In [1]:
```
import pandas as pd
import os
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
pwd
```

Out[2]:
```
'C:\\Users\\Vivek'
```

In [3]:
```
from sklearn.externals import joblib
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\__i
nit__.py:15: FutureWarning: sklearn.externals.joblib is deprecated in
0.21 and will be removed in 0.23. Please import this functionality dire
ctly from joblib, which can be installed with: pip install joblib. If t
his warning is raised when loading pickled models, you may need to re-s
erialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
```

In [4]:
```
diabetesDF = pd.read_csv('diabetes.csv')
print(diabetesDF.head())
```

```
   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

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
```

```
3                    0.167   21        0
4                    2.288   33        1
```

In [5]:
```
diabetesDF.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
```

In [6]:
```
dfTrain = diabetesDF[:650]
dfTest = diabetesDF[650:750]
dfCheck = diabetesDF[750:]
```

In [7]:
```
trainLabel = np.asarray(dfTrain['Outcome'])
trainData = np.asarray(dfTrain.drop('Outcome',1))
testLabel = np.asarray(dfTest['Outcome'])
testData = np.asarray(dfTest.drop('Outcome',1))
```

In [8]:
```
means = np.mean(trainData, axis=0)
stds = np.std(trainData, axis=0)
trainData = (trainData - means)/stds
testData = (testData - means)/stds
```

In [9]:
```
diabetesCheck = LogisticRegression()
```

```
diabetesCheck.fit(trainData, trainLabel)
```

Out[9]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbo
se=0,
                   warm_start=False)
```

In [10]:
```
accuracy = diabetesCheck.score(testData, testLabel)
print("accuracy = ", accuracy * 100, "%")
```

```
accuracy =  78.0 %
```

In [12]:
```
joblib.dump([diabetesCheck, means, stds], 'diabeteseModel.pkl')
```

Out[12]:
```
['diabeteseModel.pkl']
```

In [13]:
```
diabetesLoadedModel, means, stds = joblib.load('diabeteseModel.pkl')
accuracyModel = diabetesLoadedModel.score(testData, testLabel)
print("accuracy = ",accuracyModel * 100,"%")
```

```
accuracy =  78.0 %
```

In [14]:
```
print(dfCheck.head())
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
750            4      136             70              0        0  31.2

751            1      121             78             39       74  39.0

752            3      108             62             24        0  26.0

753            0      181             88             44      510  43.3

754            8      154             78             32        0  32.4
```

```
     DiabetesPedigreeFunction  Age  Outcome
750                     1.182   22        1
751                     0.261   28        0
752                     0.223   25        0
753                     0.222   26        1
754                     0.443   45        1
```

In [15]:
```
sampleData = dfCheck[:1]
sampleDataFeatures = np.asarray(sampleData.drop('Outcome',1))
sampleDataFeatures = (sampleDataFeatures - means)/stds
predictionProbability = diabetesLoadedModel.predict_proba(sampleDataFea
tures)
prediction = diabetesLoadedModel.predict(sampleDataFeatures)
print('Probability:', predictionProbability)
print('prediction:', prediction)
```

```
Probability: [[0.44077634 0.55922366]]
prediction: [1]
```

In [ ]:
```

```