# Project2___EDA-ML

February 13, 2024

## 1 Predict Diabetes Data

#### 1.0.1 import required packages

[2]:
```
import pandas as pd #import panda
import numpy as np  #import numpy
import matplotlib.pyplot as plt #import pyplot
import seaborn as sns  #import seaborn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score , confusion_matrix ,␣
↪classification_report
from sklearn.tree import DecisionTreeClassifier
```

#### 1.0.2 import dataset from csv

[3]:
```
df = pd.read_csv('./diabetes.csv')
df.head()
```

[3]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[4]:
```
df.shape  #dimenssion
```

[4]:
```
(768, 9)
```

### 1.0.3 Check data type

[4]:
```
df.dtypes  #types of column
```

[4]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

### 1.0.4 Find missing data

[5]:
```
df.isnull().sum()  #check missing data
```

[5]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

### 1.0.5 Statistics to verify the quality of data

[6]:
```
outliers = df.loc[df['Pregnancies'] < 0 , 'Pregnancies']  #check for anomalies
outliers
```

[6]:
```
Series([], Name: Pregnancies, dtype: int64)
```

[7]:
```
df.describe()
```

[7]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000
```

| | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|
| count | 768.000000 | 768.000000 | 768.000000 | 768.000000 |
| mean | 31.992578 | 0.471876 | 33.240885 | 0.348958 |
| std | 7.884160 | 0.331329 | 11.760232 | 0.476951 |
| min | 0.000000 | 0.078000 | 21.000000 | 0.000000 |
| 25% | 27.300000 | 0.243750 | 24.000000 | 0.000000 |
| 50% | 32.000000 | 0.372500 | 29.000000 | 0.000000 |
| 75% | 36.600000 | 0.626250 | 41.000000 | 1.000000 |
| max | 67.100000 | 2.420000 | 81.000000 | 1.000000 |

### 1.0.6 Calculate the correlation between variables

[8]:
```
df.corr()  #calculate correlation
```

[8]:

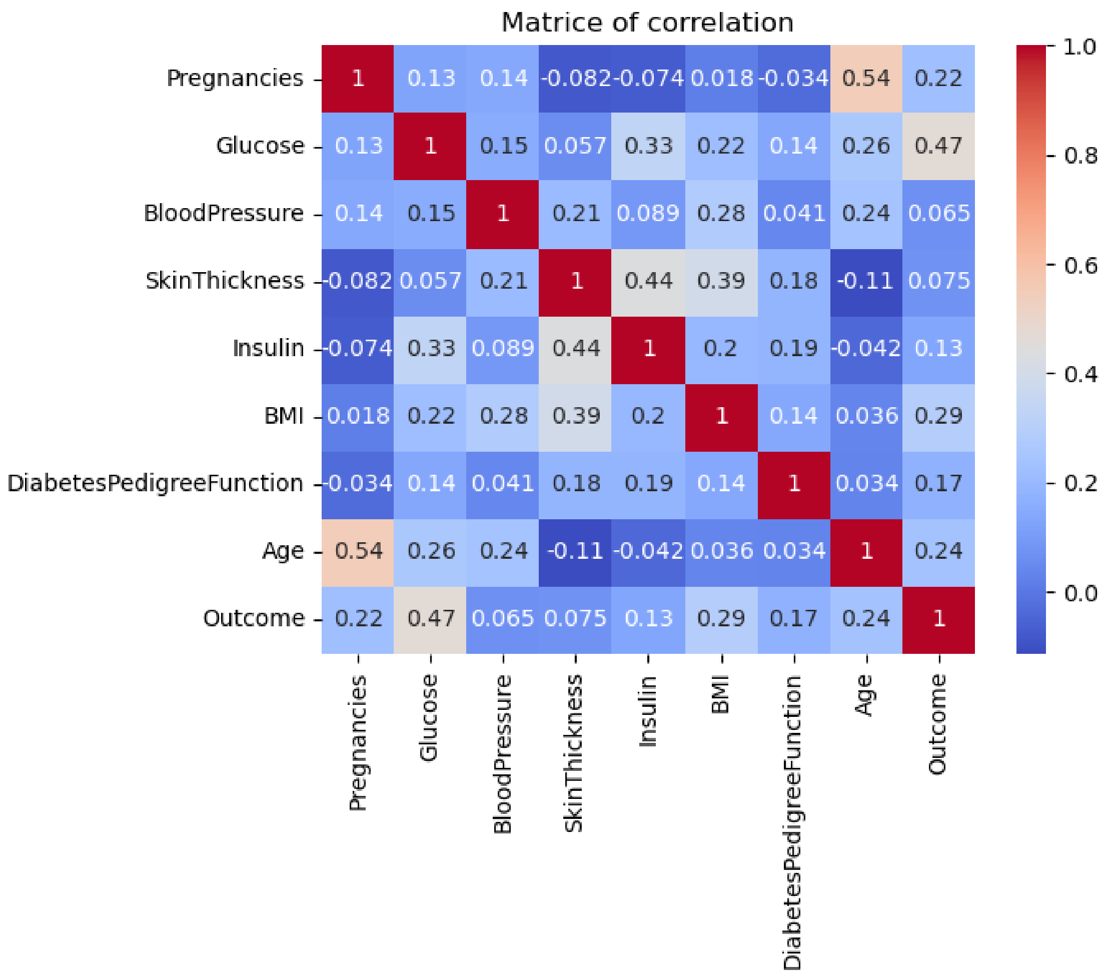
| | Pregnancies | Glucose | BloodPressure | SkinThickness | \ |
|---|---|---|---|---|---|
| Pregnancies | 1.000000 | 0.129459 | 0.141282 | -0.081672 | |
| Glucose | 0.129459 | 1.000000 | 0.152590 | 0.057328 | |
| BloodPressure | 0.141282 | 0.152590 | 1.000000 | 0.207371 | |
| SkinThickness | -0.081672 | 0.057328 | 0.207371 | 1.000000 | |
| Insulin | -0.073535 | 0.331357 | 0.088933 | 0.436783 | |
| BMI | 0.017683 | 0.221071 | 0.281805 | 0.392573 | |
| DiabetesPedigreeFunction | -0.033523 | 0.137337 | 0.041265 | 0.183928 | |
| Age | 0.544341 | 0.263514 | 0.239528 | -0.113970 | |
| Outcome | 0.221898 | 0.466581 | 0.065068 | 0.074752 | |

| | Insulin | BMI | DiabetesPedigreeFunction | \ |
|---|---|---|---|---|
| Pregnancies | -0.073535 | 0.017683 | -0.033523 | |
| Glucose | 0.331357 | 0.221071 | 0.137337 | |
| BloodPressure | 0.088933 | 0.281805 | 0.041265 | |
| SkinThickness | 0.436783 | 0.392573 | 0.183928 | |
| Insulin | 1.000000 | 0.197859 | 0.185071 | |
| BMI | 0.197859 | 1.000000 | 0.140647 | |
| DiabetesPedigreeFunction | 0.185071 | 0.140647 | 1.000000 | |
| Age | -0.042163 | 0.036242 | 0.033561 | |
| Outcome | 0.130548 | 0.292695 | 0.173844 | |

| | Age | Outcome |
|---|---|---|
| Pregnancies | 0.544341 | 0.221898 |
| Glucose | 0.263514 | 0.466581 |
| BloodPressure | 0.239528 | 0.065068 |
| SkinThickness | -0.113970 | 0.074752 |
| Insulin | -0.042163 | 0.130548 |
| BMI | 0.036242 | 0.292695 |
| DiabetesPedigreeFunction | 0.033561 | 0.173844 |
| Age | 1.000000 | 0.238356 |
| Outcome | 0.238356 | 1.000000 |

### 1.0.7 plot the correlation

[9]:
```
sns.heatmap(df.corr(), annot=True , cmap='coolwarm')
plt.title('Matrice of correlation')
plt.show()
```




### 1.0.8 split dataframe into actual data and labels

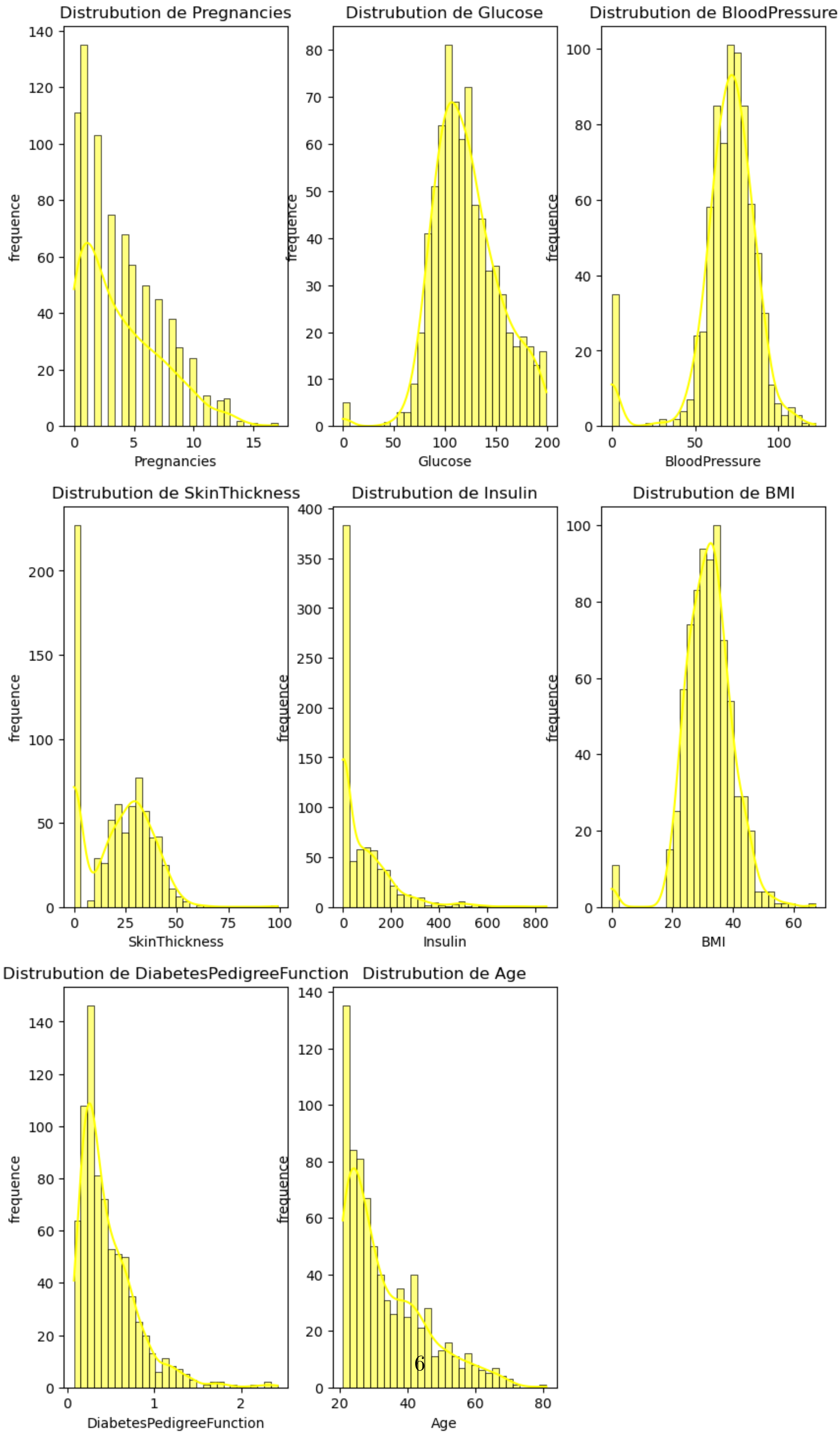
[10]:
```
data = df.iloc[: , :-1]
labels = df.iloc[: , -1]
```

### 1.0.9 verifie distribution and values

[11]:
```
plt.figure(figsize=(10,18))

