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

In [2]:
```
df1 = pd.read_csv(r'C:\Users\LaptopCheckout\Downloads\diabetes.csv')
df1.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [3]:
```
df1.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```
df1.isnull().sum()
```

Out[4]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [5]:
```
df1.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [6]:
```
dframe1=df1.dropna(subset=["BMI"], inplace=True)
```

Data cleaning

In [7]:
```
df1.isnull().sum()
```

Out[7]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [8]:
```
print("Total : ", df1[df1.BloodPressure == 0].shape[0])
Total : 35
print(df1[df1.BloodPressure == 0].groupby('Outcome')['Age'].count())
```

```
Total :  35
Outcome
0    19
1    16
Name: Age, dtype: int64
```

Insulin

In [9]:
```
print("Total : ", df1[df1.Insulin == 0].shape[0])
print(df1[df1.Insulin == 0].groupby('Outcome')['Age'].count())
```

```
Total :  374
Outcome
0    236
1    138
Name: Age, dtype: int64
```

In [10]:
```
print("Total : ", df1[df1.Glucose == 0].shape[0])
print(df1[df1.Glucose == 0].groupby('Outcome')['Age'].count())
```

```
Total :  5
Outcome
0    3
1    2
Name: Age, dtype: int64
```

In [11]:
```
print("Total : ", df1[df1.SkinThickness == 0].shape[0])
print(df1[df1.SkinThickness == 0].groupby('Outcome')['Age'].count())
```

```
Total :  227
Outcome
0    139
1     88
Name: Age, dtype: int64
```

Handling invalid data

In [12]:
```
df1_m = df1[(df1.BloodPressure != 0) & (df1.BMI != 0) & (df1.Glucose != 0)]
print(df1.shape)
```

```
(768, 9)
```

Feature engineering

In [13]:
```
feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness','Insu
X = df1_m[feature_names]
y = df1_m.Outcome
```

In [14]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 60.0 KB
```

Model selection

In [15]:
```
from sklearn.linear_model import LogisticRegression
models = []
models.append(('LR', LogisticRegression()))
```

In [16]:
```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.33, random_st
names = []
scores = []
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    scores.append(accuracy_score(y_test, y_pred))
    names.append(name)
tr_split = pd.DataFrame({'Name': names, 'Score': scores})
print(tr_split)
```

```
  Name  Score
0   LR    1.0

C:\Users\LaptopCheckout\Anaconda\lib\site-packages\sklearn\linear_model\_logist
ic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

In [ ]:

In [ ]: