In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
data=pd.read_csv('/Users/preetammukherjee/Desktop/diabetes.csv')
```

In [3]:
```
data.head(3)
```

Out[3]:

	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

In [5]:
```
predct =dict(zip(data.Pregnancies.unique(),data.Age.unique()))
predct
```

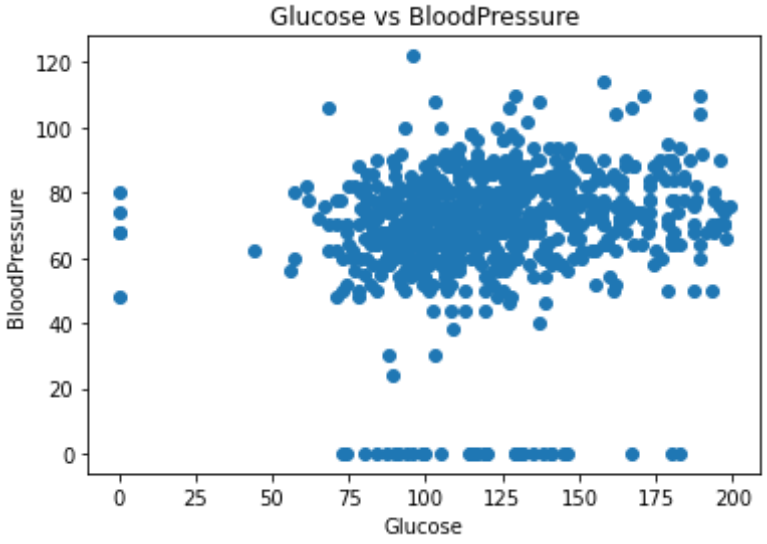
Out[5]:
```
{6: 50,
 1: 31,
 8: 32,
 0: 21,
 5: 33,
 3: 30,
 10: 26,
 2: 29,
 4: 53,
 7: 54,
 9: 34,
 11: 57,
 13: 59,
 15: 51,
 17: 27,
 12: 41,
 14: 43}
```

In [7]:
```
data['Pregnancies'].value_counts()
```

Out[7]:
```
1     135
0     111
2     103
3      75
4      68
5      57
6      50
7      45
8      38
9      28
10     24
11     11
13     10
12      9
14      2
15      1
17      1
Name: Pregnancies, dtype: int64
```

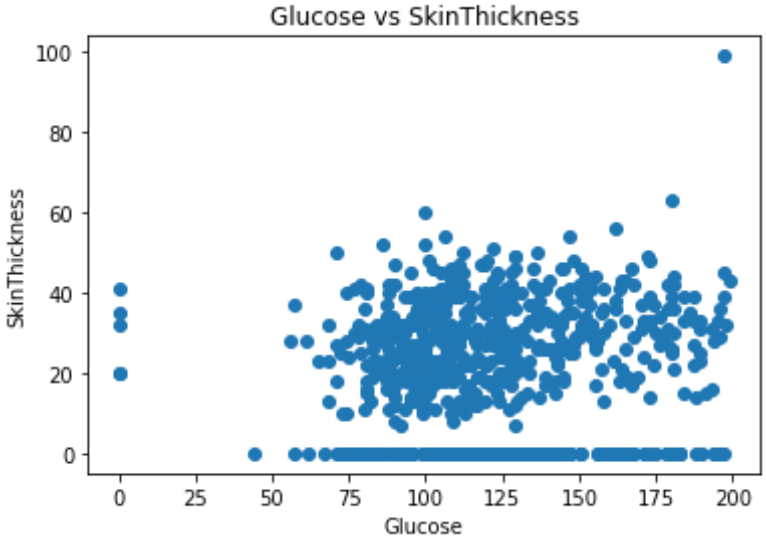
In [8]:
```
plt.scatter(data['Glucose'],data['BloodPressure'])
plt.xlabel('Glucose')
plt.ylabel('BloodPressure')
plt.title('Glucose vs BloodPressure')
```

Out[8]:
```
