.where(df.Age<30,0,1) # np.where(condition, value if true, value if false)
# calculate accuracy
N_correct=df[df.PredictedOutcome==df.Outcome].shape[0]
N_total=df.shape[0]
accuracy=N_correct/N_total
print('number of correct examples =',N_correct)
print('number of examples in total =',N_total)
print('accuracy =',accuracy)
```

```
number of correct examples = 496
number of examples in total = 768
accuracy = 0.6458333333333334
```

3. Matrice de corrélation

```
import seaborn as sns
corrmat=df.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(10,10))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="BrBG")
```

On obtient la matrice de corrélation suivante :



Conclusion :

La réalisation de ce projet a été pour nous l'occasion de consolider les connaissances acquises durant le cours de Data Mining, enrichir notre expérience en matière d'analyse et comprendre l'utilité et les perspectives de l'exploration de données. Néanmoins, ce projet ne s'est pas terminé sans difficulté. La base de données ayant fait l'objet d'autre projet, malgré notre volonté de faire une étude totalement différente de la première, la limitation des données nous faisait énormément dans le choix des variables lors de l'application des méthodes. Enfin, réaliser ce projet a été tout à fait intéressant et productif. En effet, nous avons pu confronter nos idées sur l'étude et effectuer ainsi une analyse plus détaillée.