In [1]:

```
#Importing Necessary Libraries.
import pandas as pd
import matplotlib.pyplot as plt
import pandas as np
```

In [2]:

```
#Reading the Data.
data=pd.read_csv("diabetes.csv")
```

In [3]:

```
data
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
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
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

768 rows × 9 columns

EXPLORING THE DATA

In [4]:

```
#Checking Null Values for further steps.
data.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]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

```
In [6]: data.describe()
```

Out[6]:

	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	

From the above dataset the min value and when we look at the data we can find zeros which basically means not available and it is not shown as a null value so here we have to fix it.

```
So here I have fixed by taking the mean and replacing it
```

```
In [8]: columns=['Glucose' ,'BloodPressure' ,'SkinThickness', 'Insulin' ,'BMI']
```

```
In [9]: for i in columns:
            data[i].replace(0,data[i].mean(),inplace=True)
```

In [10]:
```
data.describe()
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.681605	72.254807	26.606479	118.660163	32.450805	
std	3.369578	30.436016	12.115932	9.631241	93.080358	6.875374	
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	
50%	3.000000	117.000000	72.000000	23.000000	79.799479	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 [25]:
```
data
```

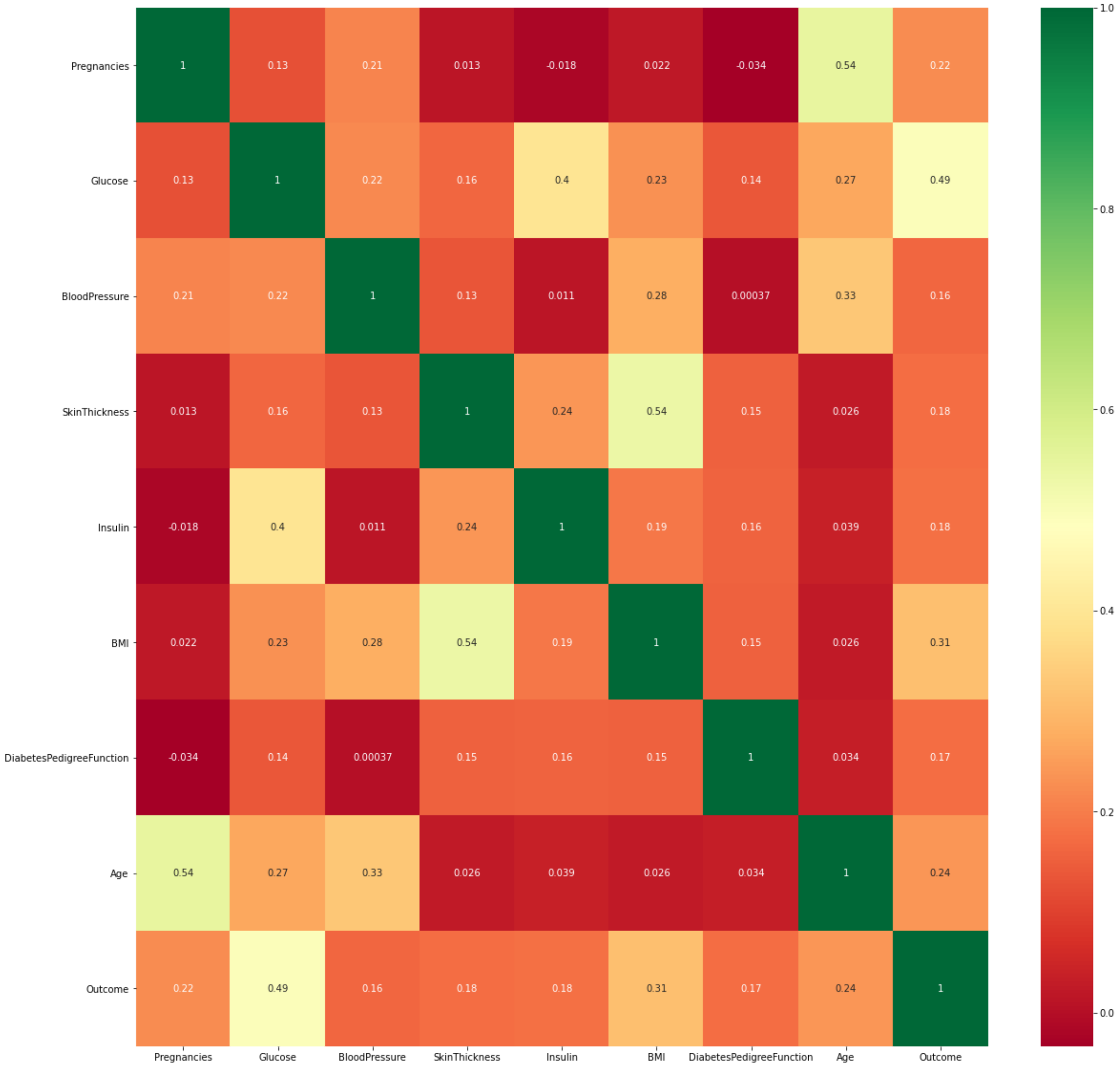
Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148.0	72.0	35.000000	79.799479	33.6	0
1	1	85.0	66.0	29.000000	79.799479	26.6	0
2	8	183.0	64.0	20.536458	79.799479	23.3	0
3	1	89.0	66.0	23.000000	94.000000	28.1	0
4	0	137.0	40.0	35.000000	168.000000	43.1	2
...	...	...	...	...	...	...	
763	10	101.0	76.0	48.000000	180.000000	32.9	0
764	2	122.0	70.0	27.000000	79.799479	36.8	0
765	5	121.0	72.0	23.000000	112.000000	26.2	0
766	1	126.0	60.0	20.536458	79.799479	30.1	0
767	1	93.0	70.0	31.000000	79.799479	30.4	0

768 rows × 9 columns

In [22]:

```
import seaborn as sns
corrmat=data.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(20,20))
#plotting the heatmap
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

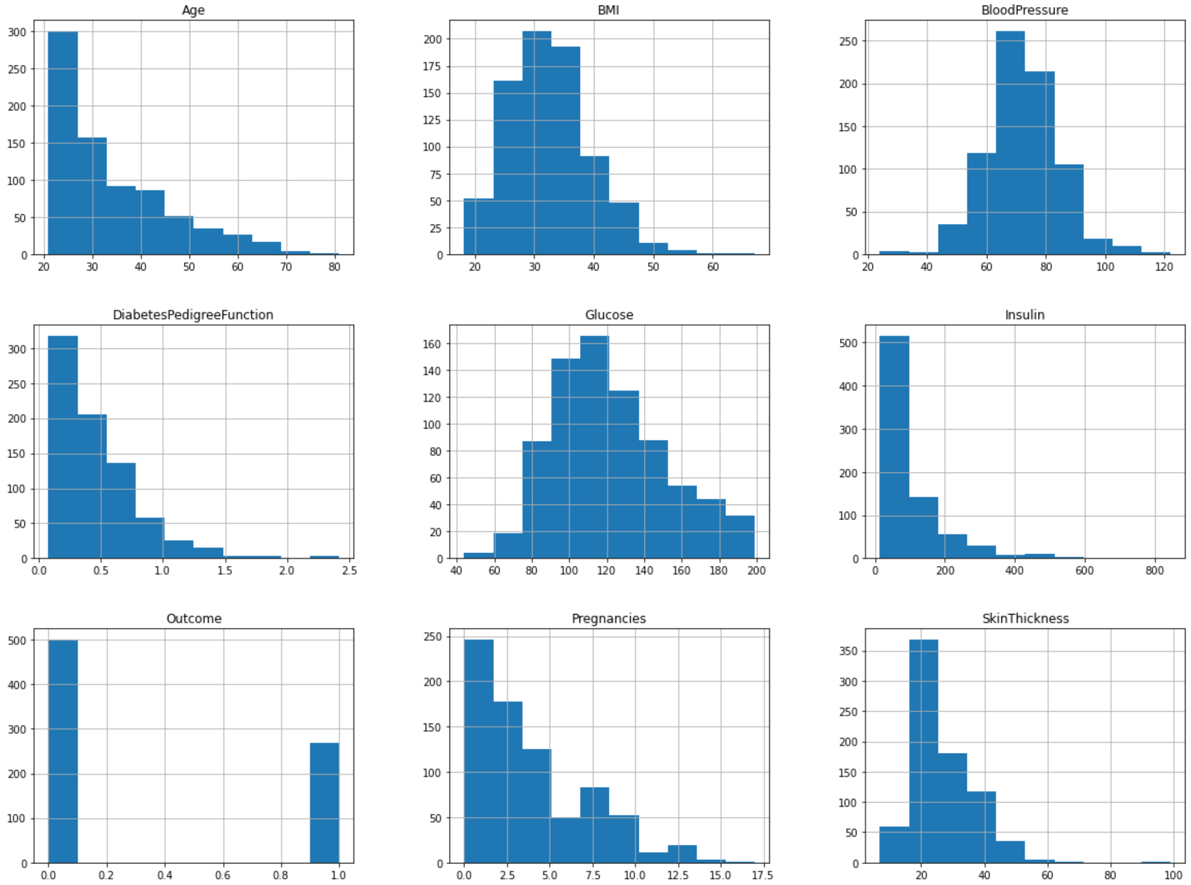


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.21	0.013	-0.018	0.022	-0.034	0.54	0.22
Glucose	0.13	1	0.22	0.16	0.4	0.23	0.14	0.27	0.49
BloodPressure	0.21	0.22	1	0.13	0.011	0.28	0.00037	0.33	0.16
SkinThickness	0.013	0.16	0.13	1	0.24	0.54	0.15	0.026	0.18
Insulin	-0.018	0.4	0.011	0.24	1	0.19	0.16	0.039	0.18
BMI	0.022	0.23	0.28	0.54	0.19	1	0.15	0.026	0.31
DiabetesPedigreeFunction	-0.034	0.14	0.00037	0.15	0.16	0.15	1	0.034	0.17
Age	0.54	0.27	0.33	0.026	0.039	0.026	0.034	1	0.24
Outcome	0.22	0.49	0.16	0.18	0.18	0.31	0.17	0.24	1

In [23]:
```
data.hist(figsize=(20,15))
```

```
C:\Users\fathi\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.p
y:307: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two m
inor releases later. Use ax.get_subplotspec().rowspan.start instead.
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
C:\Users\fathi\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.p
y:307: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two m
inor releases later. Use ax.get_subplotspec().colspan.start instead.
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
C:\Users\fathi\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.p
y:313: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two m
inor releases later. Use ax.get_subplotspec().rowspan.start instead.
  if not layout[ax.rowNum + 1, ax.colNum]:
C:\Users\fathi\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\tools.p
y:313: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two m
inor releases later. Use ax.get_subplotspec().colspan.start instead.
  if not layout[ax.rowNum + 1, ax.colNum]:
```

Out[23]:
```
array([[<AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'BMI'}>,
        <AxesSubplot:title={'center':'BloodPressure'}>],
       [<AxesSubplot:title={'center':'DiabetesPedigreeFunction'}>,
        <AxesSubplot:title={'center':'Glucose'}>,
        <AxesSubplot:title={'center':'Insulin'}>],
       [<AxesSubplot:title={'center':'Outcome'}>,
        <AxesSubplot:title={'center':'Pregnancies'}>,
        <AxesSubplot:title={'center':'SkinThickness'}>]], dtype=object)
```



SPLITTING THE DATA

In [11]:

```
x= data[data.columns[:-1]]
y= data['Outcome']
#we will predict the outcome of the diabetes.
```

In [12]:
```
x
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148.0	72.0	35.000000	79.799479	33.6	0
1	1	85.0	66.0	29.000000	79.799479	26.6	0
2	8	183.0	64.0	20.536458	79.799479	23.3	0
3	1	89.0	66.0	23.000000	94.000000	28.1	0
4	0	137.0	40.0	35.000000	168.000000	43.1	2
...	...	...	...	...	...	...	
763	10	101.0	76.0	48.000000	180.000000	32.9	0
764	2	122.0	70.0	27.000000	79.799479	36.8	0
765	5	121.0	72.0	23.000000	112.000000	26.2	0
766	1	126.0	60.0	20.536458	79.799479	30.1	0
767	1	93.0	70.0	31.000000	79.799479	30.4	0

768 rows × 8 columns

In [13]:
```
y
```

Out[13]:
```
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 [15]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_s
```

In [54]:
```
x_train.shape
```

Out[54]:
```
(614, 8)
```

In [16]:
```
from sklearn.preprocessing import StandardScaler
```

In [17]:
```
sc = StandardScaler()
```

In [18]:
```
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [19]:
```
from sklearn.linear_model import LogisticRegression
```

In [20]:
```
#Creating the model.
logmodel = LogisticRegression()
logmodel.fit(x_train, y_train)
prediction1 = logmodel.predict(x_test)
```

In [27]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [28]:
```
accuracy_score(y_test, prediction1)
```

Out[28]:
```
0.8116883116883117
```

In [29]:
```
confusion_matrix(y_test, prediction1)
```

Out[29]:
```
array([[97, 10],
       [19, 28]], dtype=int64)
```

In [79]:
```
print('Confusion Matrix:\n', confusion_matrix(y_test, prediction1))
print('\n')
print('Classification Report:\n', classification_report(y_test, prediction1))
```

```
Confusion Matrix:
 [[97 10]
 [19 28]]


Classification Report:
               precision    recall  f1-score   support

           0       0.84      0.91      0.87       107
           1       0.74      0.60      0.66        47

    accuracy                           0.81       154
   macro avg       0.79      0.75      0.76       154
weighted avg       0.81      0.81      0.81       154
```

In [31]:
```
#Saving the Model
import pickle
pickle_out = open("logmodel.pkl", "wb")
pickle.dump(logmodel, pickle_out)
pickle_out.close()
```

In [33]:
```
import streamlit as st
import pandas as pd
import numpy as np
#import plotly.express as px
#from plotly.subplots import make_subplots
#import plotly.graph_objects as go
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
```

In [34]:
```
st.sidebar.header('Diabetes Prediction')
select = st.sidebar.selectbox('Select Form', ['Form 1'], key='1')
if not st.sidebar.checkbox("Hide", True, key='1'):
    st.title('Diabetes Prediction(Only for females above 21years of    Age)')
    name = st.text_input("Name:")
    pregnancy = st.number_input("No. of times pregnant:")
    glucose = st.number_input("Plasma Glucose Concentration :")
    bp =  st.number_input("Diastolic blood pressure (mm Hg):")
    skin = st.number_input("Triceps skin fold thickness (mm):")
    insulin = st.number_input("2-Hour serum insulin (mu U/ml):")
    bmi = st.number_input("Body mass index (weight in kg/(height in m)^2):")
    dpf = st.number_input("Diabetes Pedigree Function:")
    age = st.number_input("Age:")
submit = st.button('Predict')
if submit:
        prediction = classifier.predict([[pregnancy, glucose, bp, skin, insulin,
        if prediction == 0:
            st.write('Congratulation',name,'You are not diabetic')
        else:
            st.write(name," we are really sorry to say but it seems like you are
```

```
2021-08-13 03:56:33.778
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run C:\Users\fathi\anaconda3\lib\site-packages\ipykernel_launche
r.py [ARGUMENTS]
```

In [ ]:

In [ ]:

In [69]:
```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression()
LR.fit(x_train,y_train)

from sklearn.metrics import accuracy_score,confusion_matrix

print("Train Set Accuracy:"+str(accuracy_score(y_train,LR.predict(x_train))*100))
print("Test Set Accuracy:"+str(accuracy_score(y_test,LR.predict(x_test))*100))
```

```
Train Set Accuracy:76.8729641693811
Test Set Accuracy:81.16883116883116
```

In [38]:

```
from sklearn.svm import SVC
svm=SVC()
svm.fit(x_train,y_train)

print("Train Set Accuracy:"+str(accuracy_score(y_train,svm.predict(x_train))*100)
print("Test Set Accuracy:"+str(accuracy_score(y_test,svm.predict(x_test))*100))
```

```
Train Set Accuracy:76.71009771986971
Test Set Accuracy:79.22077922077922
```