importing the dependecies

In [70]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from matplotlib.pyplot import figure
import seaborn as sns
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
import plotly.express as px
```

Data Collection And Analysis

PIMA Diabeyes Dataset

In [71]:
```
data=pd.read_csv("diabetes_data.csv")
```

In [72]:
```
#printing first 5 rows of dataset
data.head()
```

Out[72]:

	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 [73]:
```
# information of dataset
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
```

In [6]:
```
data.shape
```

Out[6]: (768, 9)

In [74]:
```
data.describe()
```

Out[74]:

	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	

Check Missing Values

In [12]:
```
1 data.isnull().sum()
```

Out[12]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [14]:
```
1 # Handle Missing Values
2 data.isnull().sum()*100/len(data)
```

Out[14]:
```
Pregnancies                 0.0
Glucose                     0.0
BloodPressure               0.0
SkinThickness               0.0
Insulin                     0.0
BMI                         0.0
DiabetesPedigreeFunction    0.0
Age                         0.0
Outcome                     0.0
dtype: float64
```

In [48]:
```
1 data['Outcome'].value_counts()
```

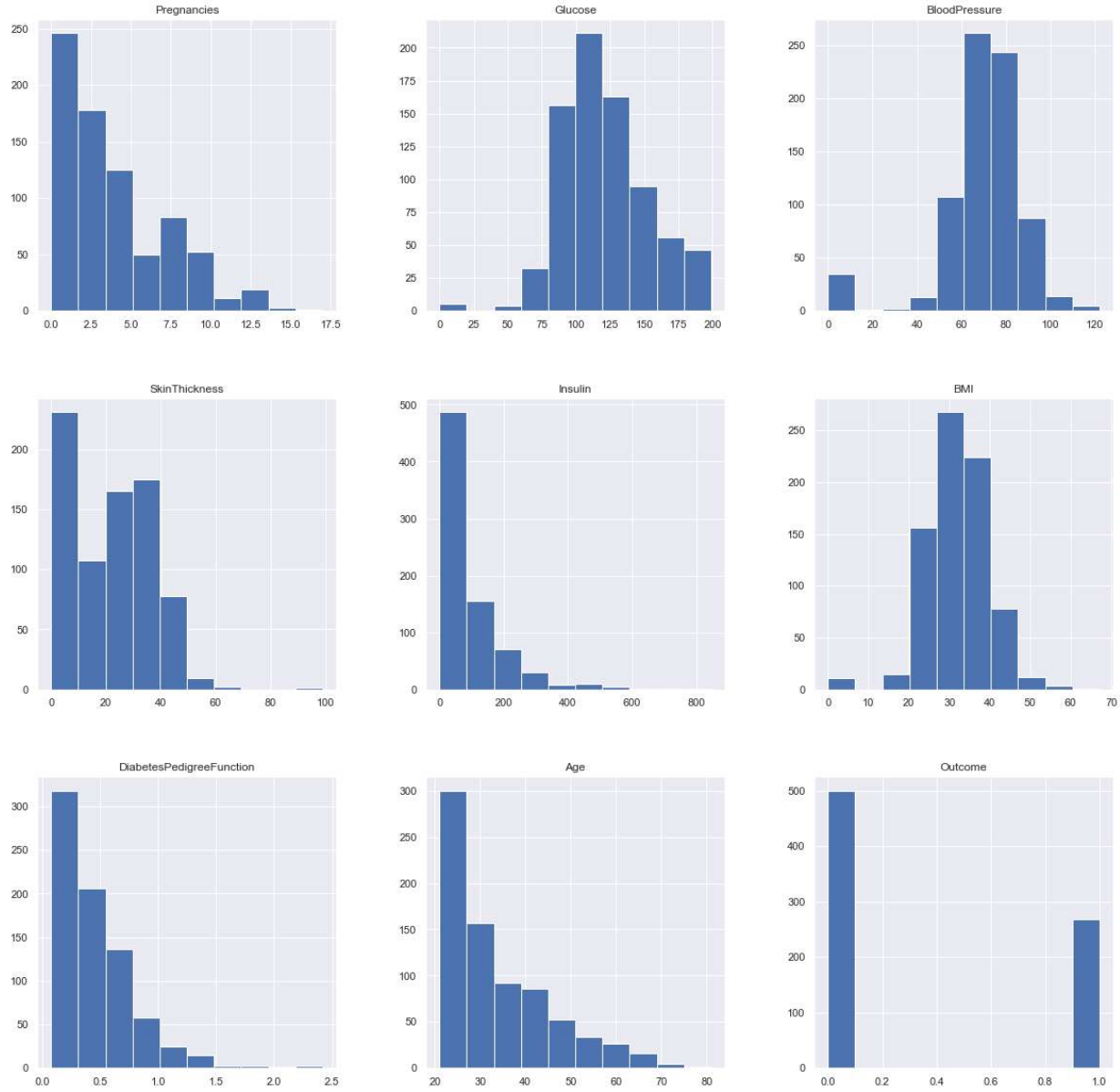
Out[48]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

0--> Non Diabetic

1--> Diabetic

Histogram

In [20]:
```
#histogram
p = data.hist(figsize = (20,20))
```



PairPlot

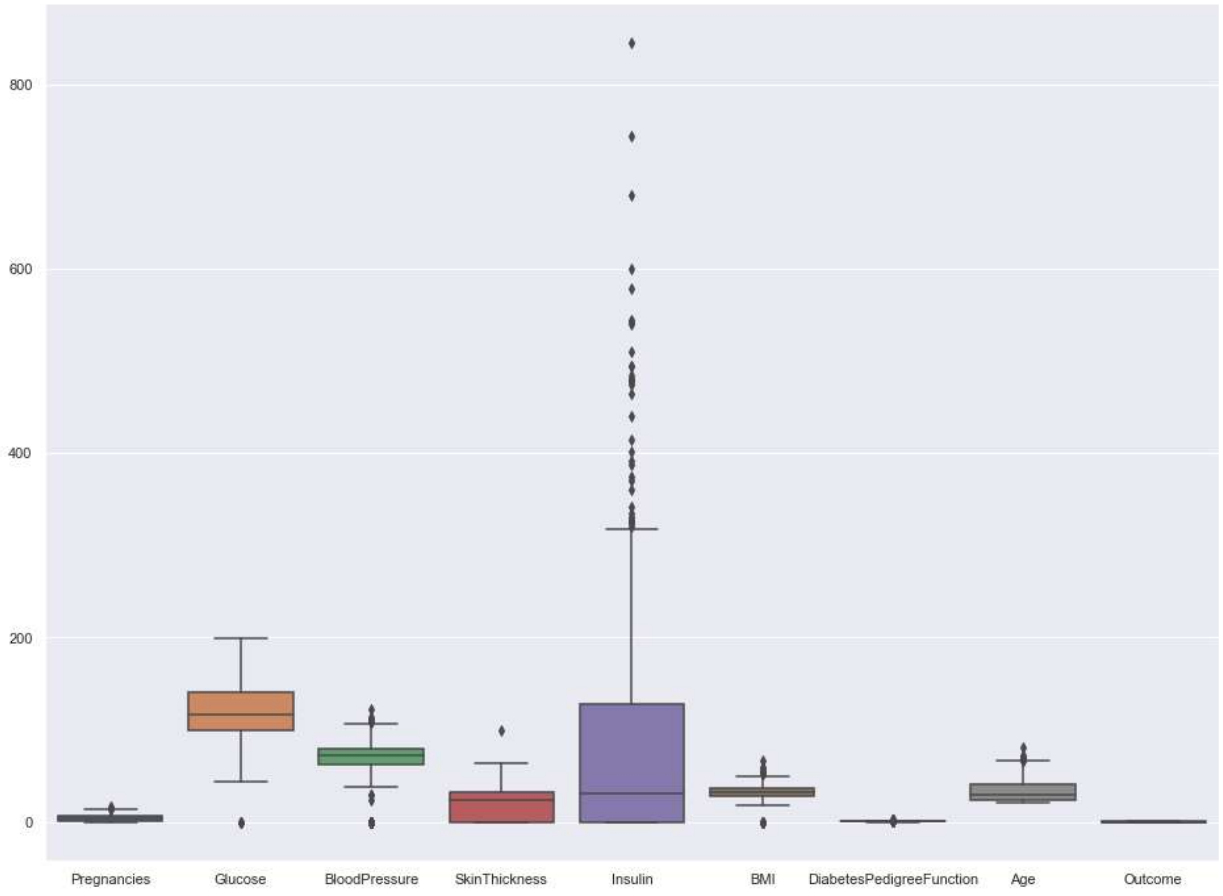
In [21]:
```
#pairplot
sns.pairplot(data)
```

Out[21]: <seaborn.axisgrid.PairGrid at 0x17c523b69a0>

Boxplot

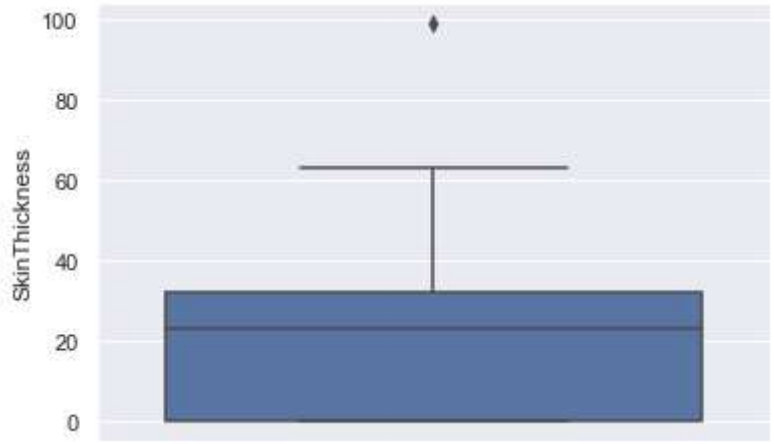
In [45]:
```
plt.figure(figsize=(16,12))
sns.boxplot(data=data)
```

Out[45]: <AxesSubplot:>



In [46]:
```
1 sns.boxplot(y = 'SkinThickness', data = data)
```

Out[46]: <AxesSubplot:ylabel='SkinThickness'>



In [62]:
```
1 from sklearn.preprocessing import StandardScaler
2 scaler = StandardScaler()
```

In [64]:
```
1 scaler.fit(x)
```

Out[64]: StandardScaler()

In [27]:
```
1 x = data.iloc[:, :-1]
2 y = data.iloc[:, -1]
```

In [28]:
```
1 x.head()
```

Out[28]:

	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 [29]:
```
1 y.head()
```

Out[29]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

Train_Test_Split

In [76]:
```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, r
```

In [77]:
```
1 print_score(X_train, X_test, y_train, y_test)
```

```
SVM:

Train score : 0.758957654723127
Test score : 0.7922077922077922
------------------------------------------------------------------------------
---------------------
```

In [80]:
```
1 X_train.shape
```

Out[80]: (614, 8)

In [82]:
```
1 y_test.shape
```

Out[82]: (154,)

In [83]:
```
1 X_train.shape
```

Out[83]: (614, 8)

In [84]:
```
1 y_train.shape
```

Out[84]: (614,)

Model Evaluation

In [34]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [86]:
```
1 svm=SVC(kernel='linear')
2 svm
```

Out[86]: SVC(kernel='linear')

In [36]:
```
1 svm.fit(X_train,y_train)
```

Out[36]:
```
SVC(kernel='linear')
```

In [85]:
```
1 y_svm=svm.predict(X_test)
2 y_svm
```

Out[85]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
      dtype=int64)
```

Confusion_Matrix

In [39]:
```
1 svm_cm_test=confusion_matrix(y_test,y_svm)
2 svm_cm_test
```

Out[39]:
```
array([[97, 10],
       [18, 29]], dtype=int64)
```

Accuracy

In [42]:
```
1 from sklearn.metrics import accuracy_score
2 svm_acc_test=accuracy_score(y_test,y_svm)
```

In [47]:
```
1
2 print(svm_acc_test)
```

```
0.8181818181818182
```

Making a Predictive System

In [75]:
```
input_data=(4,110,92,0,0,37.6,0.191,30)

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

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

#standardize the input_data
std_data=scaler.transform(input_data_reshaped)
print(std_data)

prediction=svm.predict(std_data)
print(prediction)
```

```
[[ 0.04601433 -0.34096773  1.18359575 -1.28821221 -0.69289057  0.71168975
  -0.84827977 -0.27575966]]
[0]

C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but StandardScaler was fitted with feature n
ames
  warnings.warn(
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

In [68]:
```
if(prediction[0]==0):
    print('non-diabetic')
else:
    print('diabetic')
```

```
non-diabetic
```

Making pickle file ¶

In [69]:
```
import pickle
diabetes_pickle_file='svm.pkl'
pickle.dump(svm,open('diabetes_pickle_file','wb'))
pic=pickle.load(open('diabetes_pickle_file','rb'))
pic.predict([[4,110,92,0,0,37.6,0.191,30]])
```

```
C:\Users\MSCIT\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

Out[69]:
```
array([0], dtype=int64)
```

In [ ]:
```

```