In [1]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
data = pd.read_csv('diabetes.csv')
```

In [3]:

```
data.head()
```

Out[3]:

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

In [5]:

```
data.isnull().sum()
```

Out[5]:

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

In [6]:

```
zero_not_accepted = ['Glucose','BloodPressure','SkinThickness','BMI','Insulin']
```

In [7]:

```
for col in zero_not_accepted:
    data[col]= data[col].replace(0,np.NaN)
    mean = int(data[col].mean(skipna=True))
    data[col] = data[col].replace(np.NaN,mean)
```

In [9]:

```
data.isnull().sum()
```

Out[9]:

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

In [11]:

```
data.describe()
```

Out[11]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	7
mean	3.845052	121.682292	72.386719	29.108073	155.28125	32.450911	
std	3.369578	30.435999	12.096642	8.791221	85.02155	6.875366	
min	0.000000	44.000000	24.000000	7.000000	14.00000	18.200000	
25%	1.000000	99.750000	64.000000	25.000000	121.50000	27.500000	
50%	3.000000	117.000000	72.000000	29.000000	155.00000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	155.00000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.00000	67.100000	

In [12]:

```
X = data.iloc[:,0:8]
```

In [13]:

```
X
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age
0	6	148.0	72.0	35.0	155.0	33.6	0.627	50
1	1	85.0	66.0	29.0	155.0	26.6	0.351	31
2	8	183.0	64.0	29.0	155.0	23.3	0.672	32
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.0	32.9	0.171	63
764	2	122.0	70.0	27.0	155.0	36.8	0.340	27
765	5	121.0	72.0	23.0	112.0	26.2	0.245	30
766	1	126.0	60.0	29.0	155.0	30.1	0.349	47
767	1	93.0	70.0	31.0	155.0	30.4	0.315	23

768 rows × 8 columns

In [14]:

```
y = data.iloc[:,8]
```

In [15]:

```
y
```

Out[15]:

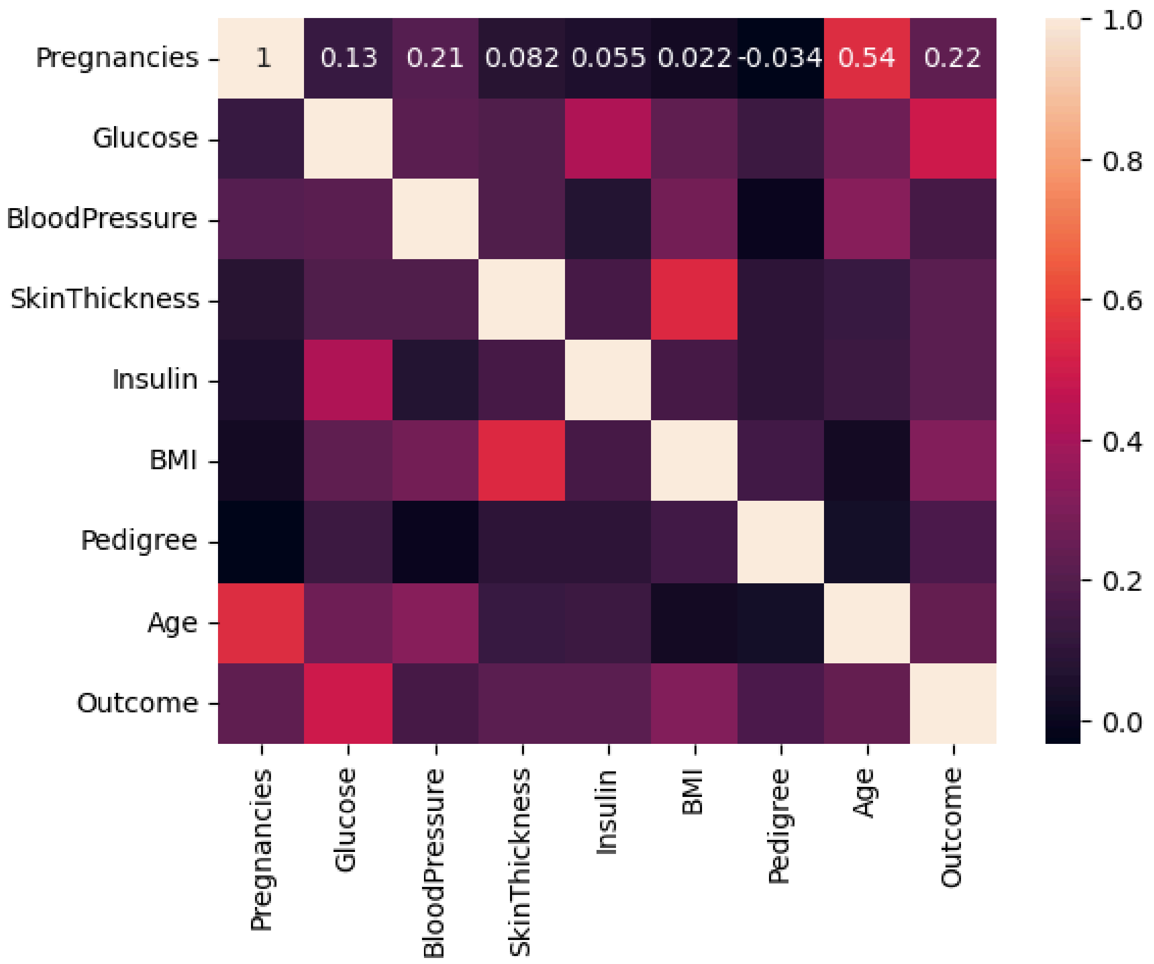
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [17]:

```
sns.heatmap(data.corr(),annot=True)
```