for i,col in enumerate(data , 1):
    plt.subplot(3,3,i)
```

```
    sns.histplot(df[col], bins = 30 ,  kde=True , color = 'yellow')
    plt.title('Distrubution de '+col)
    plt.xlabel(col)
    plt.ylabel('frequence')
```


Distrubution de Pregnancies
frequence
Pregnancies
Distrubution de Glucose
frequence
Glucose
Distrubution de BloodPressure
frequence
BloodPressure
Distrubution de SkinThickness
frequence
SkinThickness
Distrubution de Insulin
frequence
Insulin
Distrubution de BMI
frequence
BMI
Distrubution de DiabetesPedigreeFunction
frequence
DiabetesPedigreeFunction
Distrubution de Age
frequence
Age

### 1.0.10 correcting outliers

[12]:
```
df.loc[df['Glucose']==0 ,'Glucose'].count()
outlier_col = ['Glucose', 'BloodPressure','SkinThickness' , 'Insulin', 'BMI']
```

[13]:
```
for i in outlier_col:
    df[i] = df[i].replace(0 , np.nan)

df.isnull().sum()
```

[13]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

[14]:
```
for i in outlier_col:
    df[i].fillna(df[i].median() , inplace=True)

df.isnull().sum()
```

[14]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

### 1.0.11 Split data into training data and test data

[16]:
```
data_train , data_test , labels_train , labels_test = train_test_split(data ,␣
↪labels , test_size=0.2 , random_state=0)
data_train.shape , labels_train.shape , data_test.shape , labels_test.shape
```

[16]:
```
((614, 8), (614,), (154, 8), (154,))
```

### 1.0.12 Train Logistic regression model

[17]:
```
lrmodel = LogisticRegression(random_state=0 , max_iter=700)
lrmodel.fit(data_train , labels_train)
```

[17]:
```
LogisticRegression(max_iter=700, random_state=0)
```

### 1.0.13 Calculate Accuracy score

[18]:
```
accuracy1= lrmodel.score(data_test , labels_test)
labels_predicted = lrmodel.predict(data_test)
accuracy2 = accuracy_score(labels_test , labels_predicted)
accuracy1 , accuracy2
```

[18]:
```
(0.8246753246753247, 0.8246753246753247)
```

### 1.0.14 Calculate metrics

[19]:
```
#confusion matrix
confusion_mat = confusion_matrix(labels_test , labels_predicted)
confusion_mat
```

[19]:
```
array([[98,  9],
       [18, 29]], dtype=int64)
```

[20]:
```
#report cassification
report = classification_report(labels_test , labels_predicted)
report
```

[20]:
```
'              precision    recall  f1-score   support\n\n           0
0.84      0.92      0.88       107\n           1       0.76      0.62      0.68
47\n\n    accuracy                           0.82       154\n   macro avg
0.80      0.77      0.78       154\nweighted avg       0.82      0.82      0.82
154\n'
```

[21]:
```
report = classification_report(labels_test , labels_predicted, output_dict=True)
for label, metrics in report.items():
    if label.isdigit():  # Check if the key is a class label
        print(f'Class: {label}')
        for metric, value in metrics.items():
            if metric != 'support':  # Exclude 'support' from printing
                print(f'{metric.capitalize()}: {value:.2f}')
        print('\n')
```

```
Class: 0
Precision: 0.84
Recall: 0.92
F1-score: 0.88
```

```
Class: 1
Precision: 0.76
Recall: 0.62
F1-score: 0.68
```

### 1.0.15 Train Decision tree classifier model

[22]:
```
dtmodel = DecisionTreeClassifier()
dtmodel.fit(data_train , labels_train)

labels_predicted_2 = dtmodel.predict(data_test)

score_2 = dtmodel.score(data_test , labels_test)
accuracy_2 = accuracy_score(labels_test , labels_predicted)
confusion_mat_2 = confusion_matrix(labels_test , labels_predicted)
score_2, accuracy_2, confusion_mat_2
```

[22]:
```
(0.7792207792207793,
 0.8246753246753247,
 array([[98,  9],
        [18, 29]], dtype=int64))
```

### 1.0.16 Calculate metrics

[23]:
```
#repport classification
report = classification_report(labels_test , labels_predicted_2,␣
 ↪output_dict=True)
for label, metrics in report.items():
    if label.isdigit():  # Check if the key is a class label
        print(f'Class: {label}')
        for metric, value in metrics.items():
            if metric != 'support':  # Exclude 'support' from printing
                print(f'{metric.capitalize()}: {value:.2f}')
        print('\n')
```

```
Class: 0
Precision: 0.88
Recall: 0.79
F1-score: 0.83


Class: 1
Precision: 0.61
Recall: 0.74
F1-score: 0.67
```

[26]:
```python
#importance of each variable
dtmodel.feature_importances_
```

[26]:
```
array([0.05589521, 0.31432593, 0.11775423, 0.01689698, 0.04949311,
       0.17705803, 0.13027364, 0.13830288])
```

[27]:
```python
for col , feat in zip(df.columns , dtmodel.feature_importances_):
    print('features : ',col , ' \t\t\t | importance : ' , feat)
```

```
features :  Pregnancies                          | importance :
0.05589521436556582
features :  Glucose                      | importance :  0.3143259288118616
features :  BloodPressure                        | importance :
0.11775422684549706
features :  SkinThickness                        | importance :
0.01689697569805338
features :  Insulin                      | importance :  0.04949310924496884
features :  BMI                          | importance :  0.17705803194112185
features :  DiabetesPedigreeFunction                     | importance :
0.1302736363771514
features :  Age                          | importance :  0.13830287671578
```