DIABETES PREDICTION SYSTEM

Importing required libraries

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
%matplotlib inline
```

Importing the csv file from folder and reading

In [7]:

```
data=pd.read_csv("C:\\Users\\indra\\Documents\\csvfiles\\diabetes.csv")
data.head(10)
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28
5	5	116	74	0	0	25.6	0.20
6	3	78	50	32	88	31.0	0.24
7	10	115	0	0	0	35.3	0.13
8	2	197	70	45	543	30.5	0.15
9	8	125	96	0	0	0.0	0.23

In [8]:

```
#checking the number of rows and columns in the file
data.shape
```

Out[8]:

```
(768, 9)
```

In [9]:

```
#checking for null values in the data
data.isnull().values.any()
```

Out[9]:

```
False
```

Renaming the coloumns

In [10]:

```
data.rename(columns={'DiabetesPedigreeFunction':'DPF','BloodPressure':'BP'},inplace=True)
data.head(5)
```

Out[10]:

	Pregnancies	Glucose	BP	SkinThickness	Insulin	BMI	DPF	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [11]:

```
data.describe()
```

Out[11]:

	Pregnancies	Glucose	BP	SkinThickness	Insulin	BMI	DPF
count	768.000000	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	0.471876
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000

Plotting corelation heat map

In [12]:

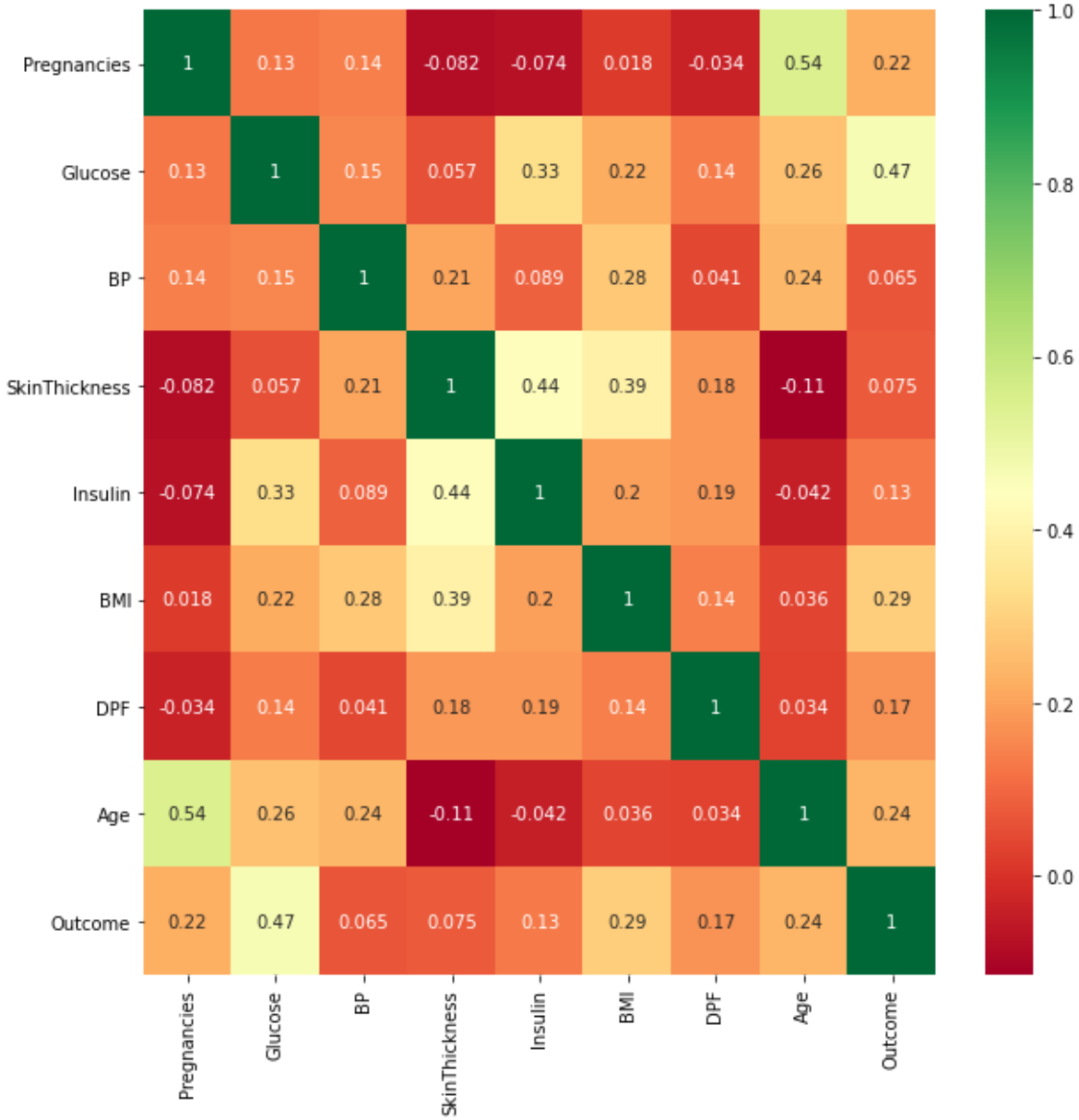
```
#correlation table
data.corr()
```

Out[12]:

	Pregnancies	Glucose	BP	SkinThickness	Insulin	BMI	DPF
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337
BP	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647
DPF	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844

In [13]:

```
#plotting corelation table using heatmap
corrmat=data.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(10,10))
g=sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Check for zero values in the columns

In [14]:

```
print("Number of zeros in Glocose : ",data[data["Glucose"]==0].shape[0])
print("Number of zeros in Blood Pressure : ",data[data["BP"]==0].shape[0])
print("Number of zeros in Skin Thickness : ",data[data["SkinThickness"]==0].shape[0])
print("Number of zeros in Insulin : ",data[data["Insulin"]==0].shape[0])
print("Number of zeros in BMI : ",data[data["BMI"]==0].shape[0])
print("Number of zeros in Diabetes Prediction Factor : ",data[data["DPF"]==0].shape[0])
print("Number of zeros in Age : ",data[data["Age"]==0].shape[0])
```

```
Number of zeros in Glocose :  5
Number of zeros in Blood Pressure :  35
Number of zeros in Skin Thickness :  227
Number of zeros in Insulin :  374
Number of zeros in BMI :  11
Number of zeros in Diabetes Prediction Factor :  0
Number of zeros in Age :  0
```

Placing the zero values with mean value of the respective column

In [15]:

```
data["Glucose"]=data["Glucose"].replace(0,data["Glucose"].mean())
data["BP"]=data["BP"].replace(0,data["BP"].mean())
data["SkinThickness"]=data["SkinThickness"].replace(0,data["SkinThickness"].mean())
data["Insulin"]=data["Insulin"].replace(0,data["Insulin"].mean())
data["BMI"]=data["BMI"].replace(0,data["BMI"].mean())
data["DPF"]=data["DPF"].replace(0,data["DPF"].mean())
data["Age"]=data["Age"].replace(0,data["Age"].mean())

print("Number of zeros in Glocose : ",data[data["Glucose"]==0].shape[0])
print("Number of zeros in Blood Pressure : ",data[data["BP"]==0].shape[0])
print("Number of zeros in Skin Thickness : ",data[data["SkinThickness"]==0].shape[0])
print("Number of zeros in Insulin : ",data[data["Insulin"]==0].shape[0])
print("Number of zeros in BMI : ",data[data["BMI"]==0].shape[0])
print("Number of zeros in Diabetes Prediction Factor : ",data[data["DPF"]==0].shape[0])
print("Number of zeros in Age : ",data[data["Age"]==0].shape[0])
```

```
Number of zeros in Glocose :  0
Number of zeros in Blood Pressure :  0
Number of zeros in Skin Thickness :  0
Number of zeros in Insulin :  0
Number of zeros in BMI :  0
Number of zeros in Diabetes Prediction Factor :  0
Number of zeros in Age :  0
```

In [16]:

```
#counting the total individual Outcomes
positive_outcome=len(data.loc[data["Outcome"]==1])
negative_outcome=len(data.loc[data["Outcome"]==0])
(positive_outcome,negative_outcome)
```

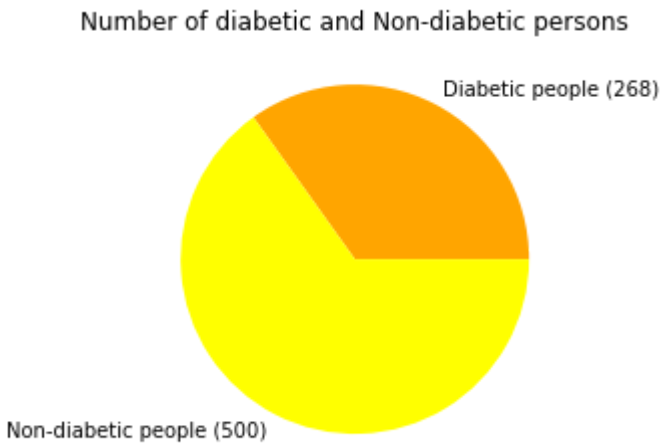
Out[16]:

