In [1]:
```python
#importing the dependencies

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

In [2]:
```python
#data collection from csv file
#loadingg the diabetes dataset to panda Dataframe
dataset=pd.read_csv("D:\diabetes prediction dataset.csv")
```

In [3]:
```python
dataset
```

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
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [4]:
```python
#printing  the first 5 rows of the dataset
dataset.head()
```

Out[4]:

	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

In [5]:
```python
#number odf rows and coloumns in this dataset
dataset.shape
```

Out[5]: (768, 9)

In [6]:
```python
#getting the statistical measures of the data
dataset.describe()
```

Out[6]:

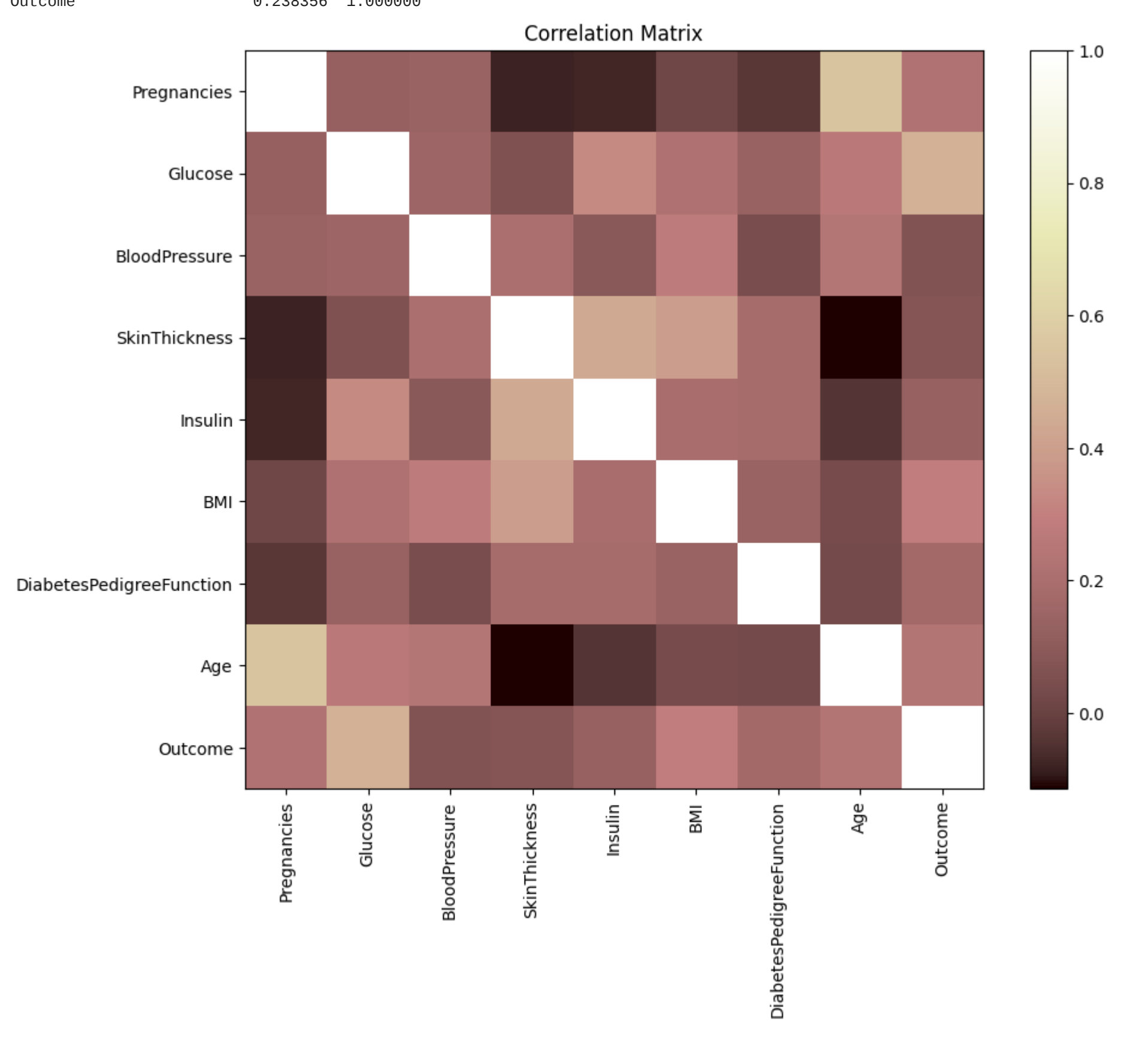
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	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

In [7]:
```python
dataset['Outcome'].value_counts()

#0-->Non-Diabetic
#1-->Diabetic
```

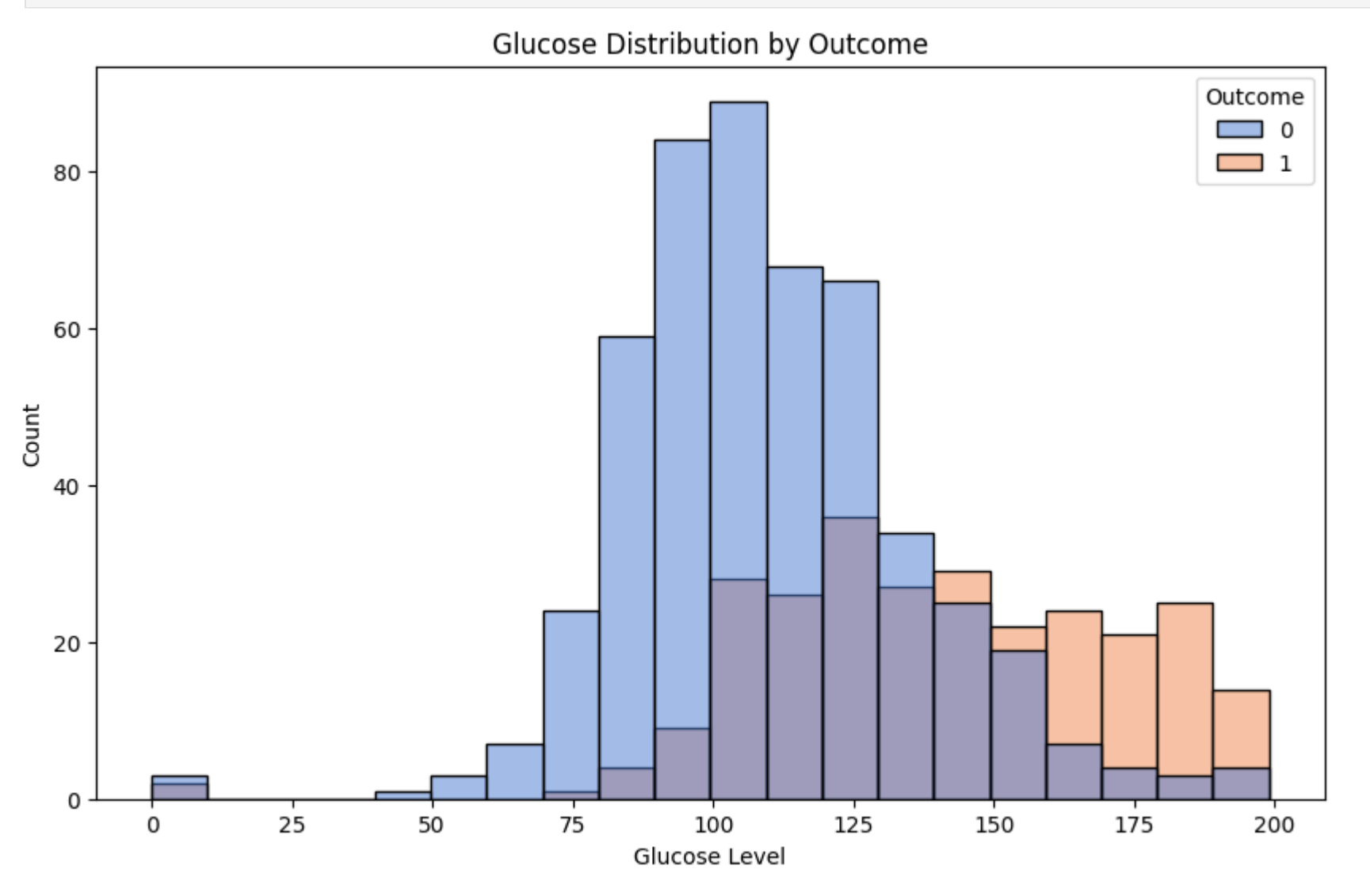
Out[7]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [8]:
```python
# Adding the correlation matrix to the code

# Get the correlation matrix
correlation_matrix = dataset.corr()

# Print the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)

# Plotting the correlation matrix as a heatmap
plt.figure(figsize=(10, 8))
plt.title('Correlation Matrix')
plt.imshow(correlation_matrix, cmap='pink')
plt.colorbar()
plt.xticks(range(len(dataset.columns)), dataset.columns,rotation=90)
plt.yticks(range(len(dataset.columns)), dataset.columns)
plt.show()
```

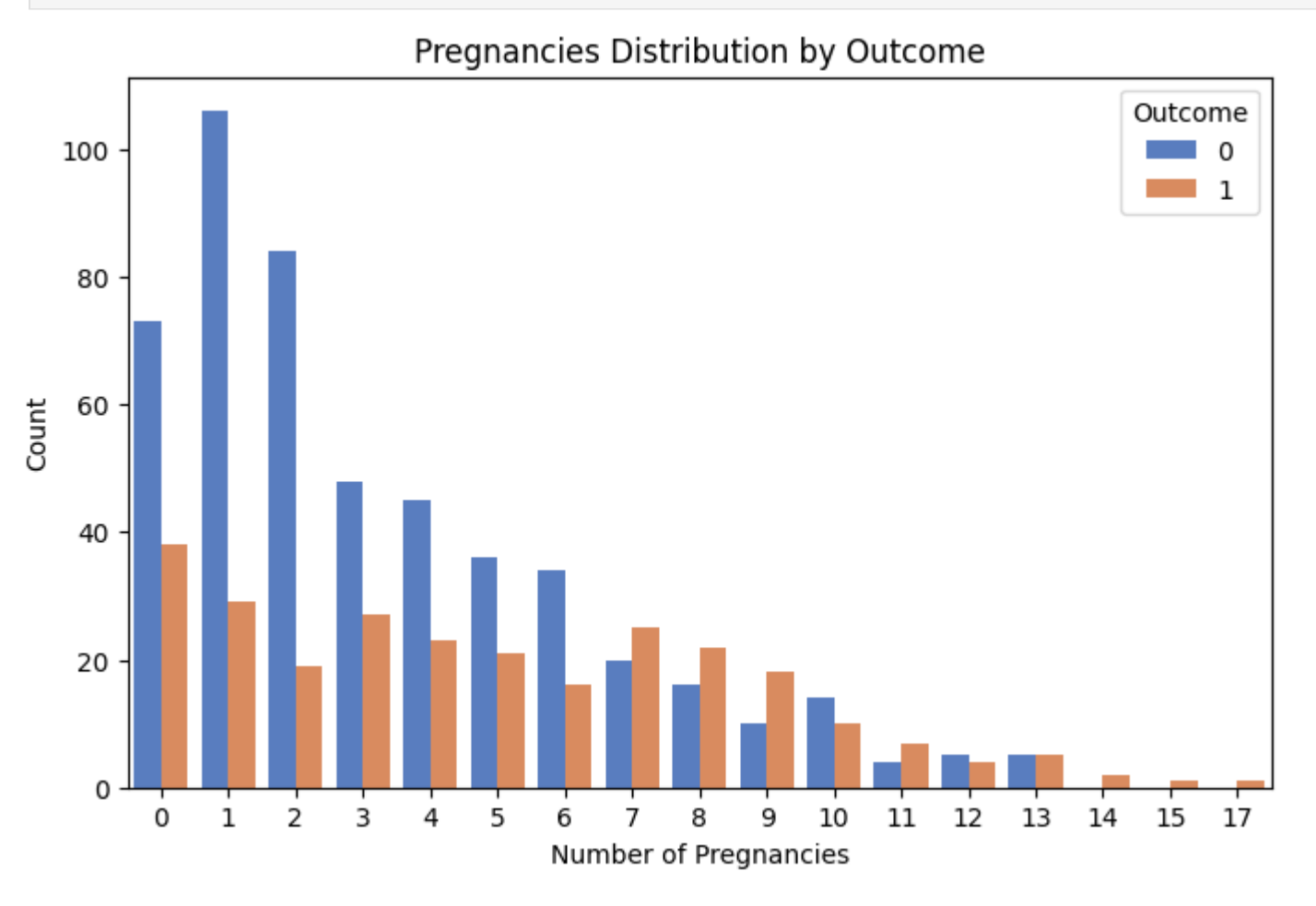
```
Correlation Matrix:
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       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
```



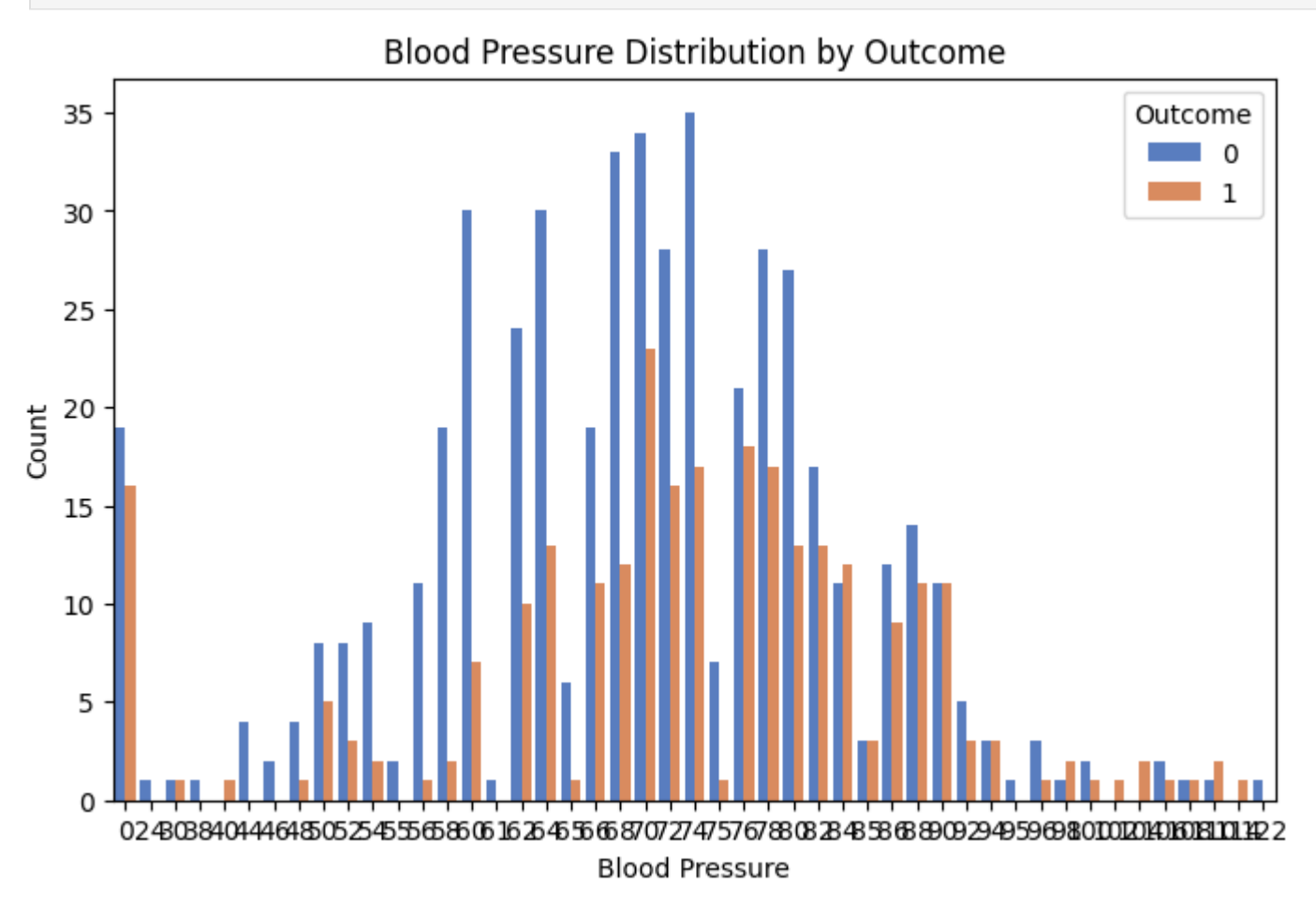
In [9]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(x='Glucose', hue='Outcome', data=dataset, bins=20, palette='muted')
plt.title('Glucose Distribution by Outcome')
plt.xlabel('Glucose Level')
plt.ylabel('Count')
plt.show()
```



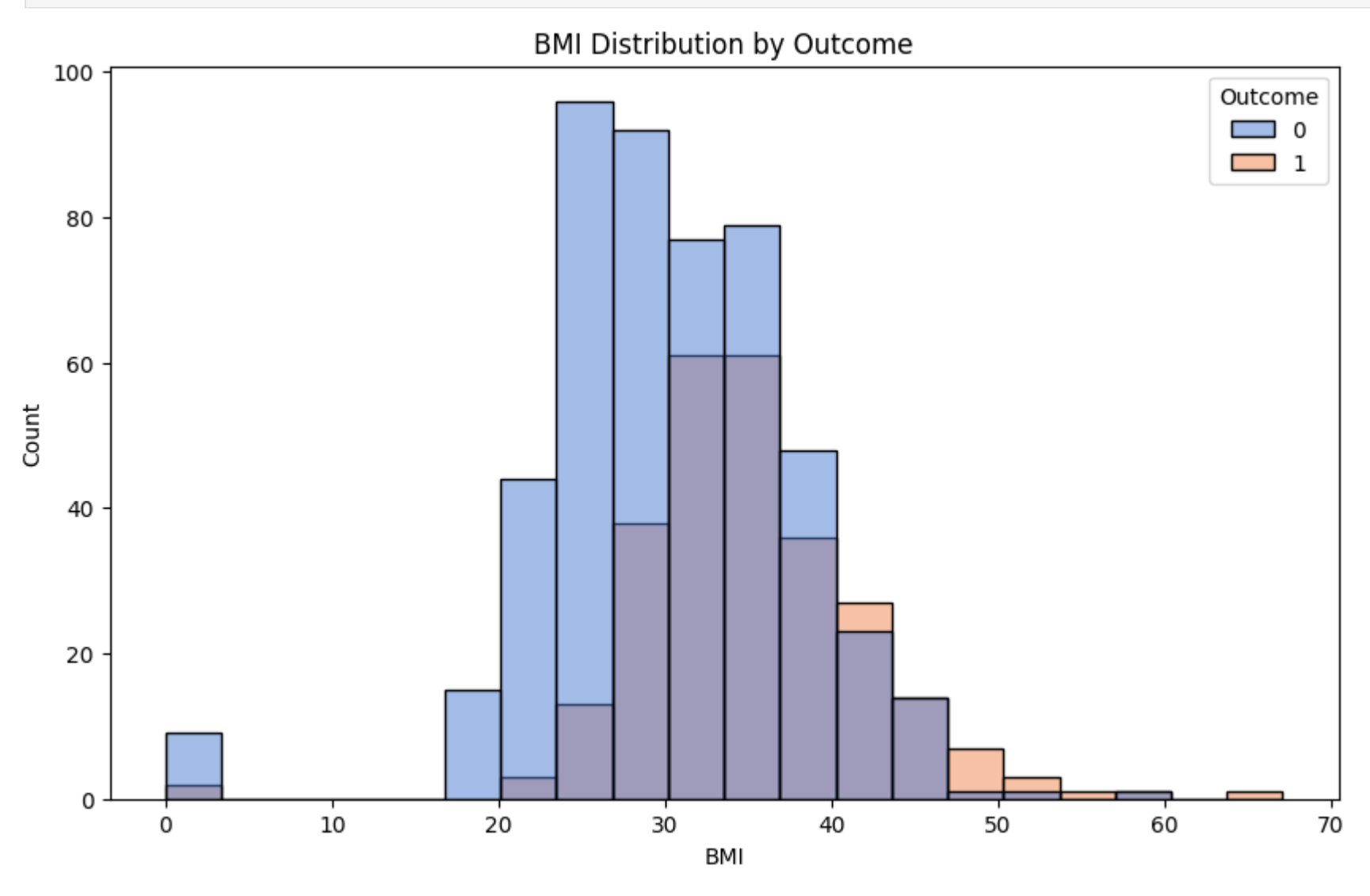
In [10]:
```python
plt.figure(figsize=(8, 5))
sns.countplot(x='Pregnancies', hue='Outcome', data=dataset, palette='muted')
plt.title('Pregnancies Distribution by Outcome')
plt.xlabel('Number of Pregnancies')
plt.ylabel('Count')
plt.show()
```



In [11]:
```python
plt.figure(figsize=(8, 5))
sns.countplot(x='BloodPressure', hue='Outcome', data=dataset, palette='muted')
plt.title('Blood Pressure Distribution by Outcome')
plt.xlabel('Blood Pressure')
plt.ylabel('Count')
plt.show()
```



In [12]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(x='BMI', hue='Outcome', data=dataset, bins=20, palette='muted')
plt.title('BMI Distribution by Outcome')
plt.xlabel('BMI')
plt.ylabel('Count')
plt.show()
```

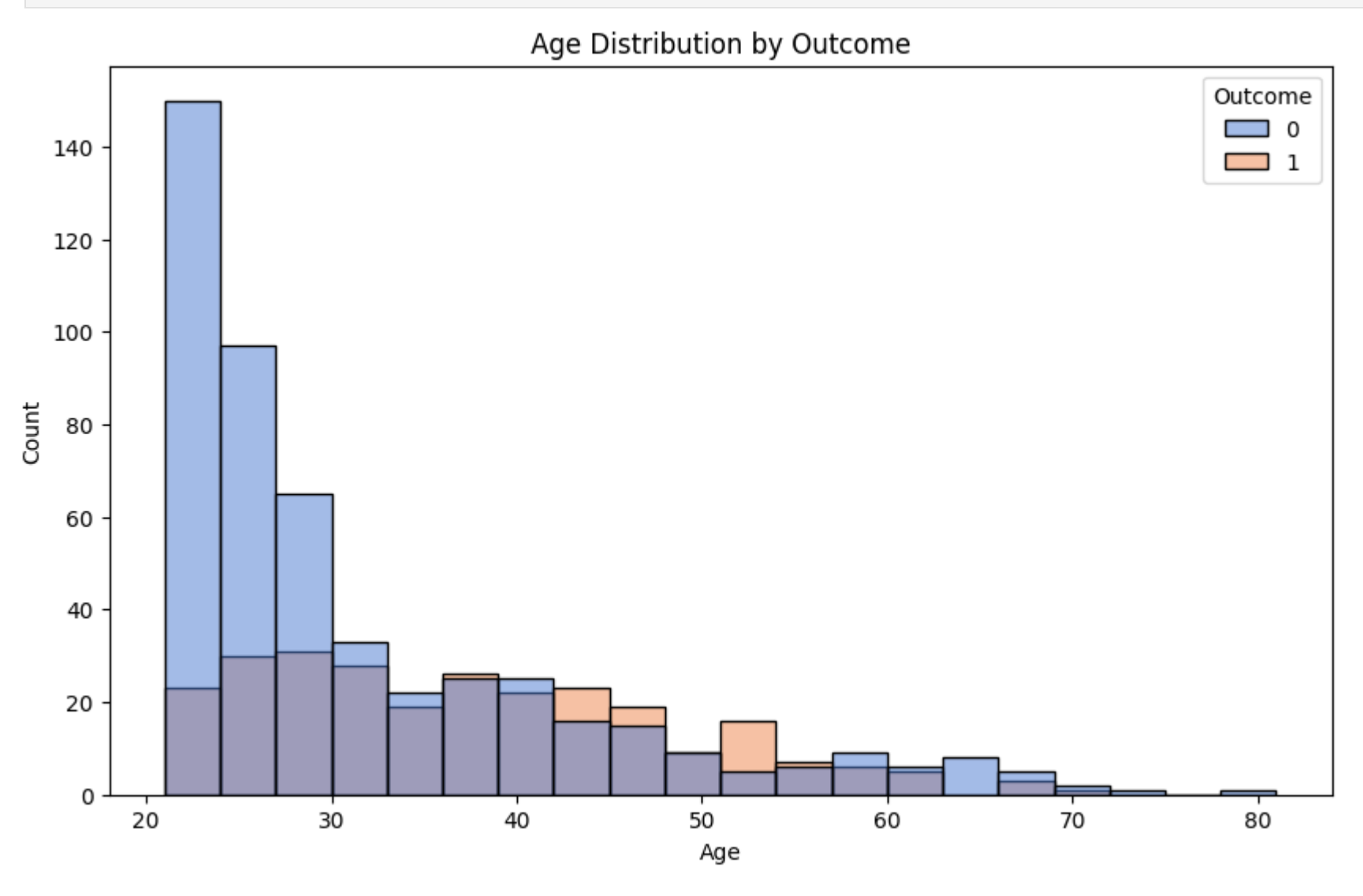


In [13]:
```python
#calculating mean for both diabetic and non diabetic
dataset.groupby('Outcome').mean()
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
Outcome								
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

In [14]:
```python
plt.figure(figsize=(10, 6))
sns.histplot(x='Age', hue='Outcome', data=dataset, bins=20, palette='muted')
plt.title('Age Distribution by Outcome')
plt.xlabel('Age')
plt.ylabel('Count')
plt.show()
```



In [15]:
```python
# Seperating the data and labels

X=dataset.drop(columns='Outcome',axis=1)

# axis =0 is for rows
# axis =1 is for columns

Y=dataset['Outcome']
```

In [16]:
```python
print(X)
```

```
     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
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age
0                       0.627   50
1                       0.351   31
2                       0.672   32
3                       0.167   21
4                       2.288   33
..                        ...  ...
763                     0.171   63
764                     0.340   27
765                     0.245   30
766                     0.349   47
767                     0.315   23

[768 rows x 8 columns]
```

In [17]:
```python
print(Y)
```

```
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 [18]:
```python
#Data Standardization

scaler=StandardScaler()
scaler.fit(X)

standarized_data=scaler.transform(X)

print(standarized_data)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
```

In [19]:
```python
X=standarized_data
Y=dataset['Outcome']

print(X)
print(Y)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.20401277  0.46849198
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.68442195 -0.36506078
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.10325546  0.60439732
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.73518964 -0.68519336
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.24020459 -0.37110101
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.20212881 -0.47378505
  -0.87137393]]
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 [20]:
```python
# Train Test Split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,stratify=Y,random_state=2)
```

In [21]:
```python
print(X.shape,X_train.shape,X_test.shape)
```

```
(768, 8) (614, 8) (154, 8)
```

In [22]:
```python
classifier=svm.SVC(kernel='linear')

#training the support vector Machine Classifier

classifier.fit(X_train,Y_train)
```

Out[22]:
```
▾ SVC
SVC(kernel='linear')
```

In [23]:
```python
# Accuracy of the training data

X_train_prediction=classifier.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)

print('Accuracy score of the training data:',training_data_accuracy)
```

```
Accuracy score of the training data: 0.7866449511400652
```

In [24]:
```python
# Accuracy of the test data

X_test_prediction=classifier.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)

print('Accuracy score of the test data:',test_data_accuracy)
```

```
Accuracy score of the test data: 0.7727272727272727
```

In [25]:
```python
feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
input_data=(1,189,60,23,846,30.1,0.398,59)

#changing the input_data to numpy array
input_data_as_numpy_array=np.asarray(input_data)

#reshaping the array as we are predicting for one instance
input_data_reshape=input_data_as_numpy_array.reshape(1,-1)

#standardize the input data

std_data=scaler.transform(input_data_reshape)
print(std_data)

prediction=classifier.predict(std_data)
print(prediction)

if (prediction[0]==0):
    print('The person is not diabetic')
else:
    print('The person is diabetic')
```

```
[[-0.84488505  2.13150675 -0.47073225  0.15453319  6.65283938 -0.24020459
  -0.2231152   2.19178518]]
[1]
The person is diabetic
```

```
C:\Users\Anja\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
```

In [ ]: