Untitled6

October 26, 2023

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
print('all lib imported')
```

```
all lib imported
```

[5]:
```python
df=pd.read_csv('health care diabetes.csv capstone project dataset.csv')
```

[3]:
```python
df
```

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

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0
```

```
[768 rows x 9 columns]
```

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

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

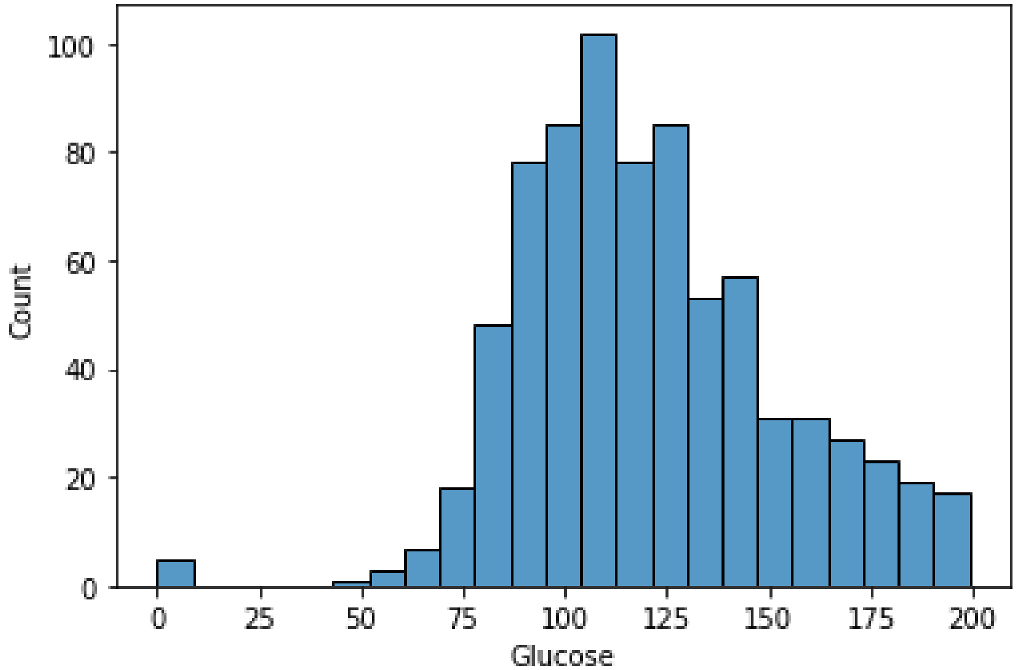
[78]:
```
df.info()
```

```
<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
```

[79]:
```
df.describe()
```

[79]:
```
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000
```

[80]:
```
# create histogram
sns.histplot(x=df['Glucose'])
plt.show()
```



[81]:
```
df['Glucose'].value_counts()
```

[81]:
```
99     17
100    17
111    14
129    14
125    14
       ..
191     1
177     1
44      1
62      1
190     1
Name: Glucose, Length: 136, dtype: int64
```

[82]:
```
df[df['Glucose']==0]
```

[82]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
75             1        0             48             20        0  24.7
182            1        0             74             20       23  27.7
342            1        0             68             35        0  32.0
```

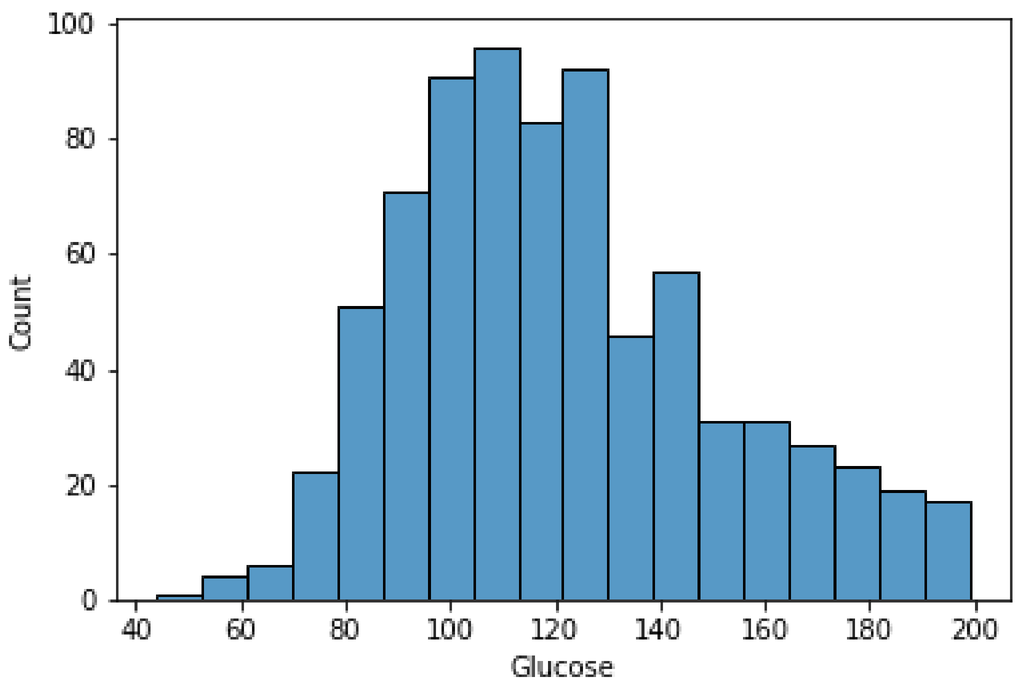
```
349              5         0             80             32         0   41.0
502              6         0             68             41         0   39.0

     DiabetesPedigreeFunction  Age  Outcome
75                      0.140   22        0
182                     0.299   21        0
342                     0.389   22        0
349                     0.346   37        1
502                     0.727   41        1
```

[8]:
```
df['Glucose'].mean()
```

[8]: 120.89453125

[85]:
```
#fill the zero value with mean of glucose
df['Glucose']=df['Glucose'].replace(0,df['Glucose'].mean())
```

[86]:
```
#create histogram
sns.histplot(df['Glucose'])
plt.show()
```



[87]:
```
df['BloodPressure'].value_counts()
```

[87]:

70	57
74	52
78	45
68	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
75	8
92	8
65	7
85	6
94	6
48	5
96	4
44	4
100	3
106	3
98	3
110	3
55	2
108	2
104	2
46	2
30	2
122	1
95	1
102	1
61	1
24	1
38	1
40	1
114	1

```
Name: BloodPressure, dtype: int64
```

[88]:

