Healthcare

Project by Asmita Dahiya

DESCRIPTION

Problem Statement

- NIDDK (National Institute of Diabetes and Digestive and Kidney Diseases) research creates knowledge about and treatments for the most chronic, costly, and consequential diseases.
- The dataset used in this project is originally from NIDDK. The objective is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Build a model to accurately predict whether the patients in the dataset have diabetes or not.

Dataset Description

The datasets consists of several medical predictor variables and one target variable (Outcome). Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and more.

Variables	**Description**
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration in an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skinfold thickness (mm)
Insulin	Two hour serum insulin
BMI	Body Mass Index
DiabetesPedigreeFunction	Diabetes pedigree function
Age	Age in years
Outcome	Class variable (either 0 or 1). 268 of 768 values are 1, and the others are 0

Project Task:

PART-I

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

• Glucose

• BloodPressure

• SkinThickness

• Insulin

• BMI

2. Visually explore these variables using histograms. Treat the missing values accordingly.

3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

Project Task:

Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.

2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

3. Perform correlation analysis. Visually explore it using a heat map.

Project Task:

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.

2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

Project Task:

Data Modeling:

1. Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc. Please be descriptive to explain what values of these parameter you have used.

```
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
df= pd.read_csv('health care diabetes.csv')
```

In [3]:
```
df
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3

2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

In [4]:
```
df.shape
```

Out[4]:
```
(768, 9)
```

In [5]:
```
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 [6]:
```
df.isnull().any()
```

Out[6]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

In [7]:
```
df.head()
```

Out[7]:

	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

In [8]:
```
df.tail()
```
Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

In [9]:
```
df.describe()
```
Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete
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 [10]:
```
df.count()
```
Out[10]:
```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [11]:
```
#Data Exploration: 1. Perform descriptive analysis. Understand the variables a
nd their corresponding values. On the columns below, a value of zero does not
make sense and thus indicates missing value:
#Glucose
#BloodPressure
#SkinThickness
#Insulin
#BMI
# 2.Visually explore these variables using histograms. Treat the missing value
s accordingly.
# 3.There are integer and float data type variables in this dataset. Create a
 count (frequency) plot describing the data types and the count of variables.
```

In [12]:
```
df['Glucose'].value_counts()
```

Out[12]:
```
100    17
99     17
129    14
125    14
111    14
       ..
177     1
172     1
169     1
160     1
199     1
Name: Glucose, Length: 136, dtype: int64
```

In [13]:
```
df['BloodPressure'].value_counts()
```

Out[13]:
```
70     57
74     52
68     45
78     45
72     44
64     43
80     40
76     39
60     37
0      35
62     34
66     30
82     30
88     25
84     23
90     22
86     21
58     21
50     13
56     12
52     11
54     11
92      8
75      8
65      7
94      6
85      6
48      5
44      4
96      4
110     3
100     3
98      3
106     3
108     2
104     2
30      2
55      2
46      2
40      1
38      1
24      1
95      1
61      1
102     1
114     1
122     1
Name: BloodPressure, dtype: int64
```

In [14]:
```
df['SkinThickness'].value_counts()
```

Out[14]:
```
0     227
32     31
30     27
27     23
23     22
33     20
18     20
28     20
31     19
39     18
19     18
29     17
37     16
26     16
22     16
40     16
25     16
35     15
41     15
36     14
15     14
17     14
20     13
24     12
42     11
13     11
21     10
34      8
46      8
38      7
12      7
14      6
16      6
11      6
43      6
45      6
10      5
44      5
48      4
47      4
50      3
49      3
54      2
52      2
7       2
8       2
60      1
56      1
63      1
51      1
99      1
Name: SkinThickness, dtype: int64
```

In [15]:
```
df['Insulin'].value_counts()
```

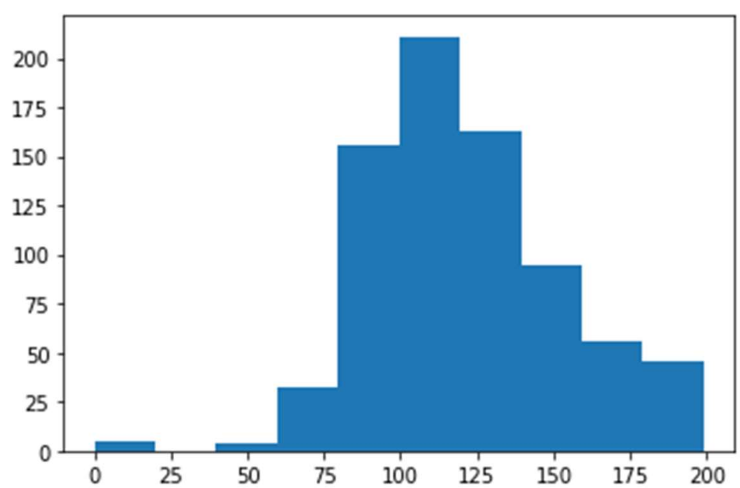
Out[15]:
```
0      374
105     11
140      9
130      9
120      8
      ...
271      1
270      1
108      1
112      1
846      1
Name: Insulin, Length: 186, dtype: int64
```

In [16]:
```
df['BMI'].value_counts()
```

Out[16]:
```
32.0    13
31.6    12
31.2    12
0.0     11
33.3    10
        ..
32.1     1
52.9     1
31.3     1
45.7     1
42.8     1
Name: BMI, Length: 248, dtype: int64
```

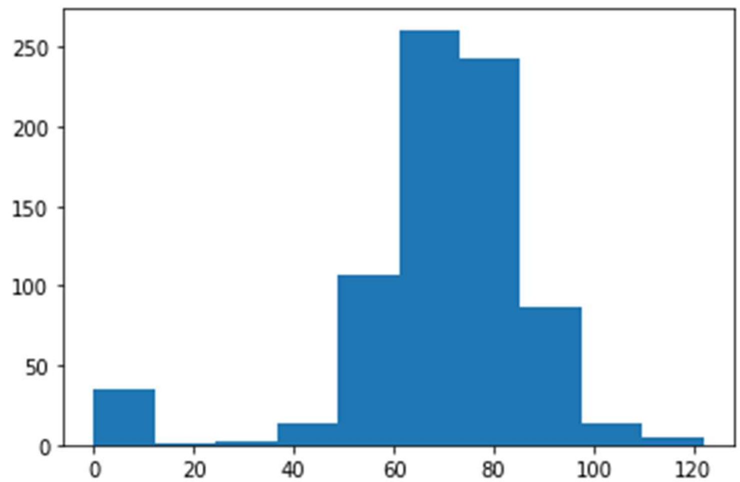
In [17]:
```
plt.hist(df['Glucose'])
```

Out[17]:
```
(array([  5.,   0.,   4.,  32., 156., 211., 163.,  95.,  56.,  46.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <BarContainer object of 10 artists>)
```



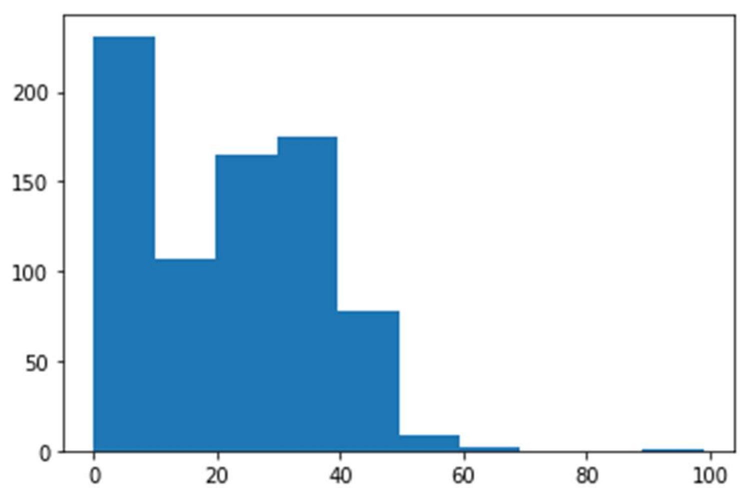
In [18]:
```
plt.hist(df['BloodPressure'])
```

Out[18]:
```
(array([ 35.,   1.,   2.,  13., 107., 261., 243.,  87.,  14.,   5.]),
 array([  0. ,  12.2,  24.4,  36.6,  48.8,  61. ,  73.2,  85.4,  97.6,
        109.8, 122. ]),
 <BarContainer object of 10 artists>)
```



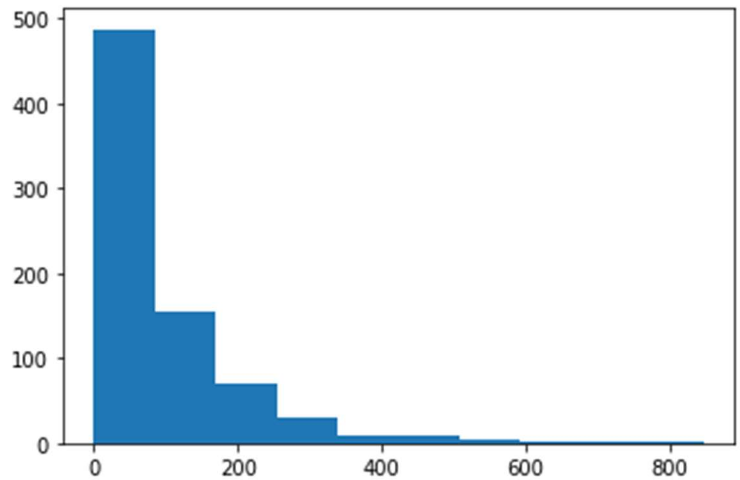
In [19]:
```
plt.hist(df['SkinThickness'])
```

Out[19]:
```
(array([231., 107., 165., 175.,  78.,   9.,   2.,   0.,   0.,   1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <BarContainer object of 10 artists>)
```



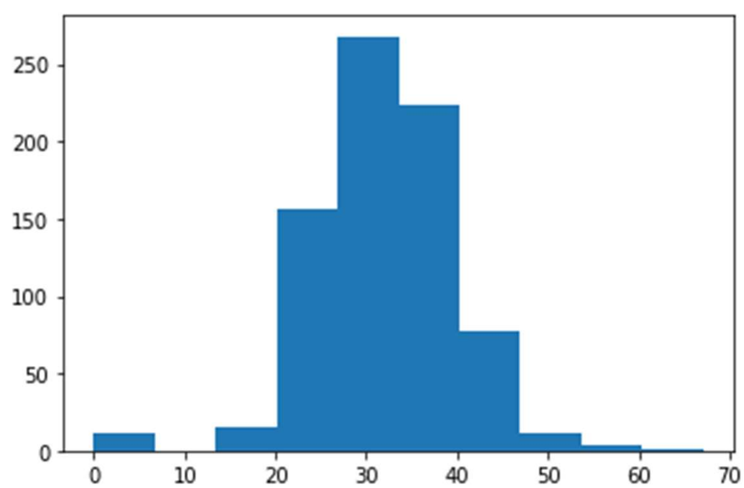
In [20]:
```
plt.hist(df['Insulin'])
```

Out[20]:
```
(array([487., 155.,  70.,  30.,   8.,   9.,   5.,   1.,   2.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <BarContainer object of 10 artists>)
```



In [21]:
```
plt.hist(df['BMI'])
```

Out[21]:
```
(array([ 11.,   0.,  15., 156., 268., 224.,  78.,  12.,   3.,   1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <BarContainer object of 10 artists>)
```



In [22]:
```
df.dtypes.count()
```

Out[22]: 9

In [23]:
```
df.dtypes
```

Out[23]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [24]:
```
df.count()
```

Out[24]:
```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
Outcome                     768
dtype: int64
```

In [25]:
```
#Data Exploration:1.Check the balance of the data by plotting the count of out
comes by their value.
# 2.Create scatter charts between the pair of variables to understand the rela
tionships.
# 3. Perform correlation analysis. Visually explore it using a heat map.
```

In [26]:
```
outcome_0 = df[df.Outcome ==0] # Outcome = 0 (i.e) Non-Diabetic Patient
outcome_1 = df[df.Outcome ==1] # Outcome = 1 (i.e) Diabetic Patient
```

In [27]:
```
outcome_0
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
1	1	85	66	29	0	26.6	0.3
3	1	89	66	23	94	28.1	0.1
5	5	116	74	0	0	25.6	0.2
7	10	115	0	0	0	35.3	0.1
10	4	110	92	0	0	37.6	0.1
...	...	...	...	...	...	...	
762	9	89	62	0	0	22.5	0.1
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
767	1	93	70	31	0	30.4	0.3