```
(268, 500)
```

Pie Chart Visualisation

In [17]:

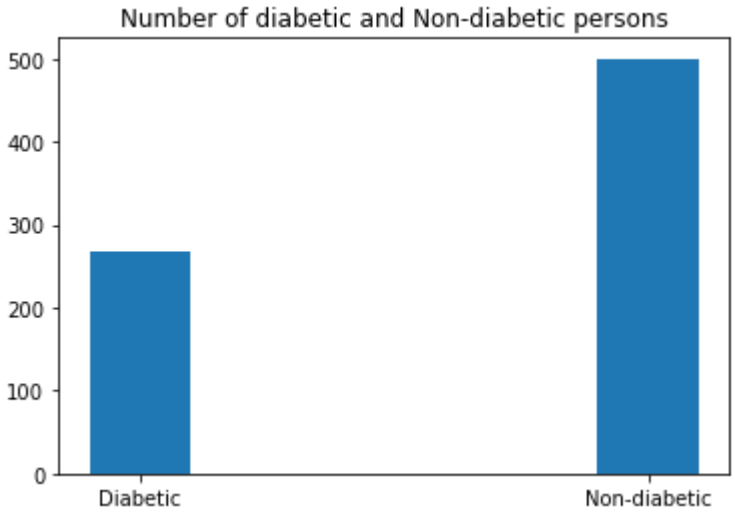
```
y=np.array([positive_outcome,negative_outcome])
mylabels=["Diabetic people (268)","Non-diabetic people (500)"]
plt.pie(y,labels=mylabels,colors=["orange","yellow"])
plt.title("Number of diabetic and Non-diabetic persons")
plt.show()
```



Bar Chart Visualization

In [18]:

```
df={'Diabetic':positive_outcome,'Non-diabetic':negative_outcome}
A=list(df.keys())
B=list(df.values())
plt.bar(A,B,width=0.2)
plt.title("Number of diabetic and Non-diabetic persons")
plt.show()
```



Test,Train and Split

In [19]:

```
X=data.drop(columns=["Outcome"])
Y=data["Outcome"]
X_test,X_train,Y_test,Y_train=train_test_split(X,Y,test_size=0.30,random_state=10)
```

Training model and prediction

In [20]:

```
model=RandomForestClassifier(random_state=10)
model.fit(X_train,Y_train.ravel())
pred=model.predict(X_test)
pred
```

Out[20]:

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

Accuracy

In [21]:

```
acc=metrics.accuracy_score(Y_test,pred)
print("ACCURACY OF THE MODEL : ",acc)
```

```
ACCURACY OF THE MODEL :  0.750465549348231
```

Prediction based on user inputs

In [22]:

```
def prediction_calculator(n):
    for i in range(n):
        print("\n ENTER THE DETAILS FOR PERSON : ",(i+1))
        Age_ip=input("\nAge : ")
        Gender=input('Gender (f/F/m/M): ')
        if Gender=='f' or Gender=='F':
             Preg_ip=input("Number of Pregnancies : ")
        else:
            Preg_ip=0
        Bmi_ip=input("BMI : ")
        Glucose_ip=input("Glucose level : ")
        Insulin_ip=input("Insulin level : ")
        Bp_ip=input("BP level : ")
        St_ip=input("Skin Thickness : ")
        Dpf_ip=input("Diabetes prediction factor : ")

        c=np.array([Preg_ip,Glucose_ip,Bp_ip,St_ip,Insulin_ip,Bmi_ip,Dpf_ip,Age_ip])
        c_rs=c.reshape(1,-1)
        pred=model.predict(c_rs)

        if pred==1:
            print("DIABETIC PERSON !!")
        else:
            print("NON-DIABETIC PERSON :)")
```

In [25]:

```
no_of_people=int(input("\n ENTER NUMBER OF PEOPLE : "))
prediction_calculator(no_of_people)
```

```
 ENTER NUMBER OF PEOPLE : 1

 ENTER THE DETAILS FOR PERSON :  1

Age : 25
Gender (f/F/m/M): m
BMI : 36.9
Glucose level : 56
Insulin level : 98
BP level : 78
Skin Thickness : 33
Diabetes prediction factor : 67
NON-DIABETIC PERSON :)
```

In [ ]: