In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [4]:

```
dataset = pd.read_csv("Diabities.csv")
dataset.head(11)
```

Out[4]:

	Pregnancies	Glucose	blood pressure	skin thickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
5	5	116	74	0	0	25.6	0.201	30
6	3	78	50	32	88	31.0	0.248	26
7	10	115	0	0	0	35.3	0.134	29
8	2	197	70	45	543	30.5	0.158	53
9	8	125	96	0	0	0.0	0.232	54
10	4	110	92	0	0	37.6	0.191	30

In [3]:

```
data.shape
```

Out[3]:

```
(768, 9)
```

In [6]:

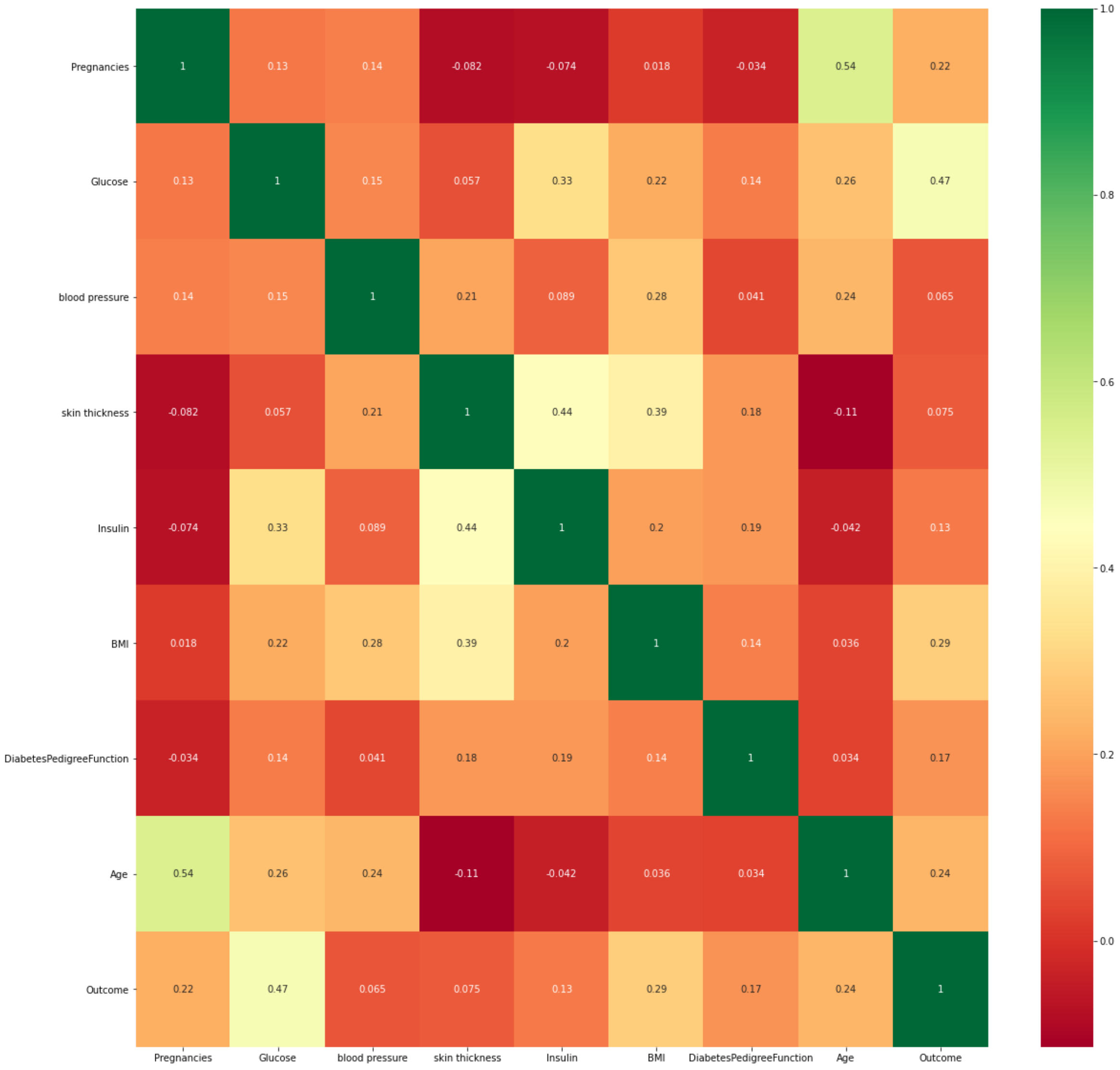
```
# check if any null value is present
dataset.isnull().values.any()
```

Out[6]:

```
False
```

In [7]:

```
## Correlation
import seaborn as sns
import matplotlib.pyplot as plt
#get correlations of each features in dataset
corrmat = dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [8]:
```
dataset.corr()
```

Out[8]:

	Pregnancies	Glucose	blood pressure	skin thickness	Insulin	BMI	Di
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	
blood pressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	
skin thickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	

In [21]:
```
st Split

n.model_selection import train_test_split
umns = ['Pregnancies','Glucose','blood pressure','skin thickness','Insulin','
lass = ['Outcome']
```

In [22]:
```
X = dataset[feature_columns].values
y = dataset[predicted_class].values


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, r
```

In [27]:

```
print("total number of rows : {0}".format(len(data)))
print("number of rows missing Glucose: {0}".format(len(data.loc[data['Glucose
print("number of rows missing Glucose: {0}".format(len(data.loc[data['Glucose
print("number of rows missing blood pressure: {0}".format(len(data.loc[data['
print("number of rows missing skin thickness: {0}".format(len(data.loc[data['
print("number of rows missing Isulin: {0}".format(len(data.loc[data['Insulin'
print("number of rows missing BMI: {0}".format(len(data.loc[data['BMI'] == 0]
print("number of rows missing DiabetesPedigreeFunction: {0}".format(len(data.
print("number of rows missing Age: {0}".format(len(data.loc[data['Age'] == 0]
```

```
total number of rows : 768
number of rows missing Glucose: 5
number of rows missing Glucose: 5
number of rows missing blood pressure: 35
number of rows missing skin thickness: 227
number of rows missing Isulin: 374
number of rows missing BMI: 11
number of rows missing DiabetesPedigreeFunction: 0
number of rows missing Age: 0
```

In [35]:

```
from sklearn import preprocessing
```

In [36]:

```
import numpy as np
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')

fill_values = SimpleImputer(missing_values=0, strategy="mean", fill_value=Non

X_train = fill_values.fit_transform(X_train)
X_test = fill_values.fit_transform(X_test)
```

In [40]:

```
## Apply Algorithm

from sklearn.ensemble import RandomForestClassifier
random_forest_model = RandomForestClassifier(random_state=0)

random_forest_model.fit(X_train, y_train.ravel())
```

Out[40]:

```
RandomForestClassifier(random_state=0)
```

In [46]:

```
## RANDOM FOREST
predict_train_data = random_forest_model.predict(X_test)
from sklearn import metrics
print("Accuracy = {0:.3f}".format(metrics.accuracy_score(y_test, predict_trai
```

```
Accuracy = 0.766
```

In [42]:

```
## Hyper Parameter Optimization

params={
 "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"        : [ 3, 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]

}
```

In [47]:

```
#LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,r2_score,classification_report
logreg = LogisticRegression(solver='lbfgs',max_iter=1000)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
acc_logreg1 = round(accuracy_score(y_pred, y_test) , 2)*100
print("Accuracy :",acc_logreg1)
```

```
Accuracy : 77.0

C:\Users\TinkuBablu\anaconda3\lib\site-packages\sklearn\utils\validation.p
y:63: DataConversionWarning: A column-vector y was passed when a 1d array w
as expected. Please change the shape of y to (n_samples, ), for example usi
ng ravel().
  return f(*args, **kwargs)
```

In [ ]: