The data was collected and made available by “National Institute of Diabetes and Digestive and Kidney Diseases” as part of the Pima Indians Diabetes Database. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here belong to the Pima Indian heritage (subgroup of Native Americans), and are females of ages 21 and above. We’ll be using Python and some of its popular data science related packages. First of all, we will import pandas to read our data from a CSV file and manipulate it for further use. We will also use numpy to convert out data into a format suitable to feed our classification model. We’ll use seaborn and matplotlib for visualizations. We will then import Logistic Regression algorithm from sklearn. This algorithm will help us build our classification model. Lastly, we will use joblib available in sklearn to save our model for future use.

we used the data provided to predict the effect each pregrancies, glucose,bloodpressure, skinThickness,Insulin.... and age to create model and train it to predict outcome

Import the Libraries needed for creating the model:

In [1]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import seaborn as sns
import statsmodels.api as sm
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

Import the dataset to the notebook:

In [2]:
```
data  = pd.read_csv('diabetes.csv',header=0)

data.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

Display the columns Independent Vs. Dependent variables:

In [3]:
```
data.columns
```

Out[3]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [ ]:
```

```

Dependent Variables: Y

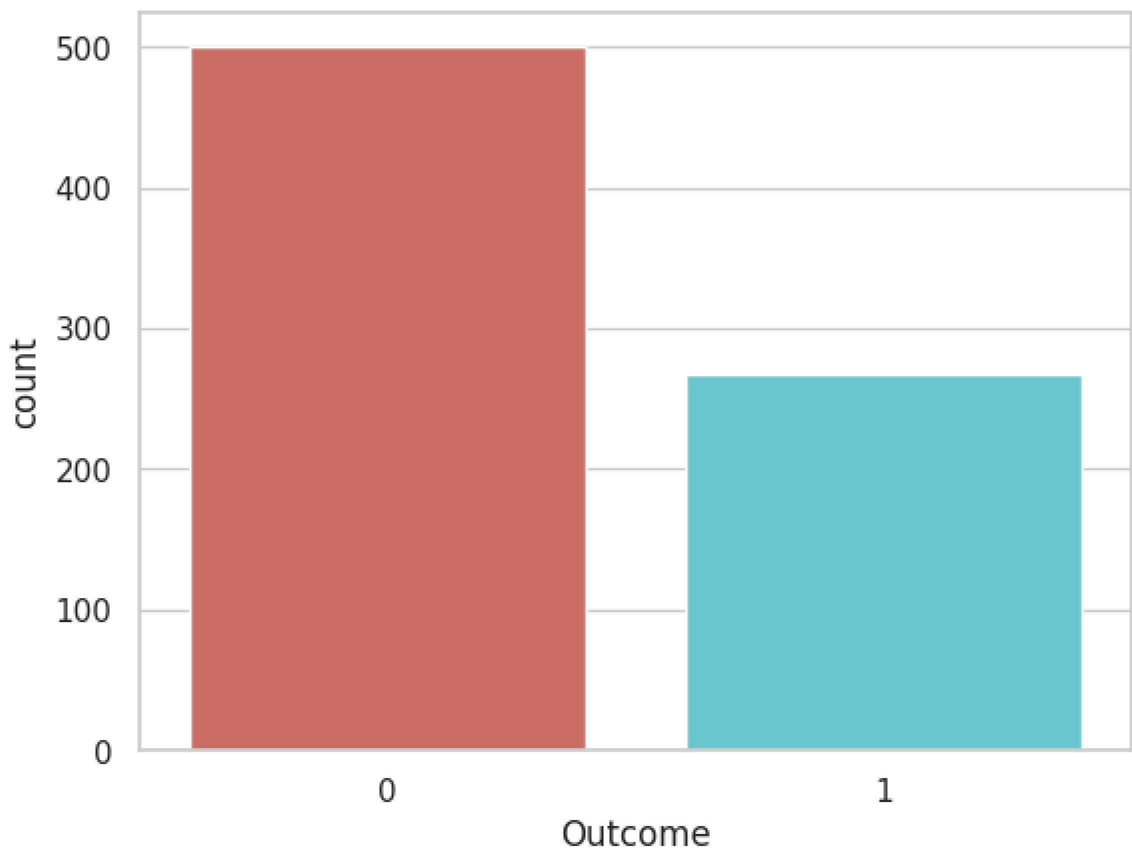
Independent Variables: 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age',

data exploration

In [4]:
```
data['Outcome'].value_counts()
```

Out[4]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [5]:
```
sns.countplot(x='Outcome',data=data,palette='hls')
plt.show()
```



In [6]:
```
count_no_diabetes = len(data[data['Outcome']==0])
count_diabetes = len(data[data['Outcome']==1])
pct_of_no_diabetes = count_no_diabetes/(count_no_diabetes+count_diabetes)
print("percentage of no diabetes is",pct_of_no_diabetes*100)
pct_of_sub = count_diabetes/(count_no_diabetes+count_diabetes)
print("percentage of diabetes", pct_of_no_diabetes*100)
```

```
percentage of no diabetes is 65.10416666666666
percentage of diabetes 65.10416666666666
```

let's do some more exploration.

In [7]:
```
data.groupby(["Outcome"]).mean()
```

Out[7]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	

In [8]:
```
data.groupby(["Pregnancies"]).mean()
```

Out[8]:

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	123.000000	67.153153	22.270270	81.675676	34.290090	
1	112.748148	67.792593	24.437037	98.674074	31.372593	
2	110.796117	63.252427	21.601942	85.844660	30.583495	
3	123.586667	66.586667	20.080000	87.453333	30.425333	
4	125.117647	70.029412	15.882353	69.441176	32.141176	
5	118.859649	76.210526	17.385965	57.298246	33.192982	
6	120.800000	68.420000	17.640000	63.580000	30.290000	
7	136.444444	70.777778	20.288889	84.466667	32.631111	
8	131.736842	75.184211	17.315789	92.815789	31.568421	
9	131.392857	77.892857	20.892857	62.428571	31.707143	
10	120.916667	70.208333	15.708333	34.791667	30.641667	
11	126.454545	74.181818	21.727273	65.454545	38.563636	
12	113.555556	76.333333	27.111111	112.555556	32.344444	
13	125.500000	73.800000	17.300000	27.900000	35.000000	
14	137.500000	70.000000	27.500000	92.000000	35.100000	
15	136.000000	70.000000	32.000000	110.000000	37.100000	
17	163.000000	72.000000	41.000000	114.000000	40.900000	

In [9]:
```
data.groupby(["Insulin"]).mean()
```

Out[9]:

Insulin	Pregnancies	Glucose	BloodPressure	SkinThickness	BMI	DiabetesPedigreeFu
0	4.433155	119.409091	67.473262	11.508021	30.943316	0.
14	0.000000	180.000000	78.000000	63.000000	59.400000	2.
15	2.000000	68.000000	62.000000	13.000000	20.100000	0.
16	2.000000	88.000000	58.000000	26.000000	28.400000	0.
18	2.000000	91.000000	65.000000	30.500000	36.450000	0.
...	...	...	...	...	...	
579	1.000000	172.000000	68.000000	49.000000	42.400000	0.
600	8.000000	124.000000	76.000000	24.000000	28.700000	0.
680	0.000000	165.000000	90.000000	33.000000	52.300000	0.
744	4.000000	197.000000	70.000000	39.000000	36.700000	2.
846	1.000000	189.000000	60.000000	23.000000	30.100000	0.

186 rows × 8 columns

In [10]:
```
data.groupby(["BMI"]).mean()
```

Out[10]:

BMI	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	DiabetesPedigreeFun
0.0	3.909091	104.272727	28.818182	4.181818	8.090909	0.43
18.2	1.000000	92.333333	67.333333	11.333333	27.333333	0.35
18.4	0.000000	104.000000	76.000000	0.000000	0.000000	0.58
19.1	1.000000	80.000000	55.000000	0.000000	0.000000	0.25
19.3	3.000000	99.000000	80.000000	11.000000	64.000000	0.28
...	...	...	...	...	...	
53.2	0.000000	162.000000	76.000000	56.000000	100.000000	0.75
55.0	1.000000	88.000000	30.000000	42.000000	99.000000	0.49
57.3	3.000000	123.000000	100.000000	35.000000	240.000000	0.88
59.4	0.000000	180.000000	78.000000	63.000000	14.000000	2.42
67.1	0.000000	129.000000	110.000000	46.000000	130.000000	0.31

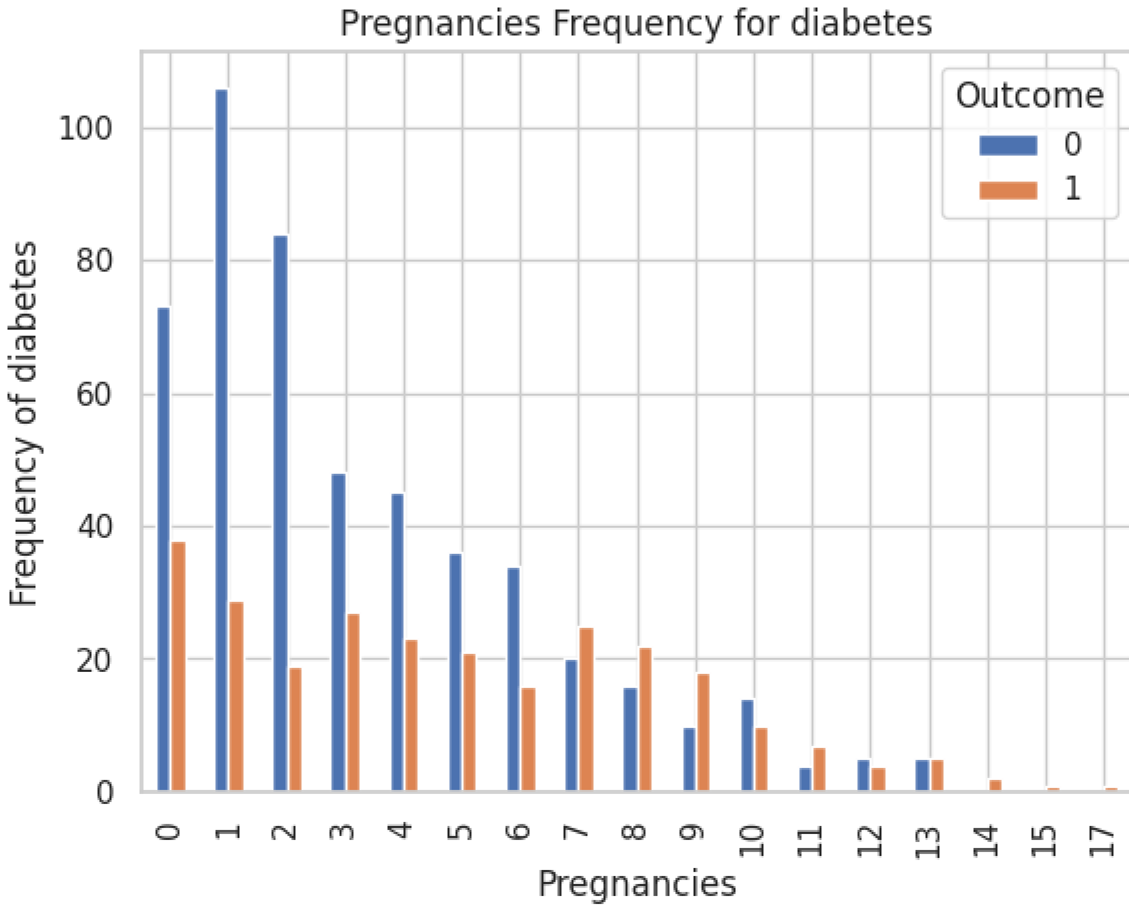
248 rows × 8 columns

data visualizations

In [11]:
```
pd.crosstab(data.Pregnancies,data.Outcome).plot(kind='bar')
plt.title('Pregnancies Frequency for diabetes')
```

```
plt.xlabel('Pregnancies')
plt.ylabel('Frequency of diabetes')
```

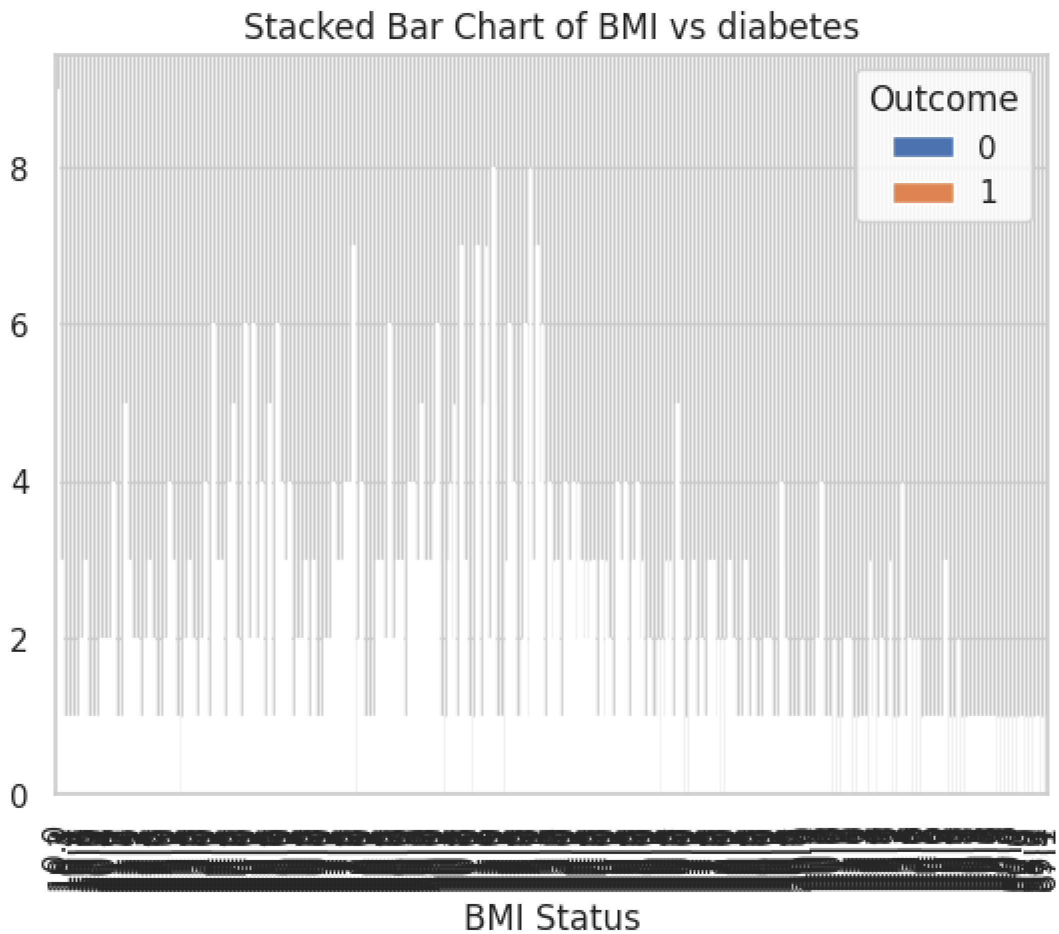
Out[11]: Text(0, 0.5, 'Frequency of diabetes')



The frequency of pregrancy has great deal on the diabetes. Thus, the pregnancy can be a good predictor of the outcome variable.

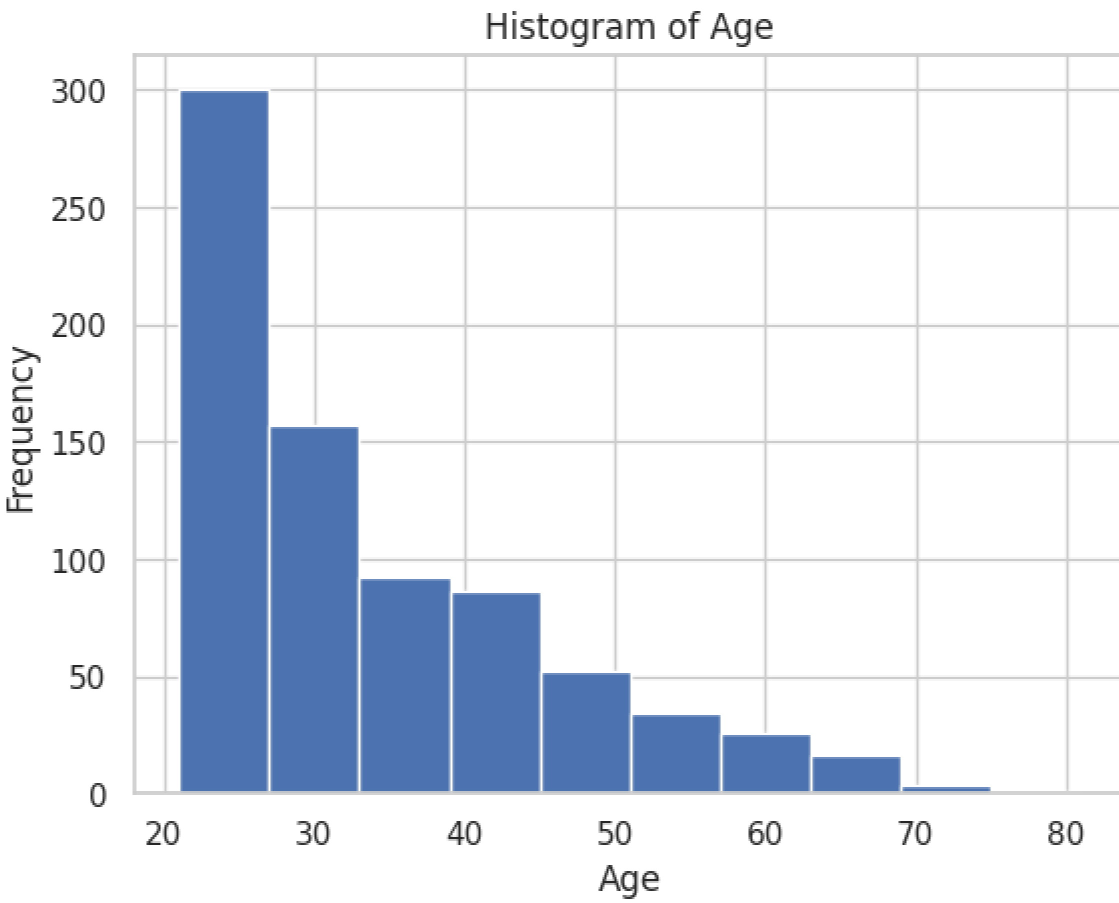
In [12]:
```
pd.crosstab(data.BMI,data.Outcome).plot(kind='bar')
plt.title('Stacked Bar Chart of BMI vs diabetes')
plt.xlabel('BMI Status')
```

Out[12]: Text(0.5, 0, 'BMI Status')



In [13]:
```
data.Age.hist()
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
```

Out[13]: `Text(0, 0.5, 'Frequency')`



the above histogram shows that most of the peoples in the given dataset are aged between 20 and 30

In [14]:
```
data.isna().sum()
```

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

check for the presence of duplicate data

In [15]:
```
data.duplicated().any()
```

Out[15]:
```
False
```

In [16]:
```
data.shape
```

Out[16]:
```
(768, 9)
```

In [17]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Identifying dependent and independent variables

x represents independent variables and

y represents dependent variables

In [18]:
```
x = data.drop(['Outcome'],axis=1)
y = data.Outcome

x
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.
764	2	122	70	27	0	36.8	0.
765	5	121	72	23	112	26.2	0.
766	1	126	60	0	0	30.1	0.
767	1	93	70	31	0	30.4	0.

768 rows × 8 columns

In [19]:
```
y
```

Out[19]:
```
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]:
```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 48.1 KB
```

In [21]:
```
y.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 768 entries, 0 to 767
Series name: Outcome
Non-Null Count  Dtype
--------------  -----
768 non-null    int64
dtypes: int64(1)
memory usage: 6.1 KB
```

In [22]:
```
model = sm.Logit(y,x)
result = model.fit()
print(result.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.608498
         Iterations 5
                             Results: Logit
========================================================================
Model:                 Logit              Pseudo R-squared:   0.059
Dependent Variable:    Outcome            AIC:                950.6528
Date:                  2023-02-10 17:53   BIC:                987.8031
No. Observations:      768                Log-Likelihood:     -467.33
Df Model:              7                  LL-Null:            -496.74
Df Residuals:          760                LLR p-value:        2.5825e-10
Converged:             1.0000             Scale:              1.0000
No. Iterations:        5.0000
------------------------------------------------------------------------
                          Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
------------------------------------------------------------------------
Pregnancies               0.1284   0.0286  4.4843 0.0000  0.0723  0.1845
Glucose                   0.0129   0.0027  4.7568 0.0000  0.0076  0.0183
BloodPressure            -0.0303   0.0047 -6.4806 0.0000 -0.0395 -0.0212
SkinThickness             0.0002   0.0061  0.0323 0.9742 -0.0117  0.0121
Insulin                   0.0007   0.0008  0.9420 0.3462 -0.0008  0.0023
BMI                      -0.0048   0.0107 -0.4494 0.6531 -0.0258  0.0162
DiabetesPedigreeFunction  0.3203   0.2399  1.3351 0.1818 -0.1499  0.7905
Age                      -0.0156   0.0084 -1.8517 0.0641 -0.0322  0.0009
========================================================================
```

In [ ]:

```python

```

In [23]:

```python
x = x.drop(["SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"
```

In [24]:

```python
x
```

Out[24]:

	Pregnancies	Glucose	BloodPressure
0	6	148	72
1	1	85	66
2	8	183	64
3	1	89	66
4	0	137	40
...	...	...	...
763	10	101	76
764	2	122	70
765	5	121	72
766	1	126	60
767	1	93	70

768 rows × 3 columns

In [25]:

```python
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.3,random_s
```

In [26]:
```
dmodel = LogisticRegression()
```

In [27]:
```
dmodel.fit(x_train,y_train)
```

Out[27]:
```
▾ LogisticRegression
LogisticRegression()
```

In [28]:
```
print("Accuracy of our model is: {:.2f}".format(dmodel.score(x_test,y_test)))
```

```
Accuracy of our model is: 0.77
```

In [29]:
```
y_pred = dmodel.predict(x_test)
```

In [30]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       150
           1       0.72      0.54      0.62        81

    accuracy                           0.77       231
   macro avg       0.75      0.71      0.73       231
weighted avg       0.76      0.77      0.76       231
```

Interpretation:

Of the entire test set, 75% of the data are free from diabetes. Of the entire test set, 76% are diabetic

In [ ]:

In [ ]:

In [ ]:

In [ ]: