In [67]:
```
# import required libraries
import numpy as np
import pandas as pd
import sklearn
```

In [53]:
```
# import the dataset
df = pd.read_csv('./diabetes-dataset.csv')
```

Examine the dataset

In [54]:
```
df.head()
```

Out[54]:

	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 [55]:
```
df.describe()
```

Out[55]:

	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 [56]:
```
df.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 [57]:
```
df.shape
```

Out[57]:
```
(768, 9)
```

In [58]:
```
df.describe().T
```

Out[58]:

	count	mean	std	min	25%	50%	75%	
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	1
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	19
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	12
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	9
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	84
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	6
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	8
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	

Detecting the null values

In [59]:
```
df.isnull().sum()
```

Out[59]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

important note...

Detecting NaNs In this dataset, as I dived into it, except in 'Outcome' & 'Pregnancies' features, all 0 values are representing NaN. Let's handle it and see how many missing values are there in the dataset.

In [60]:
```
# replace 0s with 'np.nan's
new_df = df.drop(['Outcome', 'Pregnancies'], axis=1)
df[[
    'Glucose',
    'BloodPressure',
    'SkinThickness',
    'Insulin',
    'BMI',
    'DiabetesPedigreeFunction',
    'Age']] = new_df.replace(0, np.nan)

df.head()
```

Out[60]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148.0	72.0	35.0	NaN	33.6	0.627	
1	1	85.0	66.0	29.0	NaN	26.6	0.351	
2	8	183.0	64.0	NaN	NaN	23.3	0.672	
3	1	89.0	66.0	23.0	94.0	28.1	0.167	
4	0	137.0	40.0	35.0	168.0	43.1	2.288	

In [61]:
```
df.isnull().sum()
```

Out[61]:
```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [62]:
```
# replace NaN values with each feature's mean
df.fillna(df.mean(), inplace=True)
df.isnull().sum()
```

Out[62]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

now our dataset is clean

In [142]:
```
df.head()
```

Out[142]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148.0	72.0	35.00000	155.548223	33.6	0.62
1	1	85.0	66.0	29.00000	155.548223	26.6	0.35
2	8	183.0	64.0	29.15342	155.548223	23.3	0.67
3	1	89.0	66.0	23.00000	94.000000	28.1	0.16
4	0	137.0	40.0	35.00000	168.000000	43.1	2.28

Logistic regresion & Naive Bayes

In [75]:
```
Y = df['Outcome']
X = df.drop(['Outcome'], axis=1)
```

In [111]:
```
# Logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=df.shape[0])
model.fit(X, Y)

# Naive Bayes
# from sklearn.naive_bayes import GaussianNB
# model = GaussianNB()
# model.fit(X, Y) # X is train attributes, Y is train labels
```

Out[111]:
```
LogisticRegression(max_iter=768)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Now Let's check wether or not someone has diabetes

In [171]:
```
# 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI' 'l
new_person1 = [1,85,66,29,155.548223,26.6,0.351,31]
new_person2 = [10,95,80,74,200.548223,40.6,0.751,38]
class_predict = model.predict([new_person1, new_person2])

for i in class_predict:
    print(f'{i} = Person-{i+1}: ', end="")
    if i == 1:
        print(f'Diabetic')
    else:
        print('Non-diabetic')

print('\n*****\n\n\n')
```

```
0 = Person-1: Non-diabetic
1 = Person-2: Diabetic

*****



C:\Python310\lib\site-packages\sklearn\base.py:450: UserWarning: X does not hav
e valid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

Thank you :)