500 rows × 9 columns

In [28]:
```
outcome_0.describe()
```

Out[28]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesP
count	500.000000	500.0000	500.000000	500.000000	500.000000	500.000000	
mean	3.298000	109.9800	68.184000	19.664000	68.792000	30.304200	
std	3.017185	26.1412	18.063075	14.889947	98.865289	7.689855	
min	0.000000	0.0000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	93.0000	62.000000	0.000000	0.000000	25.400000	
50%	2.000000	107.0000	70.000000	21.000000	39.000000	30.050000	
75%	5.000000	125.0000	78.000000	31.000000	105.000000	35.300000	
max	13.000000	197.0000	122.000000	60.000000	744.000000	57.300000	

In [29]:
```
outcome_1
```

Out[29]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
2	8	183	64	0	0	23.3	0.6
4	0	137	40	35	168	43.1	2.2
6	3	78	50	32	88	31.0	0.2
8	2	197	70	45	543	30.5	0.1
...	...	...	...	...	...	...	
755	1	128	88	39	110	36.5	1.0
757	0	123	72	0	0	36.3	0.2
759	6	190	92	0	0	35.5	0.2
761	9	170	74	31	0	44.0	0.4
766	1	126	60	0	0	30.1	0.3

268 rows × 9 columns

In [30]:
```
outcome_1.describe()
```

Out[30]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete
count	268.000000	268.000000	268.000000	268.000000	268.000000	268.000000	
mean	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	
std	3.741239	31.939622	21.491812	17.679711	138.689125	7.262967	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.750000	119.000000	66.000000	0.000000	0.000000	30.800000	
50%	4.000000	140.000000	74.000000	27.000000	0.000000	34.250000	
75%	8.000000	167.000000	82.000000	36.000000	167.250000	38.775000	
max	17.000000	199.000000	114.000000	99.000000	846.000000	67.100000	

In [32]:
```
outcome_0=outcome_0.drop('Outcome',axis=1)
```

In [33]:
```
outcome_0
```
Out[33]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
1	1	85	66	29	0	26.6	0.3
3	1	89	66	23	94	28.1	0.1
5	5	116	74	0	0	25.6	0.2
7	10	115	0	0	0	35.3	0.1
10	4	110	92	0	0	37.6	0.1
...	...	...	...	...	...	...	
762	9	89	62	0	0	22.5	0.1
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
767	1	93	70	31	0	30.4	0.3

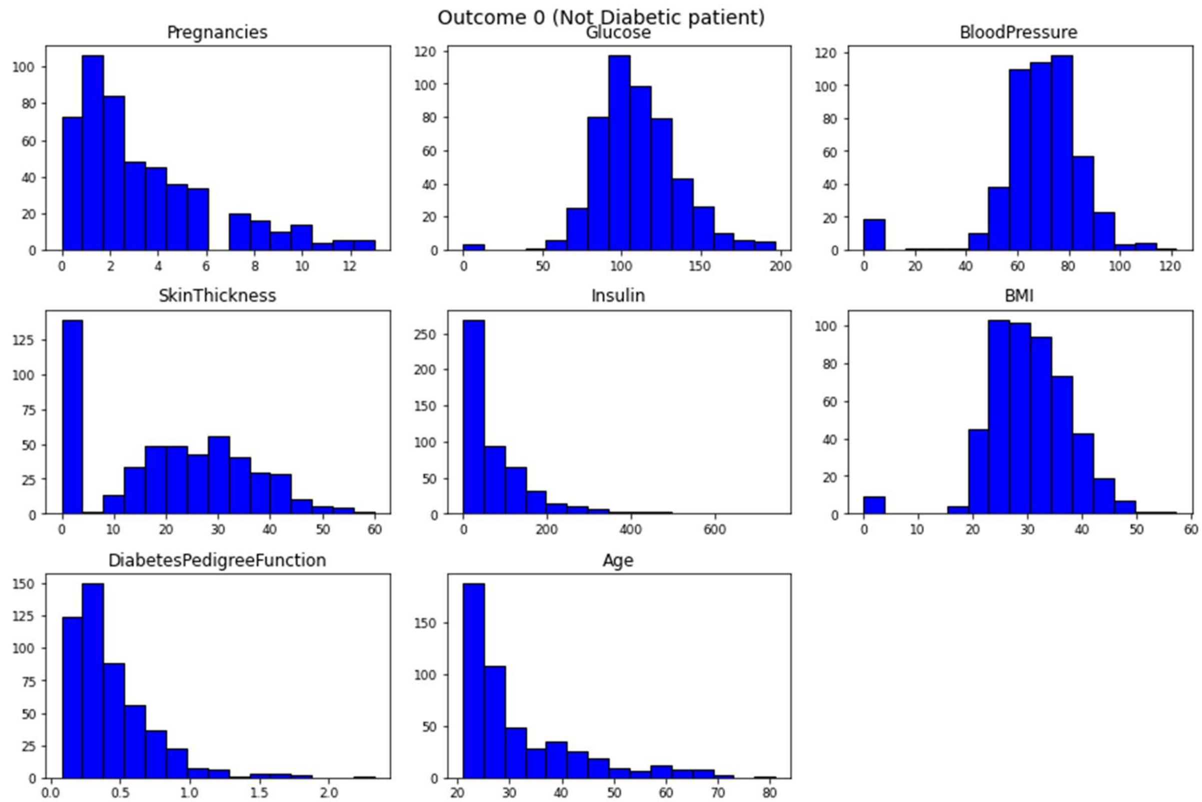
500 rows × 8 columns

In [36]:
```
outcome_0.hist(bins=15, color='blue', edgecolor='black', linewidth=1.0,xlabelsize=9, ylabelsize=9, grid=False)

plt.tight_layout(rect=(0, 0, 2, 2)) # it will change the size of the plot

plt.suptitle('Outcome 0 (Not Diabetic patient)',
             x=1, # title x position
             y=2, # title y position
             fontsize=14)
```

Out[36]:
```
Text(1, 2, 'Outcome 0 (Not Diabetic patient)')
```



In [37]:
```
outcome_1=outcome_1.drop('Outcome',axis=1)
```

In [38]:
```
outcome_1
```

Out[38]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
2	8	183	64	0	0	23.3	0.6
4	0	137	40	35	168	43.1	2.2
6	3	78	50	32	88	31.0	0.2
8	2	197	70	45	543	30.5	0.1
...	...	...	...	...	...	...	
755	1	128	88	39	110	36.5	1.0
757	0	123	72	0	0	36.3	0.2
759	6	190	92	0	0	35.5	0.2
761	9	170	74	31	0	44.0	0.4
766	1	126	60	0	0	30.1	0.3

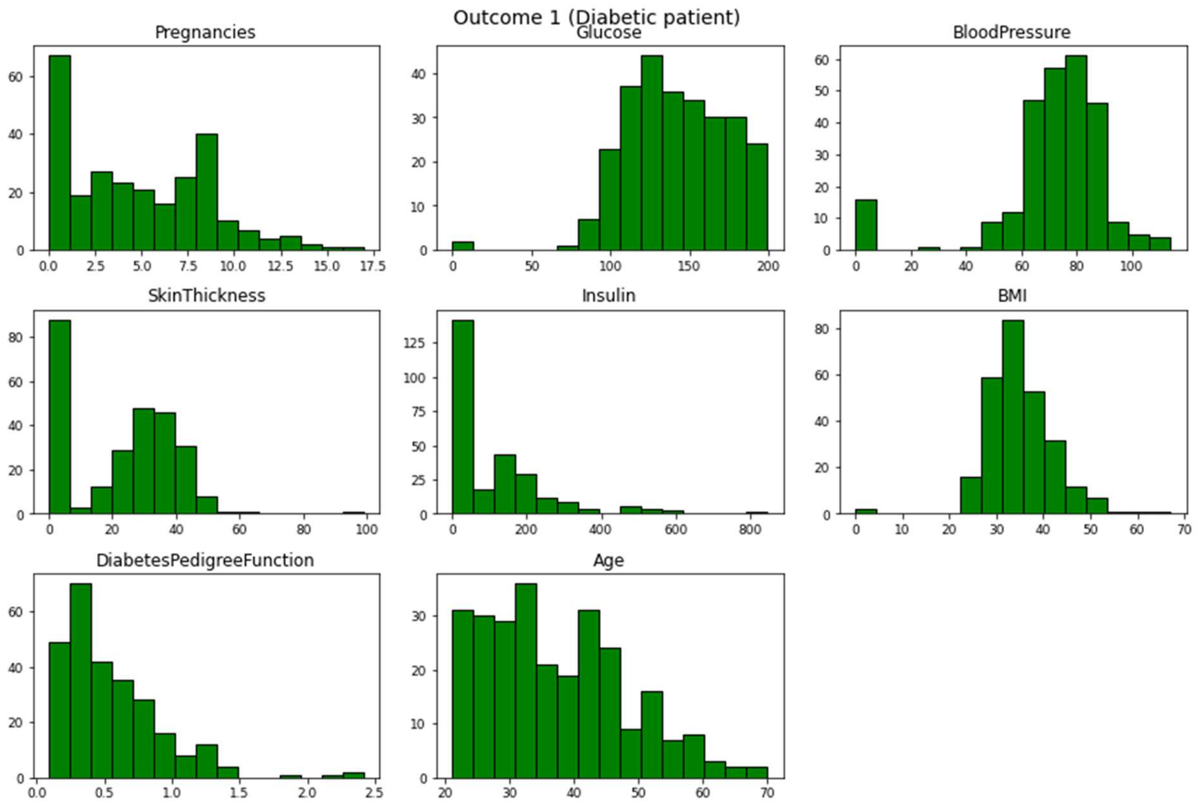
268 rows × 8 columns

In [39]:
```python
outcome_1.hist(bins=15, color='green', edgecolor='black', linewidth=1.0,xlabel
size=9, ylabelsize=9, grid=False)

plt.tight_layout(rect=(0, 0, 2, 2)) # it will change the size of the plot

plt.suptitle('Outcome 1 (Diabetic patient)',
             x=1, # title x position
             y=2, # title y position
             fontsize=14)
```

Out[39]: `Text(1, 2, 'Outcome 1 (Diabetic patient)')`



In [40]:
```python
df_1=df.drop('Outcome',axis=1)
```

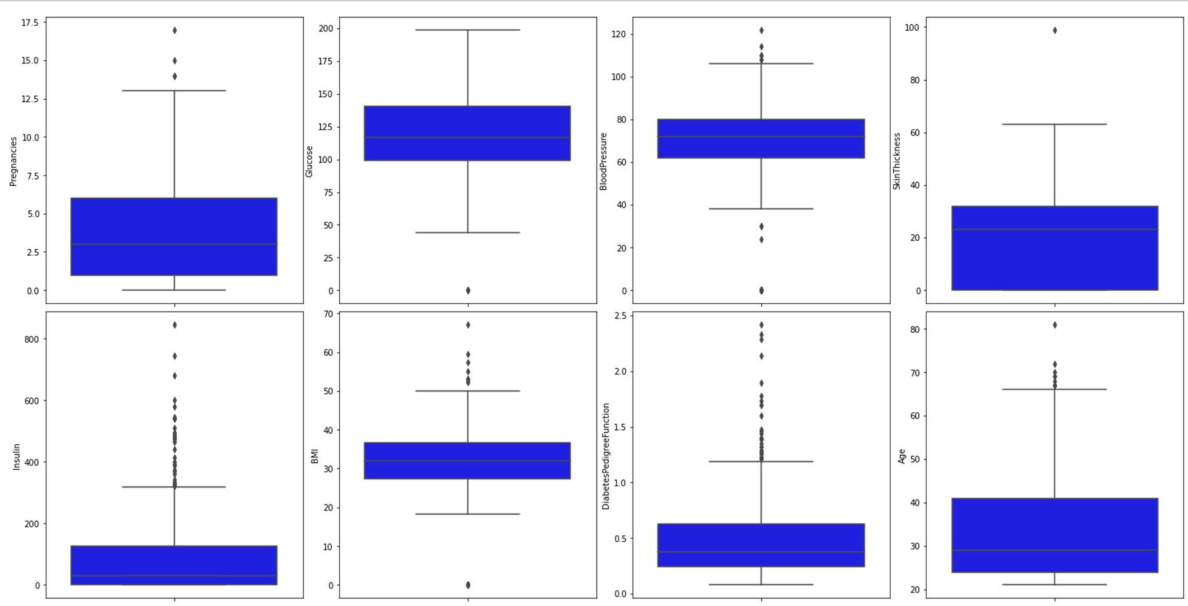
In [41]:
```
df_1
```

Out[41]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 8 columns

In [44]:
```
fig , ax = plt.subplots(nrows= 2,        # no,of plots comes in row wise
                        ncols= 4,        # no,of plots comes in column wise
                        figsize=(20,10) # size of plot
                        )
ax = ax.flatten() # It returns a flattened version of the array, to avoid nump
y.ndarray
index = 0
for i in df_1.columns:
  sns.boxplot(y=i,data = df_1, ax=ax[index],color='blue')
  index += 1
plt.tight_layout(pad=0.4)
```

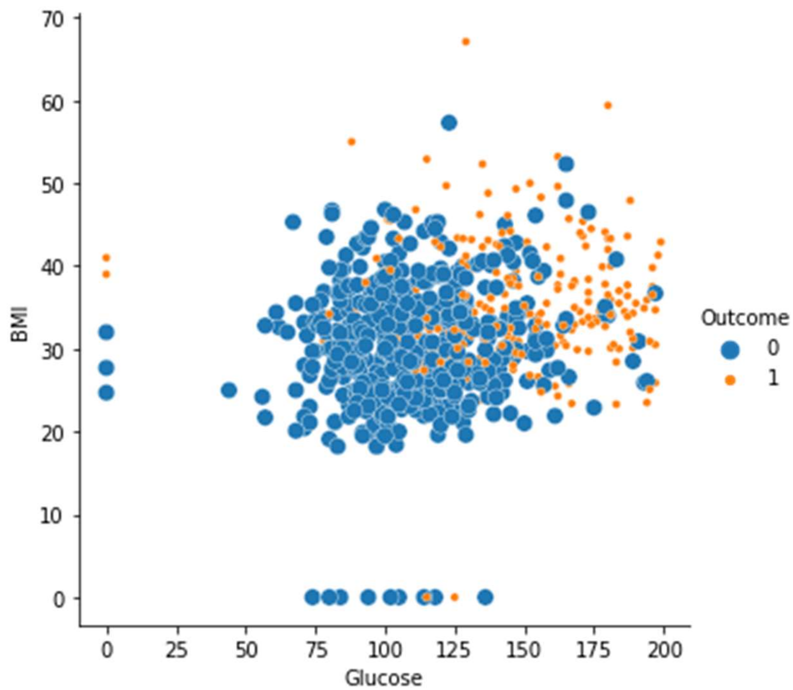


In [45]:
```python
sns.pairplot(df,                # dataset
             hue='Outcome',     # variable in dataset to map plot aspects to different color
             palette='husl',
             )
```

Out[45]:
```
<seaborn.axisgrid.PairGrid at 0x7f31b8576790>
```

In [46]:
```
sns.relplot(x='Glucose',
            y='BMI',
            data = df,
            hue = 'Outcome',
            size='Outcome')
```

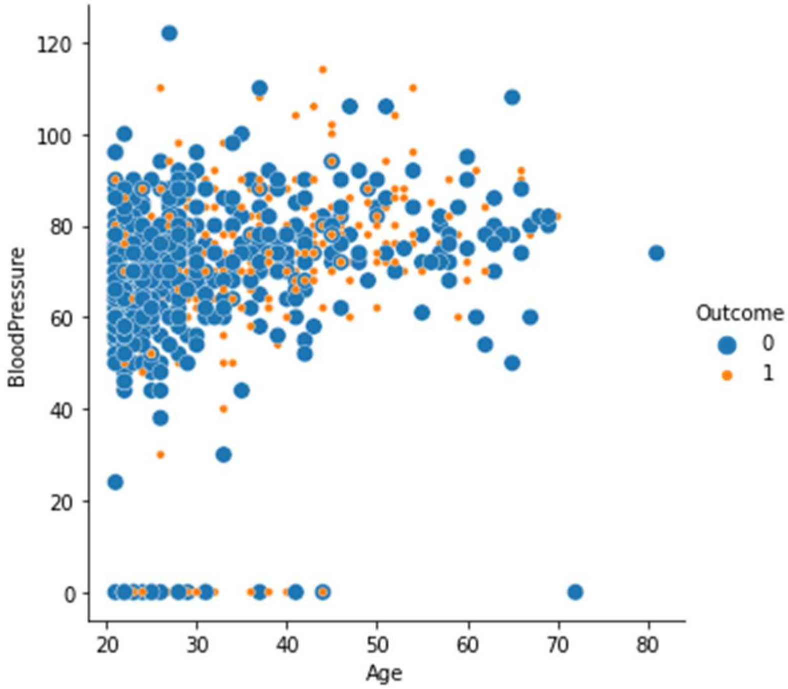
Out[46]: <seaborn.axisgrid.FacetGrid at 0x7f31b442d490>



In [47]:
```
#The above scatter plot tells, the people who have abnormal BMI and higher the Glucose level, will have higher the chance of getting Diabetic(orange small dots)
```

In [49]:
```
sns.relplot(x='Age',
            y='BloodPressure',
            data = df,
            hue = 'Outcome',
            size='Outcome')
```

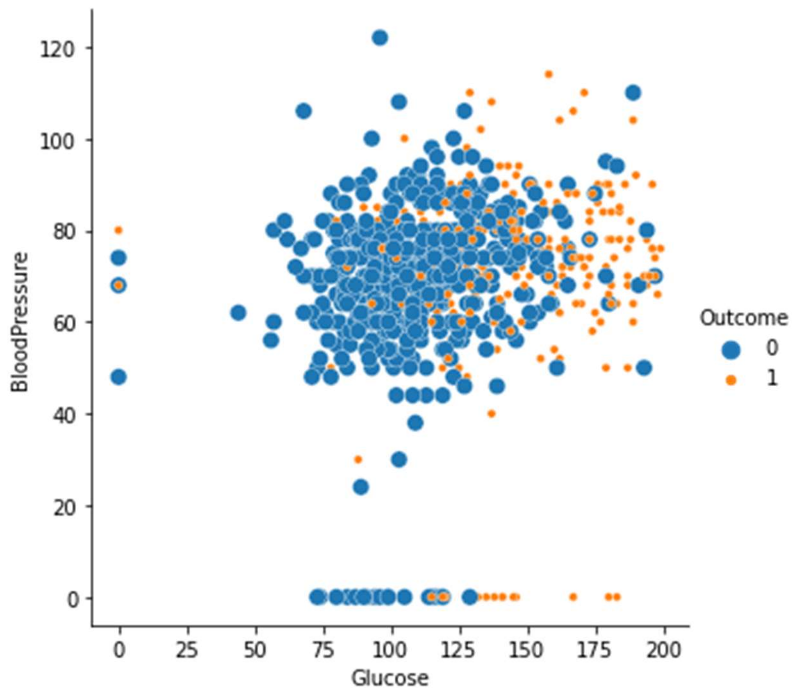
Out[49]: <seaborn.axisgrid.FacetGrid at 0x7f31b3851a10>



In [50]:
```
#the above plot shows that, higher the chance for people have High BloodPressu
re and getting Aged to be a Diabetic.
```

In [51]:
```
sns.relplot(x='Glucose',
            y='BloodPressure',
            data = df,
            hue = 'Outcome',
            size='Outcome')
```

Out[51]: <seaborn.axisgrid.FacetGrid at 0x7f31b2001e90>



In [52]:
```
#the above plot shows that, higher the chance for people have High BloodPressu
re and Glucose to be a Diabetic.
```

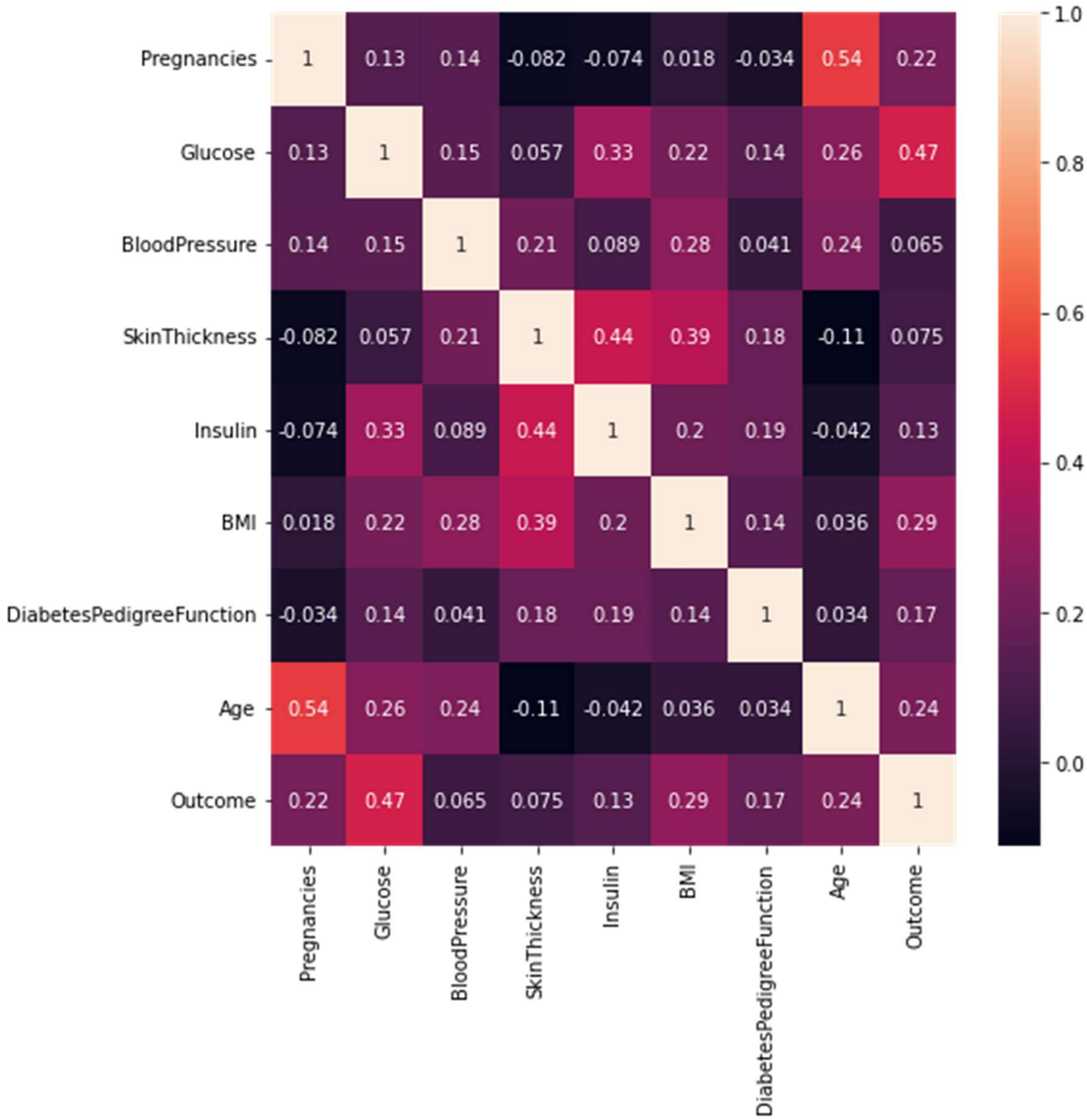
In [53]:
```
df.corr()
```

Out[53]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.01
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.22
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.28
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.39
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.19
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.00
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.14
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.03
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.29

In [54]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)  ### gives correlation value
```

Out[54]: <AxesSubplot:>



In [55]:
```
#Data Modeling:1. Devise strategies for model building. It is important to dec
ide the right validation framework. Express your thought process.
#2. Apply an appropriate classification algorithm to build a model. Compare va
rious models with the results from KNN algorithm.
```

In [56]:
```
#Train test split
from sklearn.model_selection import train_test_split
```

In [57]:
```
features = df.iloc[:,[0,1,2,3,4,5,6,7]].values
label = df.iloc[:,8].values
```

In [58]:
```
X_train,X_test,y_train,y_test = train_test_split(features,
                                                 label,
                                                 test_size=0.2,
                                                 random_state =10)
```

In [59]:
```
#Create model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[59]:
```
LogisticRegression()
```

In [60]:
```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7719869706840391
0.7662337662337663
```

In [82]:
```
from sklearn.metrics import accuracy_score
y_pred_lr = model.predict(X_test)
print('test accuracy : ', accuracy_score(y_pred_lr,y_test))
```

```
test accuracy :  0.7662337662337663
```

In [61]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

Out[61]:
```
array([[446,  54],
       [122, 146]])
```

In [62]:
```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       500
           1       0.73      0.54      0.62       268

    accuracy                           0.77       768
   macro avg       0.76      0.72      0.73       768
weighted avg       0.77      0.77      0.76       768
```

In [63]:
```
# Tree Model
from sklearn.tree import DecisionTreeClassifier
```

In [64]:
```
dtc = DecisionTreeClassifier(criterion="entropy",  # For the information gain
                             splitter="best",      # For the best split
                             random_state=9
                            )
```

In [66]:
```
dtc.fit(X_train,y_train)
y_pred_dtc = dtc.predict(X_test)
```

In [70]:
```
print('test accuracy : ', accuracy_score(y_pred_dtc,y_test))
```

```
test accuracy :  0.7272727272727273
```

In [72]:
```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(max_depth=2,
                             random_state=0,
                             n_jobs=-1)
rfc.fit(X_train,y_train)
y_pred_rfc = dtc.predict(X_test)
print('test accuracy : ', accuracy_score(y_pred_rfc,y_test))
```

```
test accuracy :  0.7272727272727273
```

In [73]:
```
#Data Modeling: Create a classification report by analyzing sensitivity, speci
ficity, AUC (ROC curve), etc.
```

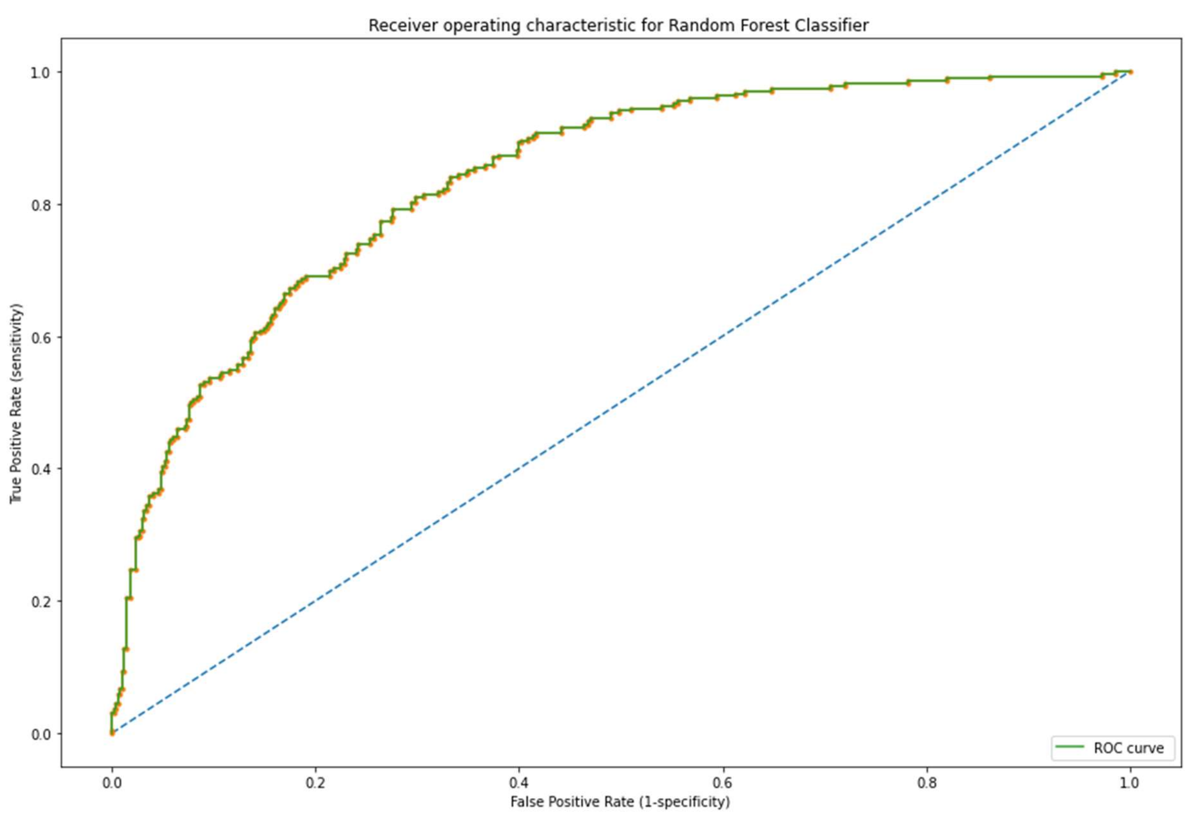
In [77]:

```python
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

plt.figure()
plt.subplots(figsize=(15,10))
# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate (1-specificity)')
plt.ylabel('True Positive Rate (sensitivity)')
plt.title('Receiver operating characteristic for Random Forest Classifier ')
plt.plot(fpr, tpr, label = 'ROC curve ')
plt.legend(loc ="lower right")
plt.show()
```

```
AUC: 0.837
```

```
<Figure size 432x288 with 0 Axes>
```



In [78]:
```
#Support Vector Classifier

from sklearn.svm import SVC
modelsvm = SVC(kernel='rbf',
           gamma='auto')
modelsvm.fit(X_train,y_train)
```

Out[78]:
```
SVC(gamma='auto')
```

In [79]:
```
modelsvm.score(X_test,y_test)
```

Out[79]:
```
0.6168831168831169
```

In [80]:
```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
modelknn = KNeighborsClassifier(n_neighbors=7,
                             metric='minkowski',
                             p = 2)
modelknn.fit(X_train,y_train)
```

Out[80]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [81]:

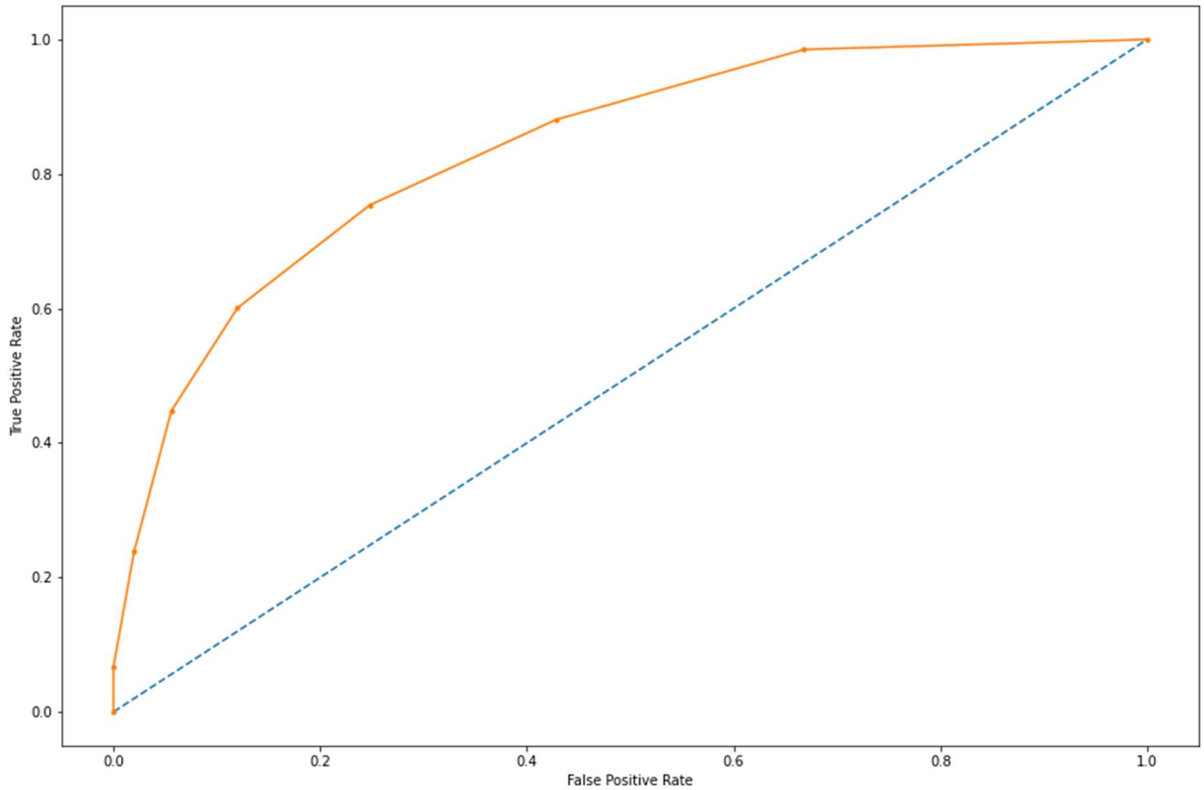
```
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

plt.figure()
plt.subplots(figsize=(15,10))
# predict probabilities
probs = modelknn.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# calculate AUC
auc = roc_auc_score(label, probs)
print('AUC: %.3f' % auc)
# calculate roc curve
fpr, tpr, thresholds = roc_curve(label, probs)
print("True Positive Rate - {}, False Positive Rate - {} Thresholds - {}".form
at(tpr,fpr,thresholds))
# plot no skill
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
AUC: 0.836
True Positive Rate - [0.         0.06716418 0.23880597 0.44776119 0.60074627
0.75373134
 0.88059701 0.98507463 1.        ], False Positive Rate - [0.    0.    0.02
0.056 0.12  0.248 0.428 0.668 1.   ] Thresholds - [2.         1.         0.85
714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]
```

Out[81]:
```
Text(0, 0.5, 'True Positive Rate')
```

```
<Figure size 432x288 with 0 Axes>
```



In [83]:
```
#Logistic Regression model gives the better accuracy when compared with other
 models.
```

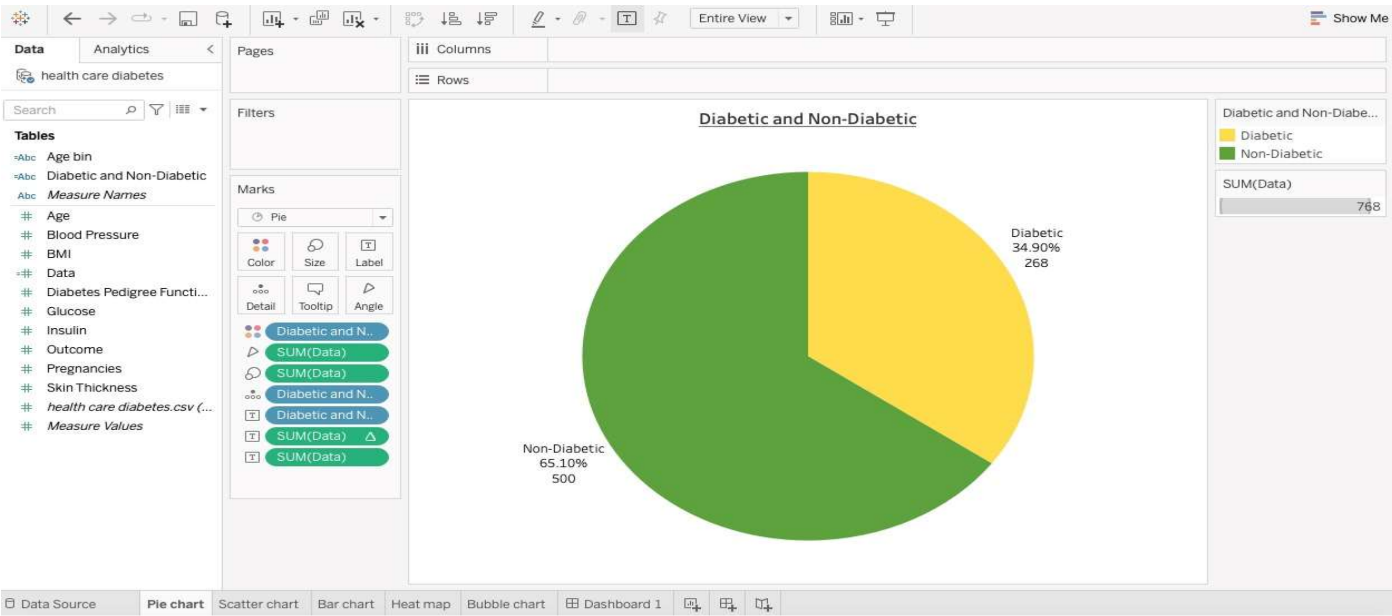
PART-II

Link: https://public.tableau.com/app/profile/asmita.dahiya/viz/FinalProject_HealthCare/Dashboard1

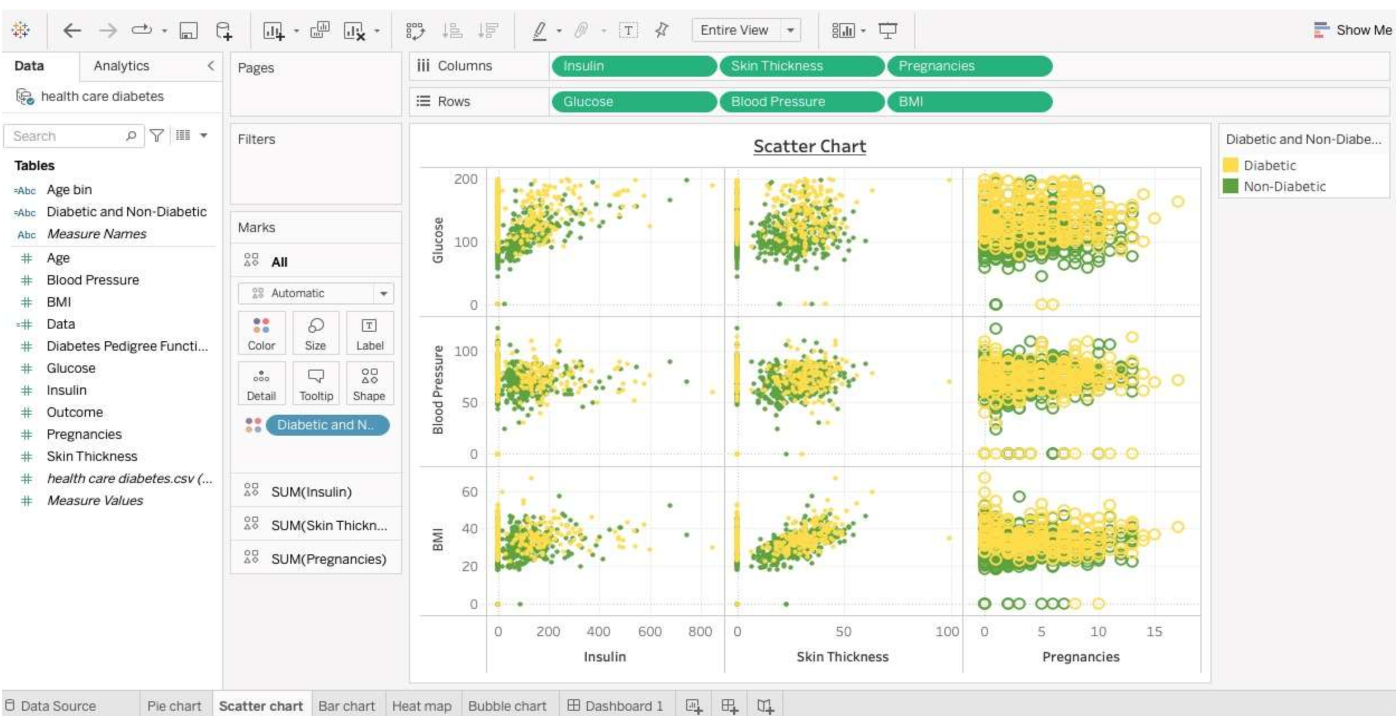
Data Reporting:

2. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

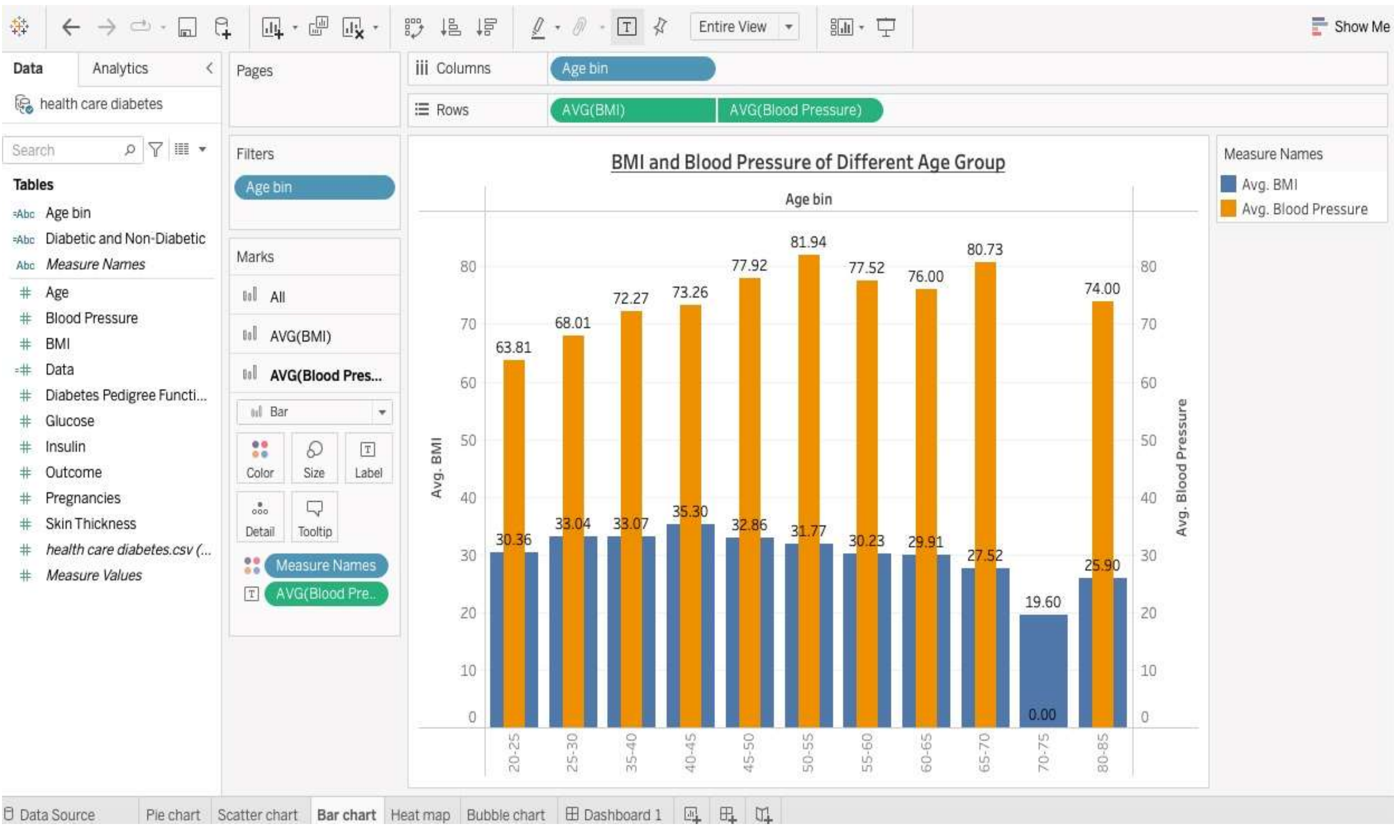
a. Pie chart to describe the diabetic or non-diabetic population



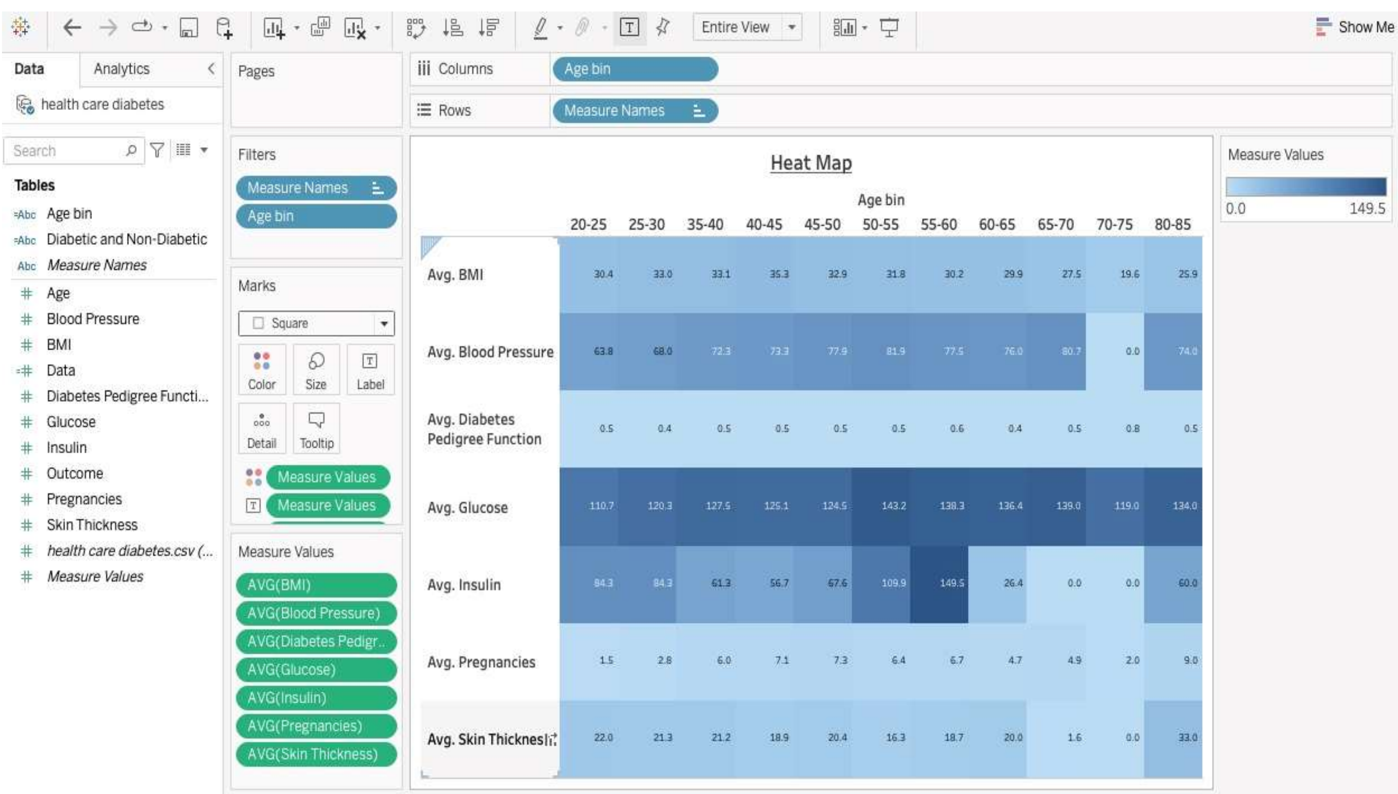
b. Scatter charts between relevant variables to analyze the relationships



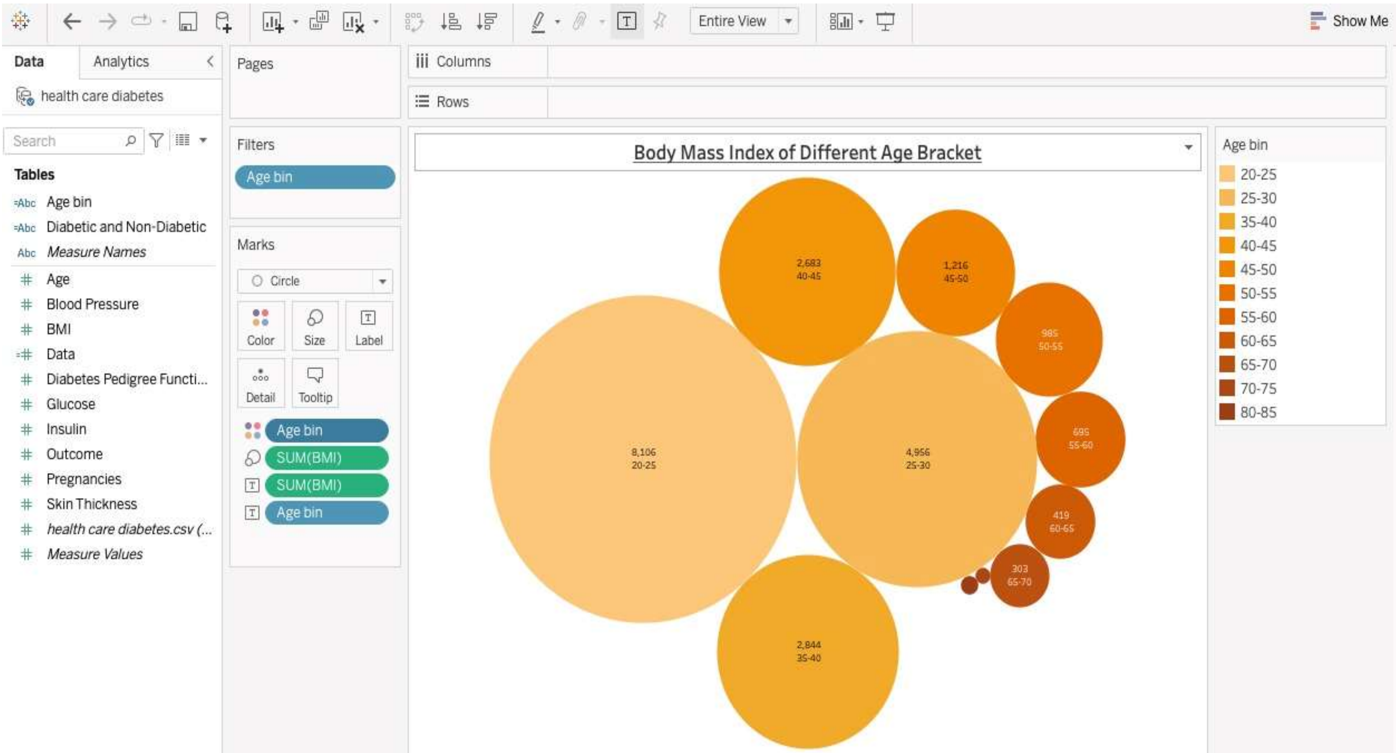
c. Histogram or frequency charts to analyze the distribution of the data

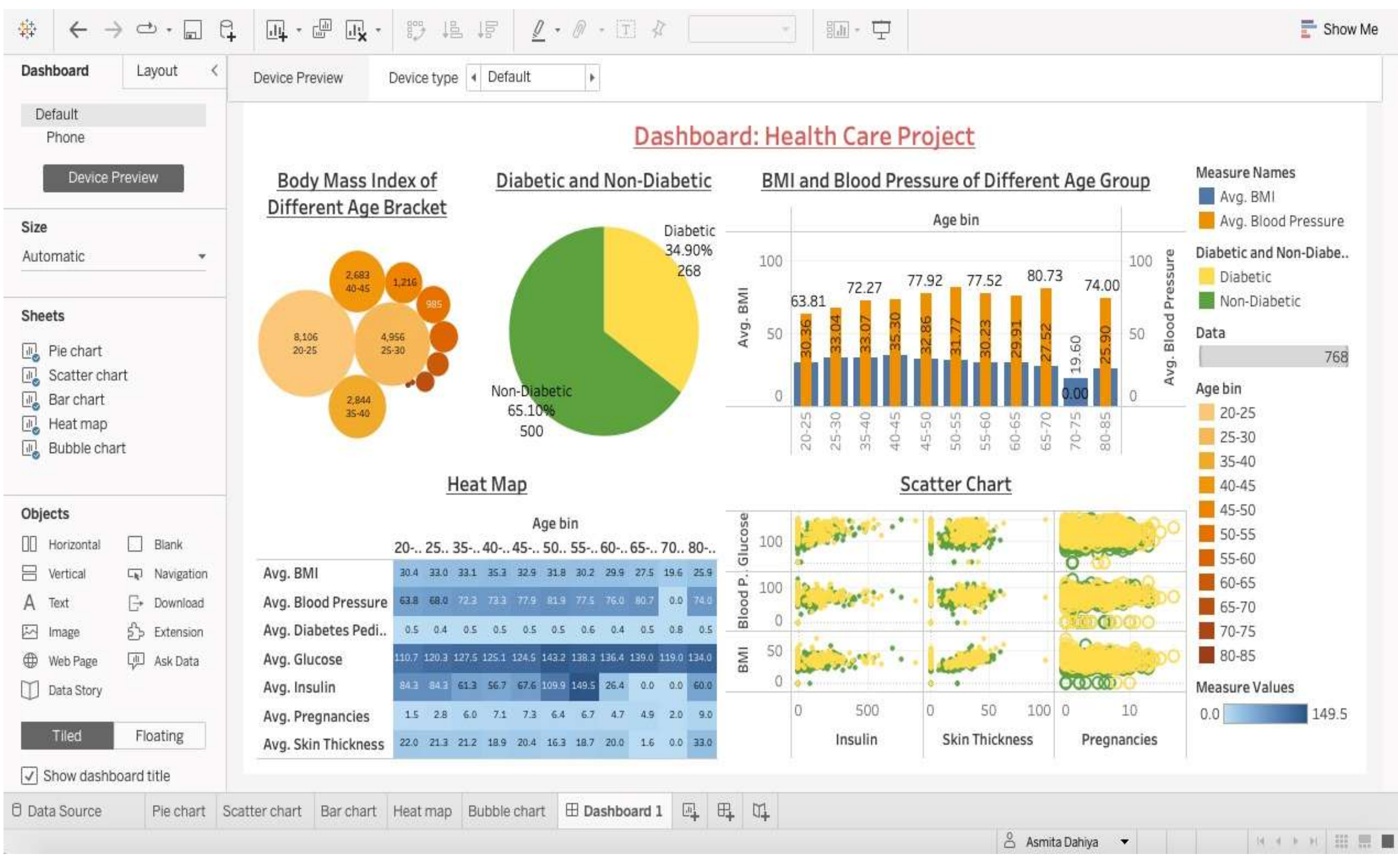


d. Heatmap of correlation analysis among the relevant variables

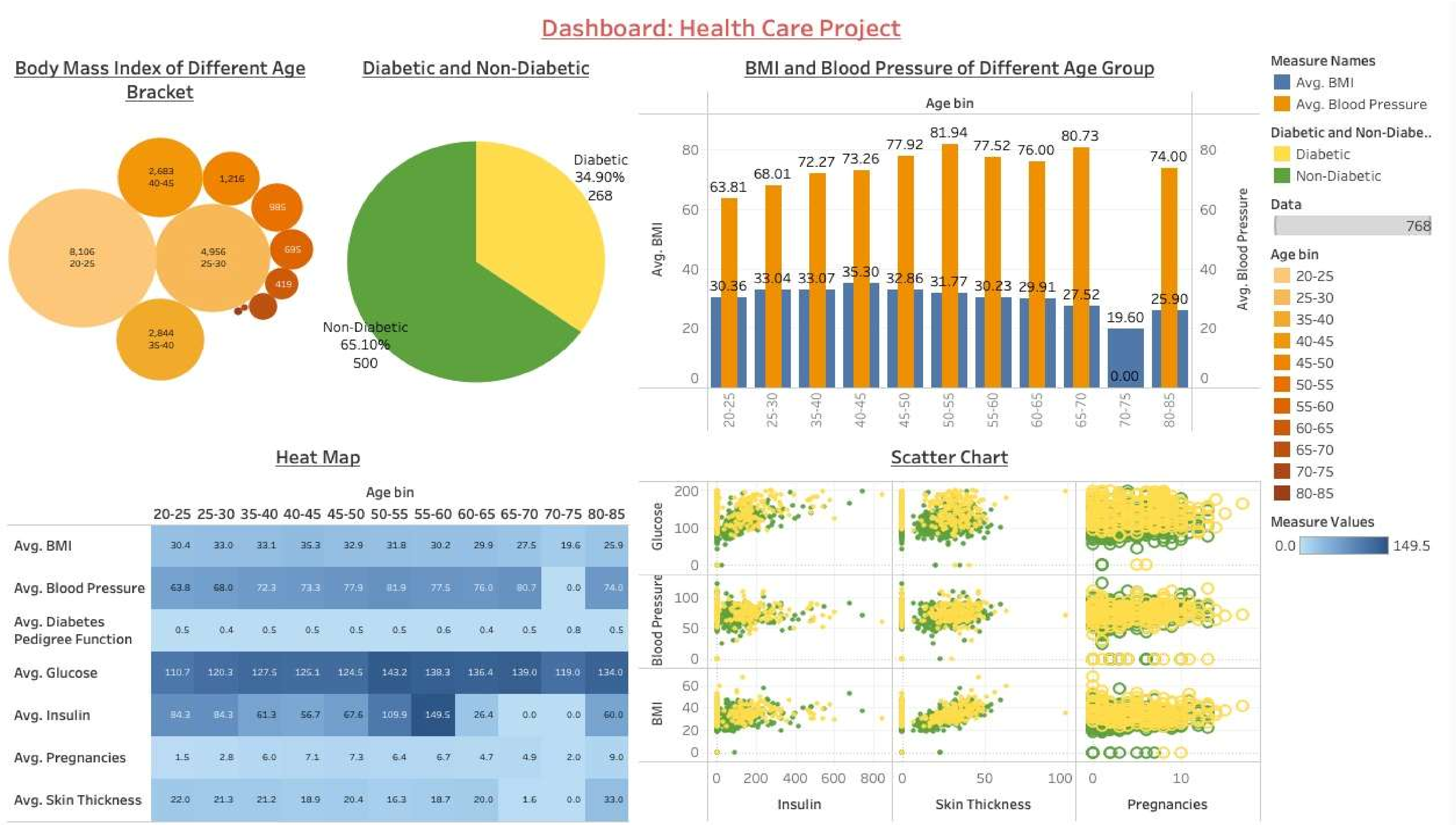


e. Create bins of these age values: 20-25, 25-30, 30-35, etc. Analyze different variables for these age brackets using a bubble chart.





DASHBOARD


Dashboard: Health Care Project
Body Mass Index of Different Age Bracket
2,683 40-45
1,216
985
695
419
8,106 20-25
4,956 25-30
2,844 35-40
Diabetic and Non-Diabetic
Diabetic 34.90% 268
Non-Diabetic 65.10% 500
BMI and Blood Pressure of Different Age Group
Age bin
Avg. BMI
Avg. Blood Pressure
63.81 68.01 72.27 73.26 77.92 81.94 77.52 76.00 80.73 74.00
30.36 33.04 33.07 35.30 32.86 31.77 30.23 29.91 27.52 19.60 0.00 25.90
20-25 25-30 35-40 40-45 45-50 50-55 55-60 60-65 65-70 70-75 80-85
Measure Names
Avg. BMI
Avg. Blood Pressure
Diabetic and Non-Diabe..
Diabetic
Non-Diabetic
Data
768
Age bin
20-25 25-30 35-40 40-45 45-50 50-55 55-60 60-65 65-70 70-75 80-85
Measure Values
0.0 149.5
Heat Map
Age bin
20-25 25-30 35-40 40-45 45-50 50-55 55-60 60-65 65-70 70-75 80-85
Avg. BMI 30.4 33.0 33.1 35.3 32.9 31.8 30.2 29.9 27.5 19.6 25.9
Avg. Blood Pressure 63.8 68.0 72.3 73.3 77.9 81.9 77.5 76.0 80.7 0.0 74.0
Avg. Diabetes Pedigree Function 0.5 0.4 0.5 0.5 0.5 0.5 0.6 0.4 0.5 0.8 0.5
Avg. Glucose 110.7 120.3 127.5 125.1 124.5 143.2 138.3 136.4 139.0 119.0 134.0
Avg. Insulin 84.3 84.3 61.3 56.7 67.6 109.9 149.5 26.4 0.0 0.0 60.0
Avg. Pregnancies 1.5 2.8 6.0 7.1 7.3 6.4 6.7 4.7 4.9 2.0 9.0
Avg. Skin Thickness 22.0 21.3 21.2 18.9 20.4 16.3 18.7 20.0 1.6 0.0 33.0
Scatter Chart
Glucose
Blood Pressure
BMI
Insulin
Skin Thickness
Pregnancies