In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
```

In [2]:

```
data = pd.read_csv('pima-indians-diabetes.csv')
df = pd.DataFrame(data=data.values, columns=['Pregnancies', 'Glucose', 'Blood_Pr
df.head()
```

Out[2]:

	Pregnancies	Glucose	Blood_Pressure	Skin_Thickness	Insulin	Bmi	Diabetes_Pedigre
0	1.0	85.0	66.0	29.0	0.0	26.6	0.35
1	8.0	183.0	64.0	0.0	0.0	23.3	0.67
2	1.0	89.0	66.0	23.0	94.0	28.1	0.16
3	0.0	137.0	40.0	35.0	168.0	43.1	2.28
4	5.0	116.0	74.0	0.0	0.0	25.6	0.20

In [3]:

```
ohe = LabelEncoder()
Y = df['Prediction'].astype(int)
```

In [4]:

```
X = df.drop(columns='Prediction')

X = np.hstack((np.ones((X.shape[0],1)),X))
sc = StandardScaler()
X = sc.fit_transform(X)
```

In [5]:

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3)
```

In [6]:

```
clf = RandomForestClassifier()
model = clf.fit(x_train, y_train)
pred = model.predict(x_test)
```

In [7]:

```
y_test.index = np.arange(len(y_test.index))
y_test.index
```

Out[7]:

```
Index([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,
       ...
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230],
      dtype='int64', length=231)
```

In [8]:

```
for i in range(len(y_test)):
    print(f"{y_test[i]}   and  {pred[i]}")
```

```
1   and  1
1   and  1
0   and  0
0   and  0
1   and  1
1   and  0
0   and  1
1   and  0
0   and  0
1   and  1
0   and  0
1   and  1
1   and  1
0   and  0
0   and  0
0   and  0
1   and  1
0   and  1
0   and  0
0   and  0
0   and  1
0   and  0
0   and  1
0   and  0
0   and  0
0   and  0
0   and  0
0   and  1
0   and  0
1   and  1
0   and  0
0   and  1
0   and  0
0   and  1
1   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
0   and  1
1   and  0
1   and  0
0   and  0
1   and  1
0   and  1
1   and  0
1   and  0
1   and  1
1   and  1
0   and  0
1   and  1
0   and  0
1   and  1
0   and  0
0   and  0
1   and  1
1   and  0
0   and  0
```

```
0   and  0
1   and  0
1   and  0
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
0   and  0
0   and  0
0   and  0
0   and  0
0   and  1
0   and  0
0   and  0
0   and  0
1   and  0
1   and  1
0   and  0
0   and  0
0   and  0
1   and  1
0   and  1
0   and  0
0   and  0
0   and  1
0   and  0
1   and  1
0   and  0
0   and  0
0   and  0
1   and  1
1   and  0
0   and  0
1   and  0
0   and  1
0   and  0
1   and  1
0   and  0
1   and  1
0   and  1
1   and  1
0   and  1
0   and  1
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  1
0   and  0
0   and  0
1   and  1
0   and  0
0   and  0
1   and  0
0   and  0
0   and  0
0   and  0
1   and  0
```

```
1   and  1
0   and  0
1   and  0
0   and  0
1   and  1
1   and  1
0   and  0
1   and  0
0   and  0
1   and  1
1   and  1
0   and  1
0   and  1
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
0   and  0
0   and  0
0   and  0
1   and  1
0   and  0
1   and  0
1   and  1
1   and  1
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
1   and  1
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
0   and  0
0   and  0
0   and  0
0   and  0
0   and  1
1   and  0
0   and  0
1   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
0   and  0
```

```
0   and  0
1   and  1
1   and  0
0   and  0
0   and  0
0   and  0
0   and  1
1   and  1
1   and  1
1   and  0
0   and  0
1   and  1
0   and  0
0   and  0
0   and  0
1   and  0
0   and  0
1   and  0
0   and  0
1   and  0
1   and  1
0   and  0
0   and  0
0   and  0
1   and  0
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
0   and  0
1   and  1
0   and  0
1   and  0
0   and  0
0   and  0
1   and  1
1   and  1
0   and  0
0   and  0
1   and  0
0   and  0
0   and  0
0   and  0
0   and  0
1   and  1
0   and  0
1   and  1
1   and  1
1   and  1
1   and  0
```

In [9]:

```
sklearn_accuracy = accuracy_score(y_test, pred)
print(f"l'accurece score est {sklearn_accuracy * 100}")
```

```
l'accurece score est 78.35497835497836
```

In [ ]:

```
import pickle
```

```python
with open('indians_diabete_randomForest.pkl', 'wb') as f:
    pickle.dump(model, f)
```

In [ ]:

```
```

In [ ]:

```
```