Text(0.5, 1.0, 'Glucose vs BloodPressure')
```



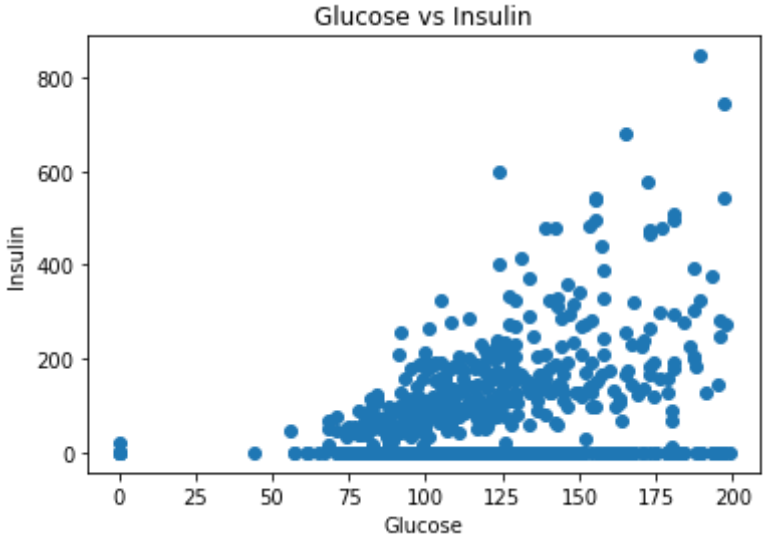
In [9]:
```
plt.scatter(data['Glucose'],data['SkinThickness'])
plt.xlabel('Glucose')
plt.ylabel('SkinThickness')
plt.title('Glucose vs SkinThickness')
```

Out[9]:
```
Text(0.5, 1.0, 'Glucose vs SkinThickness')
```



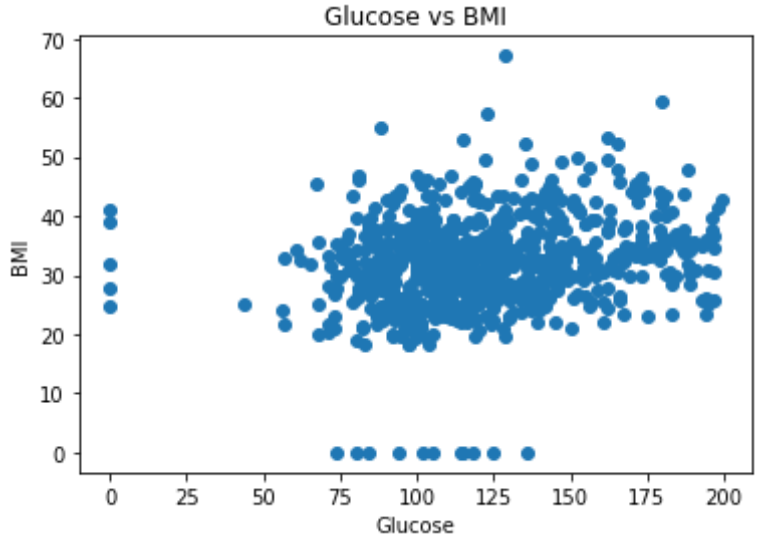
In [10]:
```
plt.scatter(data['Glucose'],data['Insulin'])
plt.xlabel('Glucose')
plt.ylabel('Insulin')
plt.title('Glucose vs Insulin')
```

Out[10]:
```
Text(0.5, 1.0, 'Glucose vs Insulin')
```



In [11]:
```
plt.scatter(data['Glucose'],data['BMI'])
plt.xlabel('Glucose')
plt.ylabel('BMI')
plt.title('Glucose vs BMI')
```

Out[11]:
```
Text(0.5, 1.0, 'Glucose vs BMI')
```



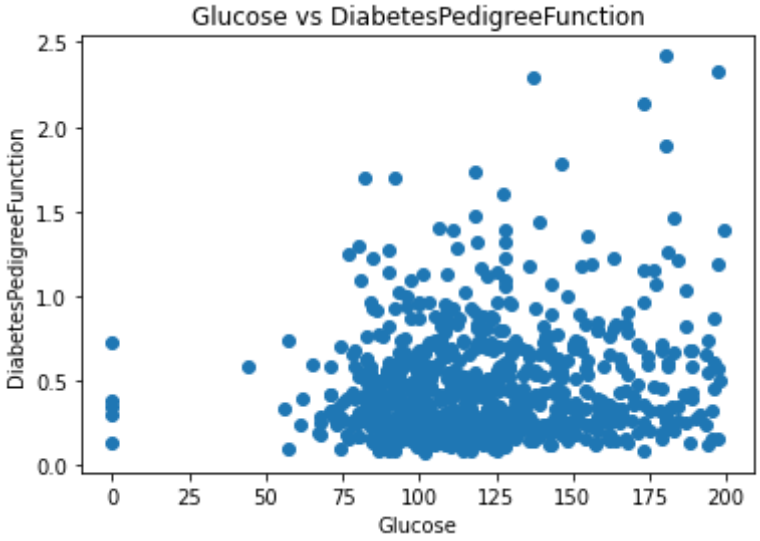
In [12]:
```
data.head(3)
```

Out[12]:

	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

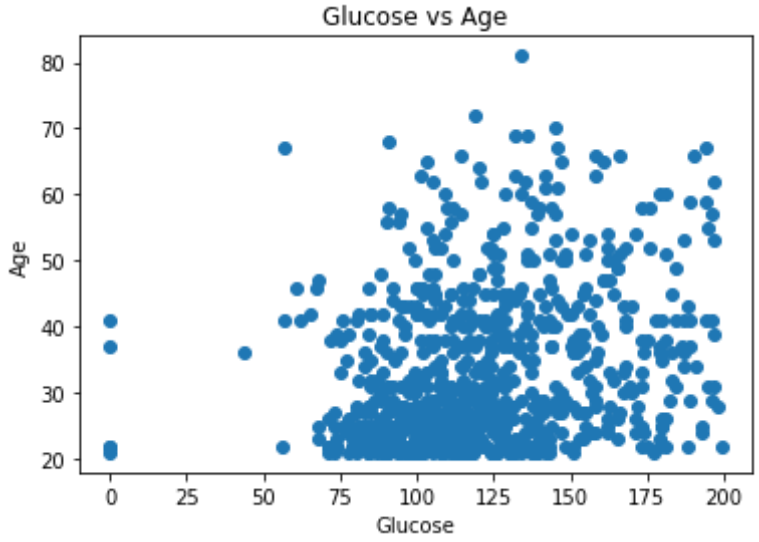
In [13]:
```
plt.scatter(data['Glucose'],data['DiabetesPedigreeFunction'])
plt.xlabel('Glucose')
plt.ylabel('DiabetesPedigreeFunction')
plt.title('Glucose vs DiabetesPedigreeFunction')
```

Out[13]:
```
Text(0.5, 1.0, 'Glucose vs DiabetesPedigreeFunction')
```



In [14]:
```
plt.scatter(data['Glucose'],data['Age'])
plt.xlabel('Glucose')
plt.ylabel('Age')
plt.title('Glucose vs Age')
```

Out[14]:
```
Text(0.5, 1.0, 'Glucose vs Age')
```



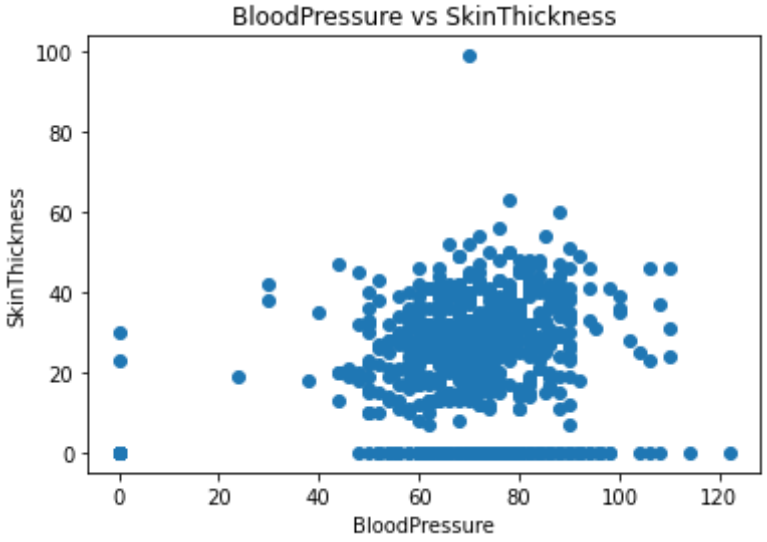
In [15]:
```
data.head(3)
```

Out[15]:

	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

In [16]:
```
plt.scatter(data['BloodPressure'],data['SkinThickness'])
plt.xlabel('BloodPressure')
plt.ylabel('SkinThickness')
plt.title('BloodPressure vs SkinThickness')
```

Out[16]:
```
Text(0.5, 1.0, 'BloodPressure vs SkinThickness')
```



In [17]:
```
X=data[['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age']].va
```

In [19]:
```
y=data[['Outcome']]
```

In [20]:
```
from sklearn.model_selection import train_test_split
```

In [32]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.05, random_state=0)
```

In [33]:
```
X_test
```

Out[33]:
```
array([[1.000e+00, 1.990e+02, 7.600e+01, 4.300e+01, 0.000e+00, 4.290e+01,
        1.394e+00, 2.200e+01],
       [2.000e+00, 1.070e+02, 7.400e+01, 3.000e+01, 1.000e+02, 3.360e+01,
        4.040e-01, 2.300e+01],
       [4.000e+00, 7.600e+01, 6.200e+01, 0.000e+00, 0.000e+00, 3.400e+01,
        3.910e-01, 2.500e+01],
       [5.000e+00, 1.660e+02, 7.200e+01, 1.900e+01, 1.750e+02, 2.580e+01,
        5.870e-01, 5.100e+01],
       [0.000e+00, 1.110e+02, 6.500e+01, 0.000e+00, 0.000e+00, 2.460e+01,
        6.600e-01, 3.100e+01],
       [1.000e+00, 8.100e+01, 7.200e+01, 1.800e+01, 4.000e+01, 2.660e+01,
        2.830e-01, 2.400e+01],
```

```
[9.000e+00, 1.520e+02, 7.800e+01, 3.400e+01, 1.710e+02, 3.420e+01,
 8.930e-01, 3.300e+01],
[3.000e+00, 1.760e+02, 8.600e+01, 2.700e+01, 1.560e+02, 3.330e+01,
 1.154e+00, 5.200e+01],
[2.000e+00, 1.270e+02, 5.800e+01, 2.400e+01, 2.750e+02, 2.770e+01,
 1.600e+00, 2.500e+01],
[6.000e+00, 1.030e+02, 7.200e+01, 3.200e+01, 1.900e+02, 3.770e+01,
 3.240e-01, 5.500e+01],
[3.000e+00, 1.580e+02, 7.600e+01, 3.600e+01, 2.450e+02, 3.160e+01,
 8.510e-01, 2.800e+01],
[5.000e+00, 1.870e+02, 7.600e+01, 2.700e+01, 2.070e+02, 4.360e+01,
 1.034e+00, 5.300e+01],
[5.000e+00, 1.140e+02, 7.400e+01, 0.000e+00, 0.000e+00, 2.490e+01,
 7.440e-01, 5.700e+01],
[2.000e+00, 8.200e+01, 5.200e+01, 2.200e+01, 1.150e+02, 2.850e+01,
 1.699e+00, 2.500e+01],
[8.000e+00, 6.500e+01, 7.200e+01, 2.300e+01, 0.000e+00, 3.200e+01,
 6.000e-01, 4.200e+01],
[0.000e+00, 1.080e+02, 6.800e+01, 2.000e+01, 0.000e+00, 2.730e+01,
 7.870e-01, 3.200e+01],
[8.000e+00, 1.790e+02, 7.200e+01, 4.200e+01, 1.300e+02, 3.270e+01,
 7.190e-01, 3.600e+01],
[1.000e+00, 7.300e+01, 5.000e+01, 1.000e+01, 0.000e+00, 2.300e+01,
 2.480e-01, 2.100e+01],
[4.000e+00, 1.320e+02, 8.600e+01, 3.100e+01, 0.000e+00, 2.800e+01,
 4.190e-01, 6.300e+01],
[5.000e+00, 9.900e+01, 5.400e+01, 2.800e+01, 8.300e+01, 3.400e+01,
 4.990e-01, 3.000e+01],
[4.000e+00, 1.440e+02, 8.200e+01, 3.200e+01, 0.000e+00, 3.850e+01,
 5.540e-01, 3.700e+01],
[2.000e+00, 1.180e+02, 8.000e+01, 0.000e+00, 0.000e+00, 4.290e+01,
 6.930e-01, 2.100e+01],
[2.000e+00, 8.700e+01, 0.000e+00, 2.300e+01, 0.000e+00, 2.890e+01,
 7.730e-01, 2.500e+01],
[3.000e+00, 6.100e+01, 8.200e+01, 2.800e+01, 0.000e+00, 3.440e+01,
 2.430e-01, 4.600e+01],
[1.000e+00, 9.700e+01, 6.600e+01, 1.500e+01, 1.400e+02, 2.320e+01,
 4.870e-01, 2.200e+01],
[1.000e+00, 1.240e+02, 6.000e+01, 3.200e+01, 0.000e+00, 3.580e+01,
 5.140e-01, 2.100e+01],
[0.000e+00, 1.040e+02, 7.600e+01, 0.000e+00, 0.000e+00, 1.840e+01,
 5.820e-01, 2.700e+01],
[1.300e+01, 1.530e+02, 8.800e+01, 3.700e+01, 1.400e+02, 4.060e+01,
 1.174e+00, 3.900e+01],
[2.000e+00, 1.120e+02, 6.600e+01, 2.200e+01, 0.000e+00, 2.500e+01,
```

```
        3.070e-01, 2.400e+01],
       [5.000e+00, 1.100e+02, 6.800e+01, 0.000e+00, 0.000e+00, 2.600e+01,
        2.920e-01, 3.000e+01],
       [7.000e+00, 1.360e+02, 7.400e+01, 2.600e+01, 1.350e+02, 2.600e+01,
        6.470e-01, 5.100e+01],
       [0.000e+00, 1.020e+02, 6.400e+01, 4.600e+01, 7.800e+01, 4.060e+01,
        4.960e-01, 2.100e+01],
       [0.000e+00, 1.020e+02, 8.600e+01, 1.700e+01, 1.050e+02, 2.930e+01,
        6.950e-01, 2.700e+01],
       [2.000e+00, 1.220e+02, 5.200e+01, 4.300e+01, 1.580e+02, 3.620e+01,
        8.160e-01, 2.800e+01],
       [1.000e+00, 1.190e+02, 5.400e+01, 1.300e+01, 5.000e+01, 2.230e+01,
        2.050e-01, 2.400e+01],
       [5.000e+00, 1.680e+02, 6.400e+01, 0.000e+00, 0.000e+00, 3.290e+01,
        1.350e-01, 4.100e+01],
       [1.100e+01, 1.360e+02, 8.400e+01, 3.500e+01, 1.300e+02, 2.830e+01,
        2.600e-01, 4.200e+01],
       [2.000e+00, 1.170e+02, 9.000e+01, 1.900e+01, 7.100e+01, 2.520e+01,
        3.130e-01, 2.100e+01],
       [1.000e+00, 1.190e+02, 4.400e+01, 4.700e+01, 6.300e+01, 3.550e+01,
        2.800e-01, 2.500e+01]])
```

In [34]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [35]:
```
knn=KNeighborsClassifier()
knn.fit(X_train,y_train)
```

```
<ipython-input-35-fe56fab2aae3>:2: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  knn.fit(X_train,y_train)
```

Out[35]:
```
KNeighborsClassifier()
```

In [36]:
```
knn.score(X_test,y_test)*100
```

Out[36]:
```
82.05128205128204
```

In [37]:
```
X_test
```

Out[37]:
```
array([[1.000e+00, 1.990e+02, 7.600e+01, 4.300e+01, 0.000e+00, 4.290e+01,
        1.394e+00, 2.200e+01],
       [2.000e+00, 1.070e+02, 7.400e+01, 3.000e+01, 1.000e+02, 3.360e+01,
        4.040e-01, 2.300e+01],
```

```
[4.000e+00, 7.600e+01, 6.200e+01, 0.000e+00, 0.000e+00, 3.400e+01,
 3.910e-01, 2.500e+01],
[5.000e+00, 1.660e+02, 7.200e+01, 1.900e+01, 1.750e+02, 2.580e+01,
 5.870e-01, 5.100e+01],
[0.000e+00, 1.110e+02, 6.500e+01, 0.000e+00, 0.000e+00, 2.460e+01,
 6.600e-01, 3.100e+01],
[1.000e+00, 8.100e+01, 7.200e+01, 1.800e+01, 4.000e+01, 2.660e+01,
 2.830e-01, 2.400e+01],
[9.000e+00, 1.520e+02, 7.800e+01, 3.400e+01, 1.710e+02, 3.420e+01,
 8.930e-01, 3.300e+01],
[3.000e+00, 1.760e+02, 8.600e+01, 2.700e+01, 1.560e+02, 3.330e+01,
 1.154e+00, 5.200e+01],
[2.000e+00, 1.270e+02, 5.800e+01, 2.400e+01, 2.750e+02, 2.770e+01,
 1.600e+00, 2.500e+01],
[6.000e+00, 1.030e+02, 7.200e+01, 3.200e+01, 1.900e+02, 3.770e+01,
 3.240e-01, 5.500e+01],
[3.000e+00, 1.580e+02, 7.600e+01, 3.600e+01, 2.450e+02, 3.160e+01,
 8.510e-01, 2.800e+01],
[5.000e+00, 1.870e+02, 7.600e+01, 2.700e+01, 2.070e+02, 4.360e+01,
 1.034e+00, 5.300e+01],
[5.000e+00, 1.140e+02, 7.400e+01, 0.000e+00, 0.000e+00, 2.490e+01,
 7.440e-01, 5.700e+01],
[2.000e+00, 8.200e+01, 5.200e+01, 2.200e+01, 1.150e+02, 2.850e+01,
 1.699e+00, 2.500e+01],
[8.000e+00, 6.500e+01, 7.200e+01, 2.300e+01, 0.000e+00, 3.200e+01,
 6.000e-01, 4.200e+01],
[0.000e+00, 1.080e+02, 6.800e+01, 2.000e+01, 0.000e+00, 2.730e+01,
 7.870e-01, 3.200e+01],
[8.000e+00, 1.790e+02, 7.200e+01, 4.200e+01, 1.300e+02, 3.270e+01,
 7.190e-01, 3.600e+01],
[1.000e+00, 7.300e+01, 5.000e+01, 1.000e+01, 0.000e+00, 2.300e+01,
 2.480e-01, 2.100e+01],
[4.000e+00, 1.320e+02, 8.600e+01, 3.100e+01, 0.000e+00, 2.800e+01,
 4.190e-01, 6.300e+01],
[5.000e+00, 9.900e+01, 5.400e+01, 2.800e+01, 8.300e+01, 3.400e+01,
 4.990e-01, 3.000e+01],
[4.000e+00, 1.440e+02, 8.200e+01, 3.200e+01, 0.000e+00, 3.850e+01,
 5.540e-01, 3.700e+01],
[2.000e+00, 1.180e+02, 8.000e+01, 0.000e+00, 0.000e+00, 4.290e+01,
 6.930e-01, 2.100e+01],
[2.000e+00, 8.700e+01, 0.000e+00, 2.300e+01, 0.000e+00, 2.890e+01,
 7.730e-01, 2.500e+01],
[3.000e+00, 6.100e+01, 8.200e+01, 2.800e+01, 0.000e+00, 3.440e+01,
 2.430e-01, 4.600e+01],
[1.000e+00, 9.700e+01, 6.600e+01, 1.500e+01, 1.400e+02, 2.320e+01,
```

```
 4.870e-01, 2.200e+01],
[1.000e+00, 1.240e+02, 6.000e+01, 3.200e+01, 0.000e+00, 3.580e+01,
 5.140e-01, 2.100e+01],
[0.000e+00, 1.040e+02, 7.600e+01, 0.000e+00, 0.000e+00, 1.840e+01,
 5.820e-01, 2.700e+01],
[1.300e+01, 1.530e+02, 8.800e+01, 3.700e+01, 1.400e+02, 4.060e+01,
 1.174e+00, 3.900e+01],
[2.000e+00, 1.120e+02, 6.600e+01, 2.200e+01, 0.000e+00, 2.500e+01,
 3.070e-01, 2.400e+01],
[5.000e+00, 1.100e+02, 6.800e+01, 0.000e+00, 0.000e+00, 2.600e+01,
 2.920e-01, 3.000e+01],
[7.000e+00, 1.360e+02, 7.400e+01, 2.600e+01, 1.350e+02, 2.600e+01,
 6.470e-01, 5.100e+01],
[0.000e+00, 1.020e+02, 6.400e+01, 4.600e+01, 7.800e+01, 4.060e+01,
 4.960e-01, 2.100e+01],
[0.000e+00, 1.020e+02, 8.600e+01, 1.700e+01, 1.050e+02, 2.930e+01,
 6.950e-01, 2.700e+01],
[2.000e+00, 1.220e+02, 5.200e+01, 4.300e+01, 1.580e+02, 3.620e+01,
 8.160e-01, 2.800e+01],
[1.000e+00, 1.190e+02, 5.400e+01, 1.300e+01, 5.000e+01, 2.230e+01,
 2.050e-01, 2.400e+01],
[5.000e+00, 1.680e+02, 6.400e+01, 0.000e+00, 0.000e+00, 3.290e+01,
 1.350e-01, 4.100e+01],
[1.100e+01, 1.360e+02, 8.400e+01, 3.500e+01, 1.300e+02, 2.830e+01,
 2.600e-01, 4.200e+01],
[2.000e+00, 1.170e+02, 9.000e+01, 1.900e+01, 7.100e+01, 2.520e+01,
 3.130e-01, 2.100e+01],
[1.000e+00, 1.190e+02, 4.400e+01, 4.700e+01, 6.300e+01, 3.550e+01,
 2.800e-01, 2.500e+01]])
```

In [38]:
```
y_test
```

Out[38]:

	Outcome
661	1
122	0
113	0
14	1
529	0
103	0

	Outcome
338	1
588	1
395	0
204	0
31	1
546	1
278	0
593	0
737	0
202	0
175	1
55	0
479	0
365	0
417	1
577	1
172	0
352	0
27	0
605	0
239	0
744	0
79	0
496	0
285	0

	Outcome
422	0
640	0
374	0
385	0
404	1
648	1
500	0
575	0

In [39]:
```
predct
```

Out[39]:
```
{6: 50,
 1: 31,
 8: 32,
 0: 21,
 5: 33,
 3: 30,
 10: 26,
 2: 29,
 4: 53,
 7: 54,
 9: 34,
 11: 57,
 13: 59,
 15: 51,
 17: 27,
 12: 41,
 14: 43}
```

In [40]:
```
data. head()
```

Out[40]:

	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

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
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

In [41]:
```
y_pred=knn.predict([[2,103,71,32,156,45.6,1.568,55]])
```

In [45]:
```
if y_pred==1:
    print("daabaties")
else:
        print("no diabaties")
```

```
daabaties
```

In [ ]:
```

```