Out[17]:

```
<Axes: >
```



```
In [18]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=
```

```
In [19]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

```
In [20]: classifier = KNeighborsClassifier(n_neighbors=11,p=2,metric='euclidean')
```

```
In [21]: classifier.fit(X_train,y_train)
```

```
Out[21]: ▾ KNeighborsClassifier
         KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

```
In [22]: y_pred = classifier.predict(X_test)
```

```
In [23]: conf_matrix = confusion_matrix(y_test,y_pred)
         print(conf_matrix)
         print(f1_score(y_test,y_pred))
```

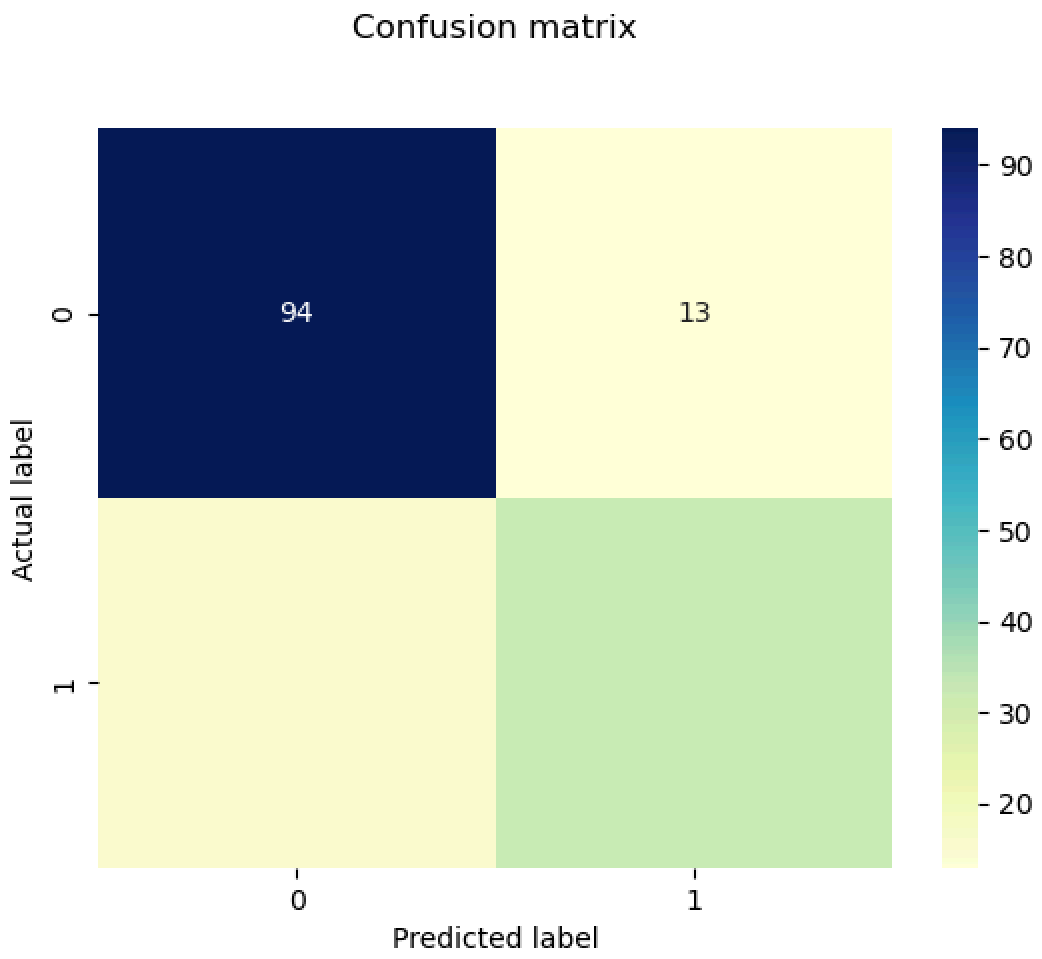
```
[[94 13]
 [15 32]]
0.6956521739130435
```

```
In [31]: from sklearn.metrics import confusion_matrix
         from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_sc
         y_pred = classifier.predict(X_test)
```

```
cnf_matrix = confusion_matrix(y_test, y_pred)
```

```
In [32]: p = sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
         plt.title('Confusion matrix', y=1.1)
         plt.ylabel('Actual label')
         plt.xlabel('Predicted label')
```

```
Out[32]: Text(0.5, 23.52222222222222, 'Predicted label')
```



```
In [33]: accuracy_score(y_test, y_pred)

Out[33]: 0.8181818181818182
```

```
In [34]: precision_score(y_test, y_pred)

Out[34]: 0.7111111111111111
```

```
In [35]: recall_score(y_test, y_pred)

Out[35]: 0.6808510638297872
```

```
In [36]: f1_score(y_test, y_pred)

Out[36]: 0.6956521739130435
```

```
In [ ]:
```