```
df[df['BloodPressure']==0]
```

[88]:

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
7             10    115.0              0              0        0  35.3
15             7    100.0              0              0        0  30.0
49             7    105.0              0              0        0   0.0
60             2     84.0              0              0        0   0.0
78             0    131.0              0              0        0  43.2
81             2     74.0              0              0        0   0.0
172            2     87.0              0             23        0  28.9
193           11    135.0              0              0        0  52.3
222            7    119.0              0              0        0  25.2
261            3    141.0              0              0        0  30.0
266            0    138.0              0              0        0  36.3
269            2    146.0              0              0        0  27.5
300            0    167.0              0              0        0  32.3
332            1    180.0              0              0        0  43.3
336            0    117.0              0              0        0  33.8
347            3    116.0              0              0        0  23.5
357           13    129.0              0             30        0  39.9
426            0     94.0              0              0        0   0.0
430            2     99.0              0              0        0  22.2
435            0    141.0              0              0        0  42.4
453            2    119.0              0              0        0  19.6
468            8    120.0              0              0        0  30.0
484            0    145.0              0              0        0  44.2
494            3     80.0              0              0        0   0.0
522            6    114.0              0              0        0   0.0
533            6     91.0              0              0        0  29.8
535            4    132.0              0              0        0  32.9
589            0     73.0              0              0        0  21.1
601            6     96.0              0              0        0  23.7
604            4    183.0              0              0        0  28.4
619            0    119.0              0              0        0  32.4
643            4     90.0              0              0        0  28.0
697            0     99.0              0              0        0  25.0
703            2    129.0              0              0        0  38.5
706           10    115.0              0              0        0   0.0

     DiabetesPedigreeFunction  Age  Outcome
7                       0.134   29        0
15                      0.484   32        1
49                      0.305   24        0
60                      0.304   21        0
78                      0.270   26        1
```

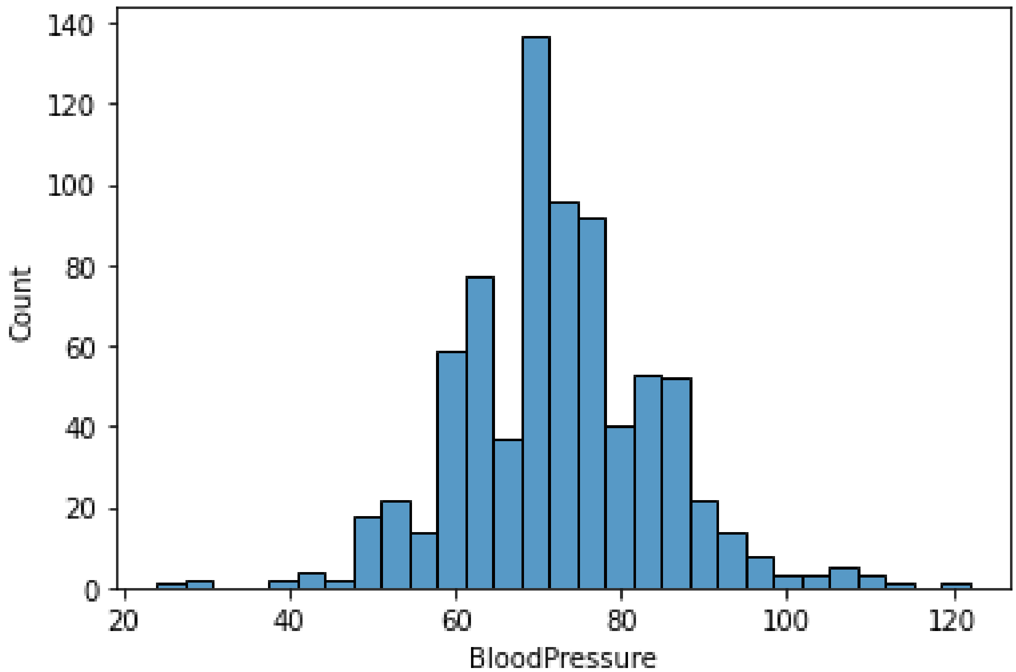
81	0.102	22	0
172	0.773	25	0
193	0.578	40	1
222	0.209	37	0
261	0.761	27	1
266	0.933	25	1
269	0.240	28	1
300	0.839	30	1
332	0.282	41	1
336	0.932	44	0
347	0.187	23	0
357	0.569	44	1
426	0.256	25	0
430	0.108	23	0
435	0.205	29	1
453	0.832	72	0
468	0.183	38	1
484	0.630	31	1
494	0.174	22	0
522	0.189	26	0
533	0.501	31	0
535	0.302	23	1
589	0.342	25	0
601	0.190	28	0
604	0.212	36	1
619	0.141	24	1
643	0.610	31	0
697	0.253	22	0
703	0.304	41	0
706	0.261	30	1

[89]:
```
df['BloodPressure'].mean()
```

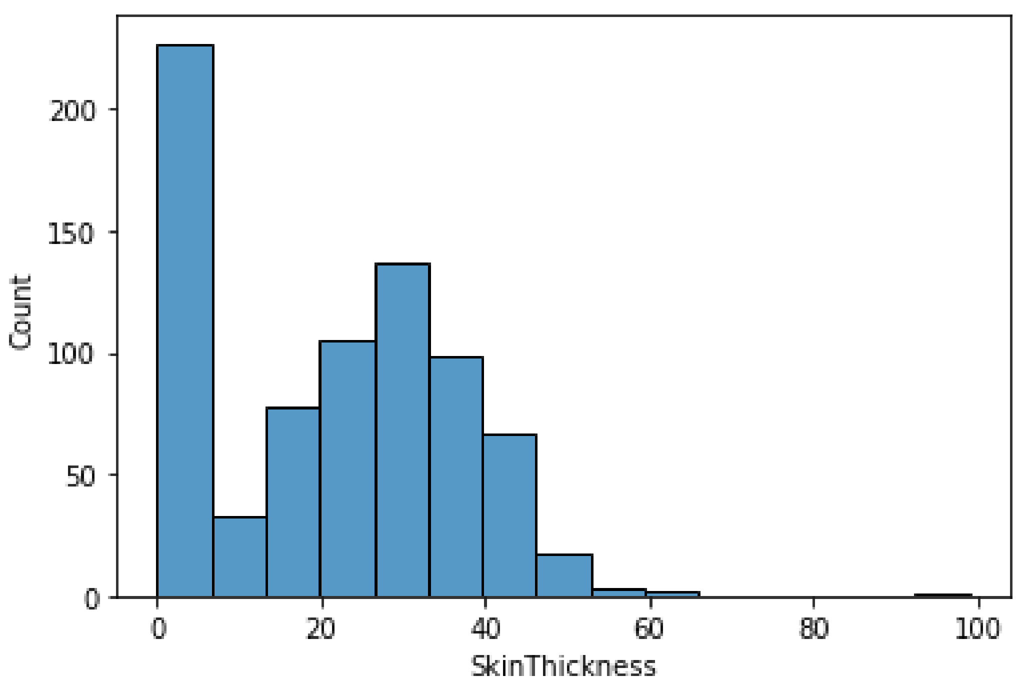
[89]: 69.10546875

[92]:
```
df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].mean())
```

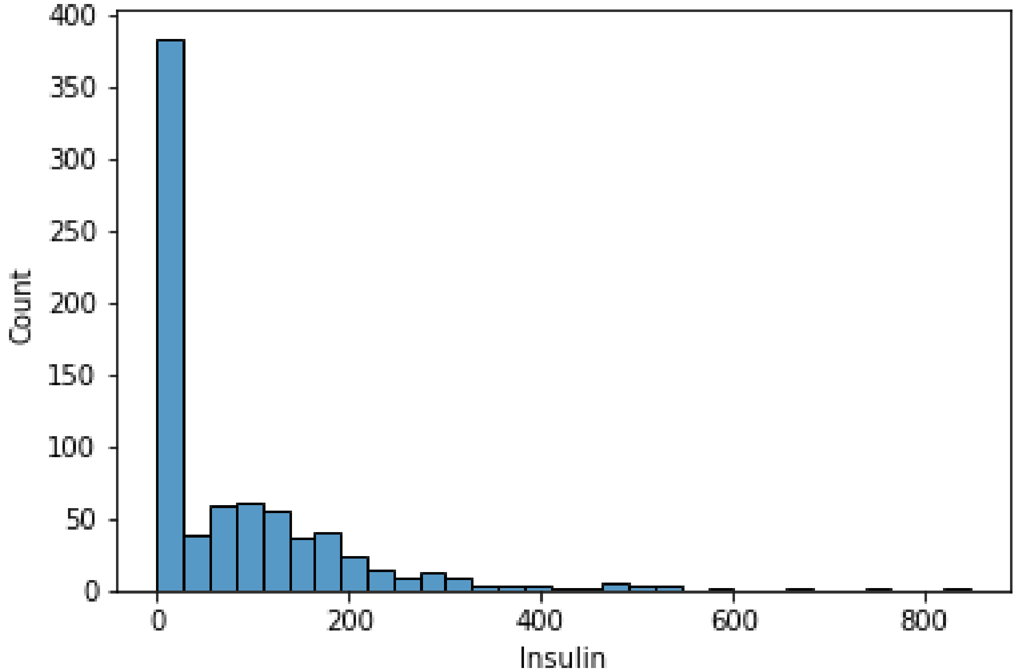
[93]:
```
sns.histplot(x=df['BloodPressure'])
plt.show()
```



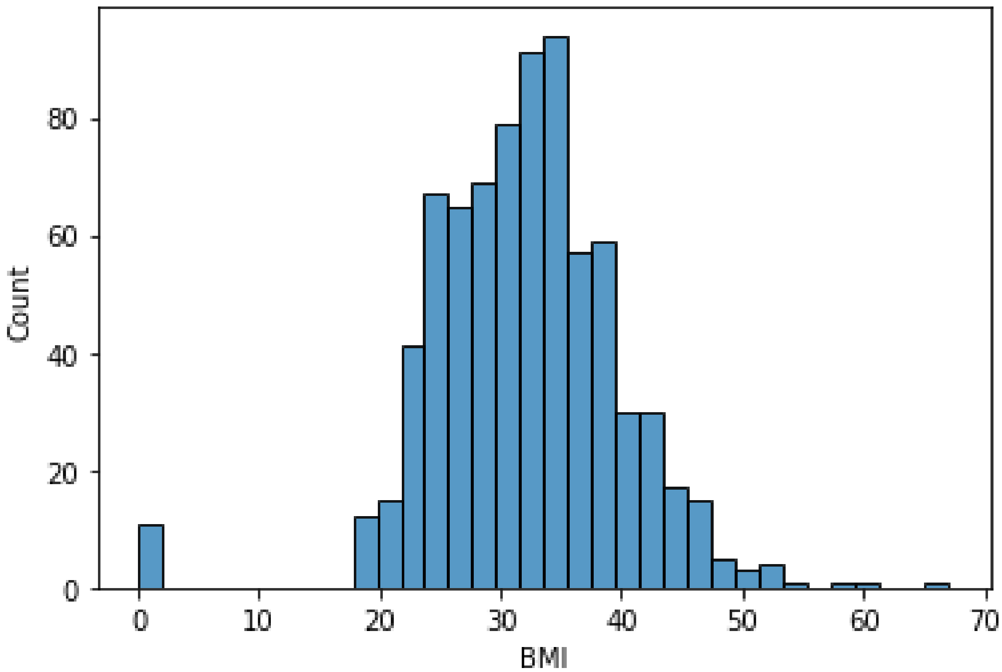
[94]:
```
sns.histplot(x=df['SkinThickness'])
plt.show()
```



[95]:
```
sns.histplot(x=df['Insulin'])
plt.show()
```



[96]:
```
sns.histplot(x=df['BMI'])
plt.show()
```



[97]:
```
df.columns
```

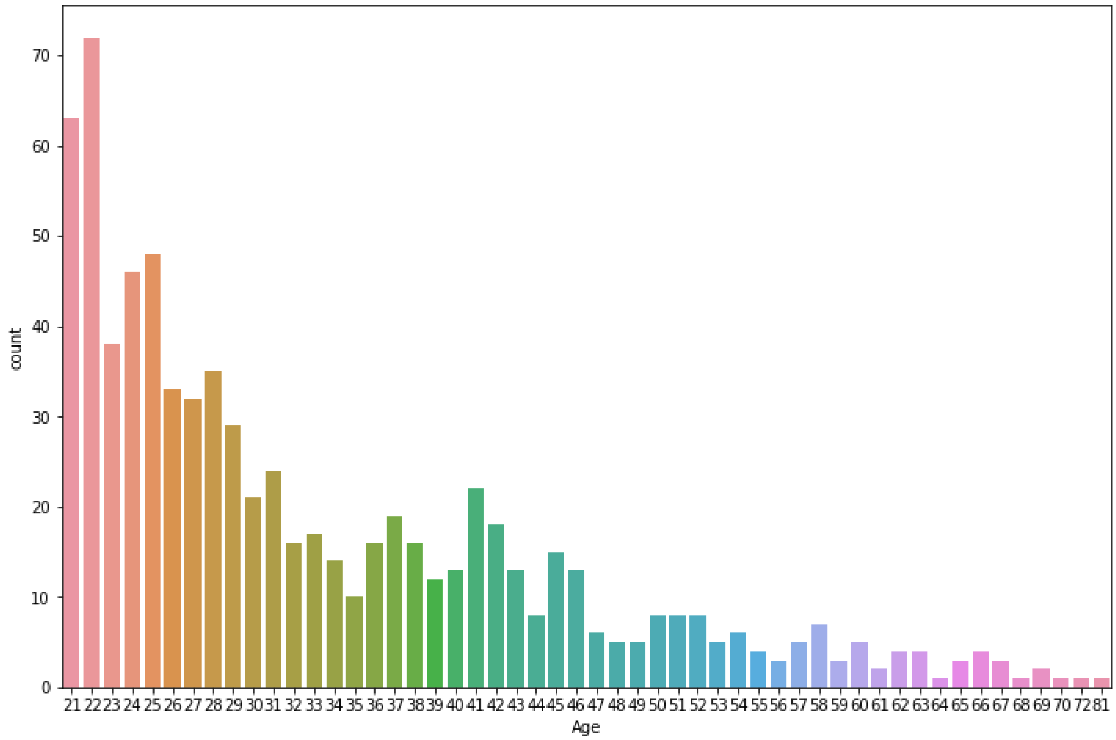
[97]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[6]:
```
variables=['SkinThickness', 'Insulin','BMI']
for i in variables:
    df[i].replace(0,df[i].median(),inplace=True)
```

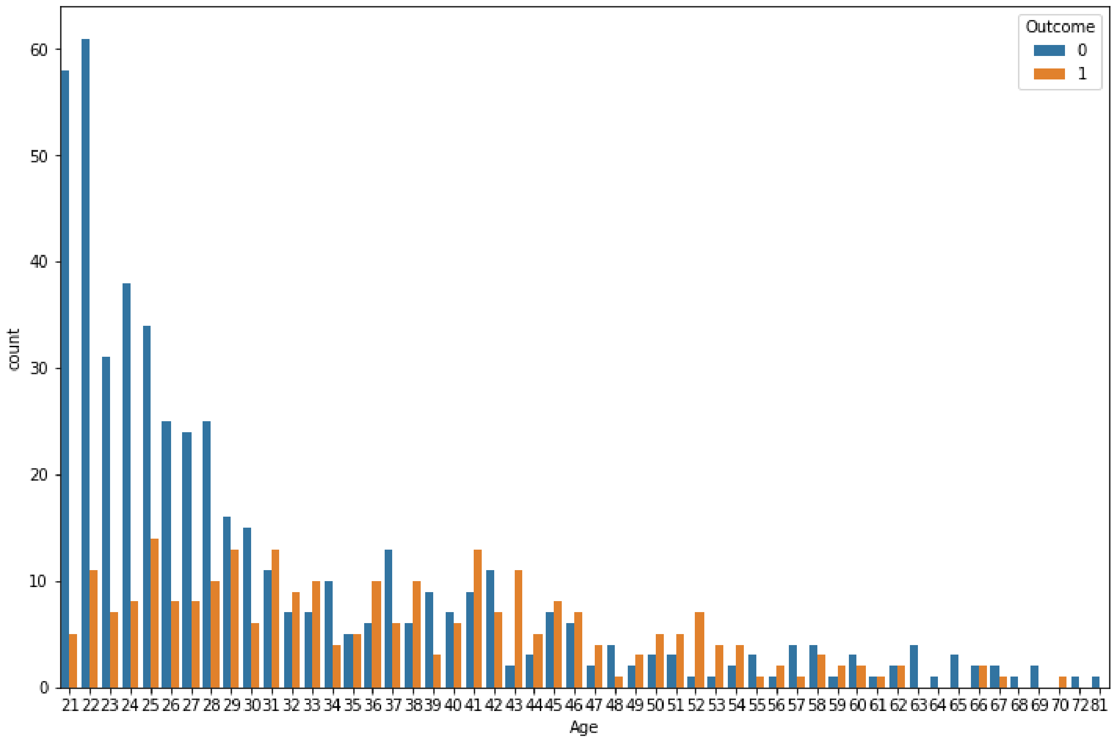
[99]:
```
df.dtypes
```

[99]:
```
Pregnancies                   int64
Glucose                     float64
BloodPressure               float64
SkinThickness                 int64
Insulin                     float64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

[100]:
```
plt.figure(figsize=(12,8))
sns.countplot(x=df['Age'])
plt.show()
```



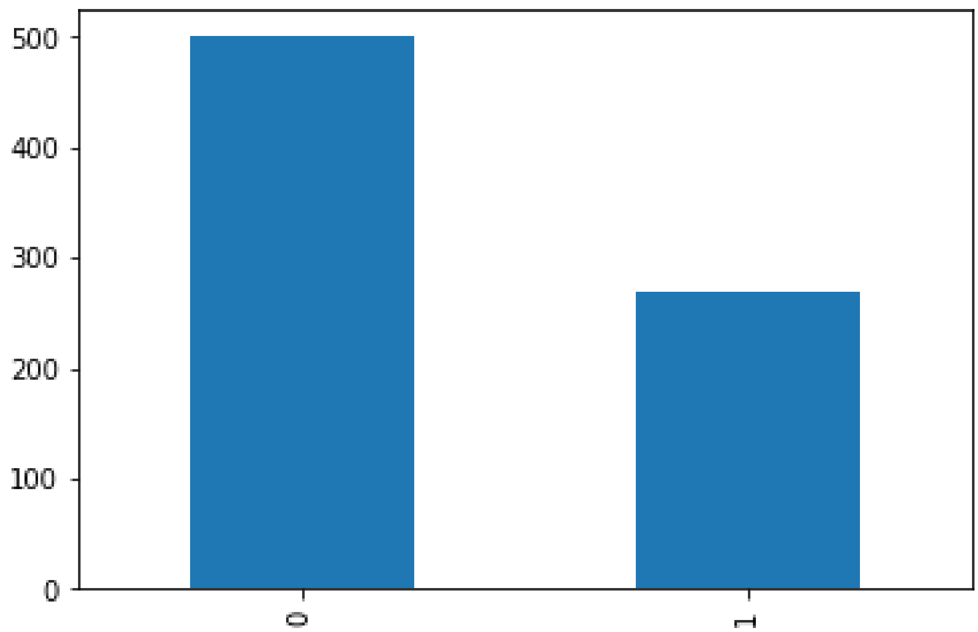
[101]:
```
plt.figure(figsize=(12,8))
sns.countplot(x=df['Age'],hue='Outcome',data=df)
plt.show()
```



[102]:
```
df['Outcome'].value_counts()
```

[102]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

[103]:
```
df['Outcome'].value_counts().plot(kind='bar')
plt.show()
```

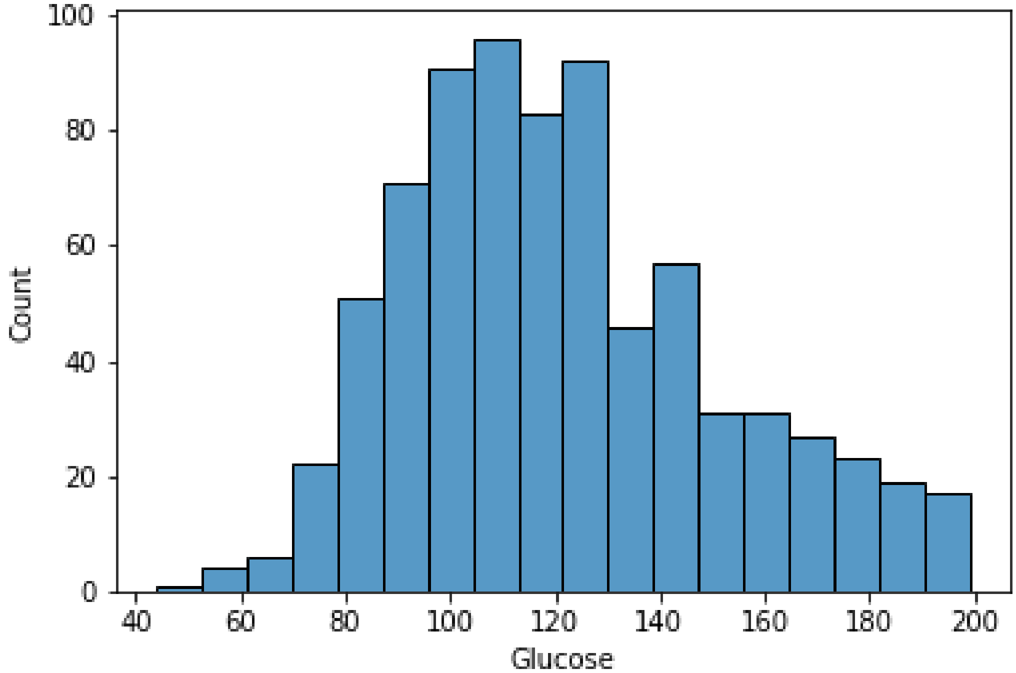


[104]:
```
df['Glucose'].value_counts()
```

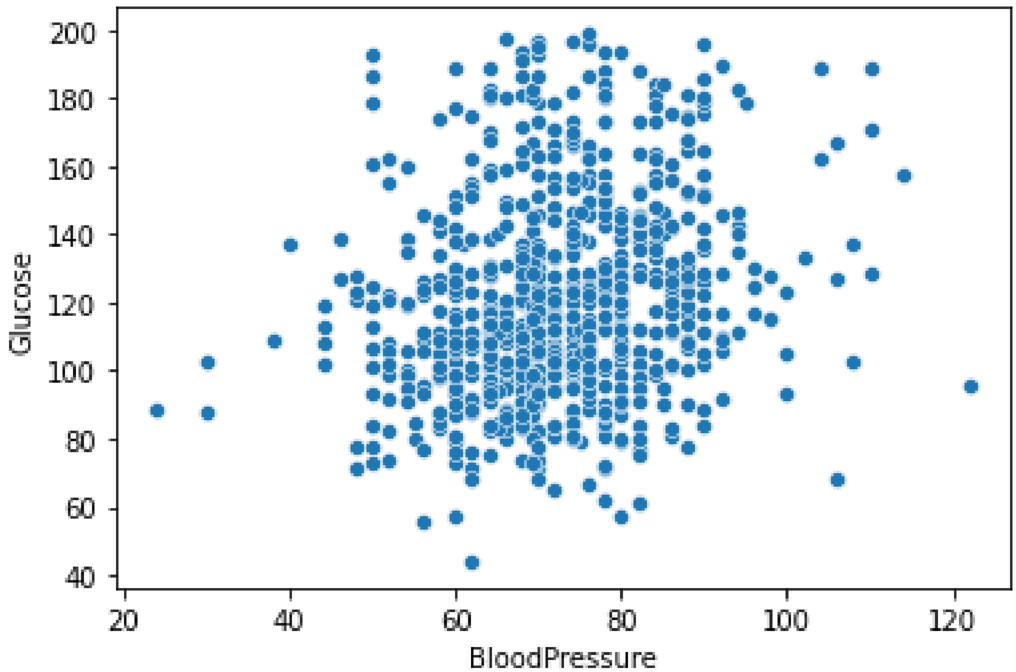
[104]:
```
99.0     17
100.0    17
111.0    14
129.0    14
125.0    14
         ..
191.0     1
177.0     1
44.0      1
62.0      1
190.0     1
Name: Glucose, Length: 136, dtype: int64
```

[105]:
```
sns.histplot(x=df['Glucose'])
```

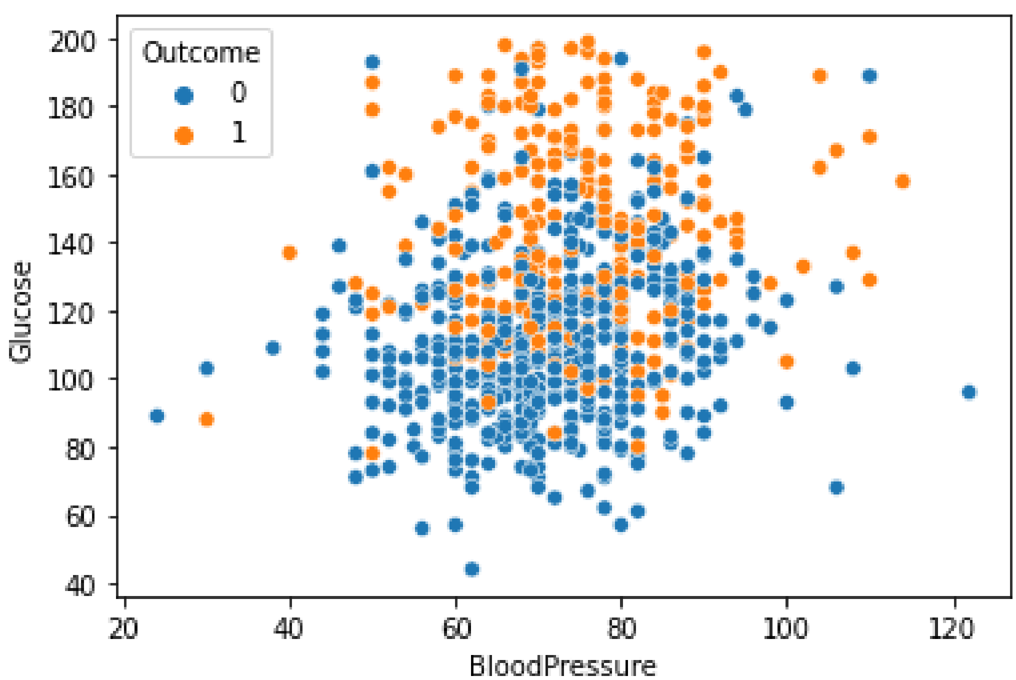
[105]:
```
<AxesSubplot: xlabel='Glucose', ylabel='Count'>
```



[106]:
```
#scatter charts created here because to understand the relationships between␣
↪the pair of variables.
#bivariate
sns.scatterplot(x=df['BloodPressure'],y=df['Glucose'])
plt.show()
```



[107]:
```
sns.scatterplot(x=df['BloodPressure'],y=df['Glucose'],hue='Outcome',data=df)
plt.show()
```



[108]:

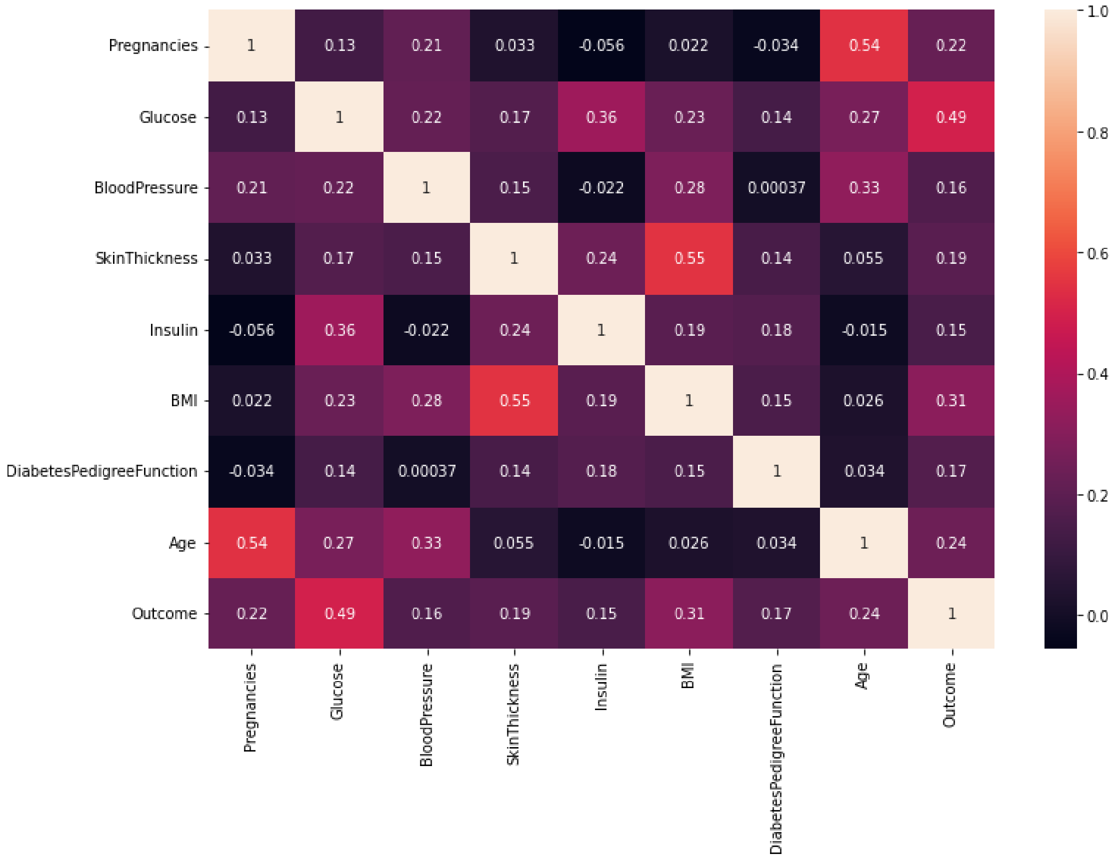
```
df.corr()
```

[108]:

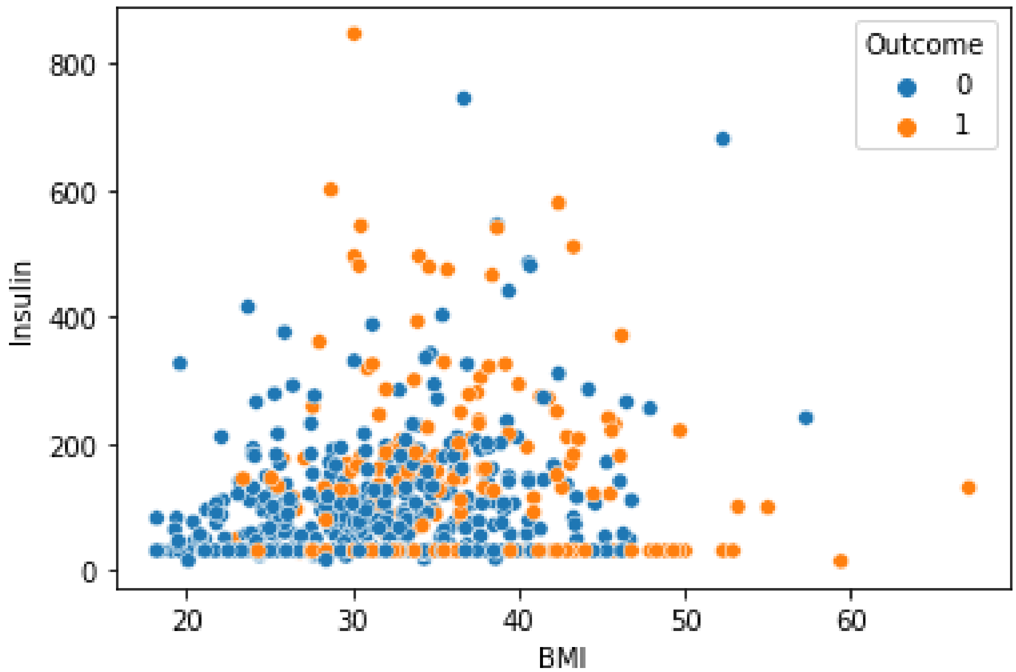
```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.127964       0.208984       0.032568
Glucose                      0.127964  1.000000       0.219666       0.172361
BloodPressure                0.208984  0.219666       1.000000       0.152458
SkinThickness                0.032568  0.172361       0.152458       1.000000
Insulin                     -0.055697  0.357081      -0.022049       0.238188
BMI                          0.021546  0.231469       0.281232       0.546951
DiabetesPedigreeFunction    -0.033523  0.137106       0.000371       0.142977
Age                          0.544341  0.266600       0.326740       0.054514
Outcome                      0.221898  0.492908       0.162986       0.189065

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.055697  0.021546                 -0.033523
Glucose                   0.357081  0.231469                  0.137106
BloodPressure            -0.022049  0.281232                  0.000371
SkinThickness             0.238188  0.546951                  0.142977
Insulin                   1.000000  0.189022                  0.178029
BMI                       0.189022  1.000000                  0.153506
DiabetesPedigreeFunction  0.178029  0.153506                  1.000000
Age                      -0.015413  0.025744                  0.033561
Outcome                   0.148457  0.312249                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.266600  0.492908
BloodPressure             0.326740  0.162986
SkinThickness             0.054514  0.189065
Insulin                  -0.015413  0.148457
BMI                       0.025744  0.312249
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

[109]:

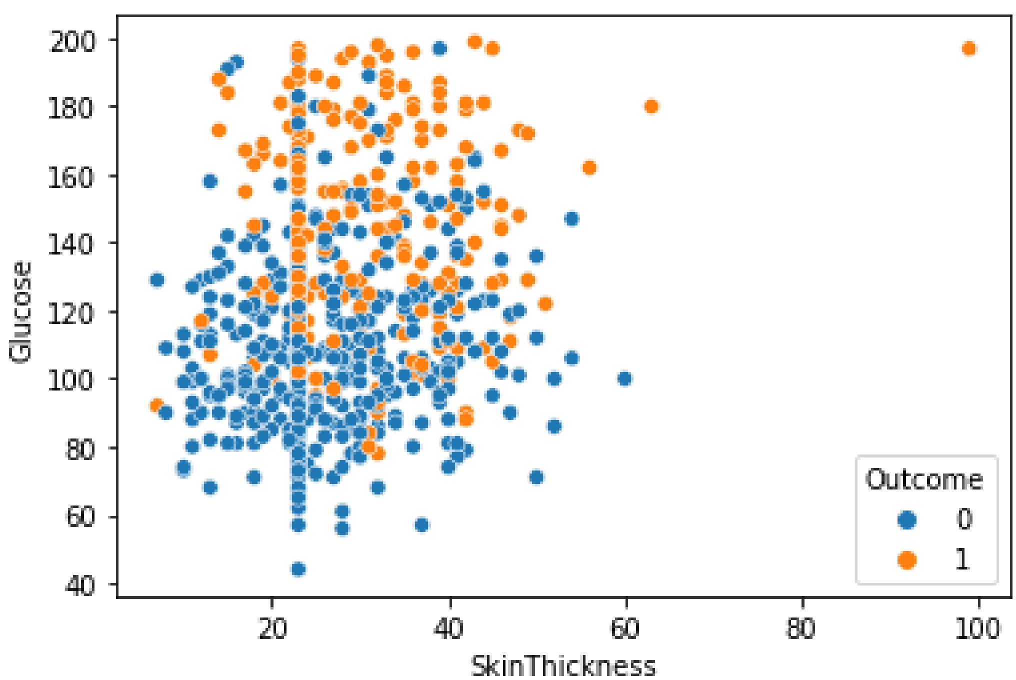
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True)
plt.show()
```



[135]:
```
sns.scatterplot(x=df['BMI'],y=df['Insulin'],hue='Outcome',data=df)
plt.show()
```



[111]:
```
sns.scatterplot(x=df['SkinThickness'],y=df['Glucose'],hue='Outcome',data=df)
plt.show()
```



[7]:

```
# Data Modeling
#create depth and indepth variables...
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

[113]:

```
x
```

[113]:

```
array([[  6.   , 148.   ,  72.   , …,  33.6  ,   0.627,  50.   ],
       [  1.   ,  85.   ,  66.   , …,  26.6  ,   0.351,  31.   ],
       [  8.   , 183.   ,  64.   , …,  23.3  ,   0.672,  32.   ],
       …,
       [  5.   , 121.   ,  72.   , …,  26.2  ,   0.245,  30.   ],
       [  1.   , 126.   ,  60.   , …,  30.1  ,   0.349,  47.   ],
       [  1.   ,  93.   ,  70.   , …,  30.4  ,   0.315,  23.   ]])
```

[114]:

```
y
```

[114]:

```
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       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, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
```

```
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

[8]:
```
#create split data
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_state=42)
```

[9]:
```
x_train.shape
```

[9]: (614, 8)

[138]:
```
x_test.shape
```

[138]: (154, 8)

[12]:
```
#Apply logistic Regression
from sklearn.linear_model import LogisticRegression
log_reg=LogisticRegression()
```

[13]:
```
log_reg.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/site-packages/sklearn/linear_model/_logistic.py:460:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[13]: LogisticRegression()

[14]:
```
y_pred=log_reg.predict(x_test)
```

[15]:
```
#Evaluate the model1
from sklearn.metrics import␣
↪confusion_matrix,accuracy_score,classification_report
```

[16]:
```
confusion_matrix(y_test,y_pred)
```

[16]:
```
array([[82, 17],
       [17, 38]])
```

[17]:
```
#print Accuracy
print('Accuracy_score=',accuracy_score(y_test,y_pred))
```

```
Accuracy_score= 0.7792207792207793
```

[18]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.83      0.83        99
           1       0.69      0.69      0.69        55

    accuracy                           0.78       154
   macro avg       0.76      0.76      0.76       154
weighted avg       0.78      0.78      0.78       154
```

[19]:
```
from sklearn.metrics import roc_auc_score,roc_curve
prob=log_reg.predict_proba(x)
```

[20]:
```
prob
```

[20]:
```
array([[0.21669737, 0.78330263],
       [0.95197413, 0.04802587],
       [0.28517013, 0.71482987],
       …,
       [0.88484086, 0.11515914],
       [0.61514118, 0.38485882],
       [0.93599551, 0.06400449]])
```

[21]:
```
prob=prob[:,1]
prob
```

[21]:
```
array([0.78330263, 0.04802587, 0.71482987, 0.03075186, 0.97301334,
       0.09946202, 0.05829103, 0.57507876, 0.69890241, 0.3949515 ,
       0.19744866, 0.87238633, 0.8394048 , 0.62130257, 0.64410435,
       0.40120915, 0.4763238 , 0.13762706, 0.48602766, 0.28041873,
       0.39209429, 0.33951694, 0.94320208, 0.17960227, 0.65046505,
       0.33813868, 0.75190338, 0.03301445, 0.40100067, 0.26711034,
       0.58504649, 0.54944975, 0.03271877, 0.01717337, 0.36465334,
       0.1404747 , 0.55460954, 0.40222962, 0.20364543, 0.77173009,
       0.71624203, 0.7222478 , 0.08473635, 0.94064195, 0.57047882,
       0.98137169, 0.46791957, 0.03406521, 0.35580707, 0.36797164,
       0.02345093, 0.08164041, 0.04870012, 0.82757015, 0.69302601,
```

```
0.01627194, 0.85968384, 0.54394706, 0.94552571, 0.20085351,
0.16041524, 0.46532547, 0.02158147, 0.27561148, 0.33939674,
0.08989962, 0.27976197, 0.69915019, 0.02302847, 0.21574614,
0.22594116, 0.27844481, 0.79158201, 0.1607359 , 0.04746864,
0.00150222, 0.07178552, 0.20468287, 0.78714334, 0.07470387,
0.06458997, 0.09591164, 0.12338584, 0.03937939, 0.71541476,
0.19706682, 0.44662208, 0.19963743, 0.66870044, 0.05692222,
0.01269436, 0.22743575, 0.39908849, 0.35959147, 0.2055333 ,
0.46369509, 0.06849049, 0.01240793, 0.09268621, 0.54117379,
0.91032002, 0.22584123, 0.04102098, 0.02584384, 0.31481118,
0.23866763, 0.01191253, 0.40164028, 0.10587822, 0.09999022,
0.52452956, 0.64041557, 0.04699989, 0.08352966, 0.71659669,
0.67893832, 0.34806951, 0.15410237, 0.08769909, 0.02850645,
0.94209079, 0.22027987, 0.13366918, 0.45708458, 0.1326785 ,
0.7148407 , 0.50888202, 0.18701774, 0.23379229, 0.30544059,
0.5714211 , 0.68821471, 0.64503964, 0.27320609, 0.0441359 ,
0.22784055, 0.1009822 , 0.066354  , 0.30501288, 0.11989486,
0.18917211, 0.34311386, 0.15885214, 0.32307715, 0.34696055,
0.11268926, 0.04687542, 0.41652881, 0.74587792, 0.03000017,
0.34915174, 0.11717124, 0.85280985, 0.48526673, 0.96683836,
0.89985335, 0.07026849, 0.11728124, 0.03683821, 0.96191383,
0.38782366, 0.3107189 , 0.23612418, 0.08456216, 0.33659391,
0.2279767 , 0.38197242, 0.33115438, 0.22013841, 0.10010638,
0.12163325, 0.47470061, 0.24706437, 0.26985263, 0.05469434,
0.8547314 , 0.12450306, 0.88165721, 0.8223703 , 0.7326887 ,
0.02594875, 0.30921456, 0.00165237, 0.03427543, 0.32679567,
0.95624612, 0.82803449, 0.48948821, 0.17432299, 0.26299444,
0.0455168 , 0.42934795, 0.63854236, 0.9827361 , 0.0706379 ,
0.63793418, 0.04599449, 0.08360387, 0.36693034, 0.26836571,
0.18502051, 0.46470336, 0.17127787, 0.02662948, 0.40531774,
0.08649446, 0.95810934, 0.73756759, 0.08386423, 0.86855726,
0.04050918, 0.61947738, 0.86033624, 0.59128506, 0.22215566,
0.87013987, 0.27144104, 0.2763039 , 0.19657175, 0.42593677,
0.70039475, 0.84452621, 0.38082649, 0.71731093, 0.06593232,
0.05277611, 0.11763551, 0.8083686 , 0.96733044, 0.33536058,
0.70682456, 0.67058579, 0.02617467, 0.34766683, 0.03390157,
0.8813836 , 0.89574695, 0.88038205, 0.7405312 , 0.03401891,
0.04535485, 0.09484996, 0.23349014, 0.46981637, 0.47916485,
0.94178067, 0.38160269, 0.67379574, 0.26550591, 0.0672056 ,
0.35167668, 0.14109662, 0.02314194, 0.08708125, 0.22774721,
0.22001694, 0.24435775, 0.09014097, 0.58816702, 0.92131511,
0.74434889, 0.73177576, 0.14111046, 0.62075913, 0.30058498,
0.38215541, 0.83588039, 0.72030897, 0.04131552, 0.5386443 ,
0.78042881, 0.04950891, 0.1165917 , 0.03995048, 0.45944497,
0.34195746, 0.11902796, 0.07174108, 0.35530124, 0.08374889,
0.54250281, 0.48423856, 0.28973036, 0.59989724, 0.15686028,
0.45752521, 0.54155391, 0.55453949, 0.03884774, 0.25234549,
```

0.05990924, 0.30865534, 0.76537408, 0.55269231, 0.61341058,
0.70006066, 0.18283975, 0.13293154, 0.37641838, 0.38943882,
0.90306255, 0.37622794, 0.08486455, 0.6087116 , 0.1938848 ,
0.31509282, 0.57823723, 0.08170618, 0.43448705, 0.36038568,
0.06832688, 0.24603374, 0.30169845, 0.21463217, 0.62891655,
0.17533331, 0.02369194, 0.6708649 , 0.2486017 , 0.77530845,
0.21245026, 0.12634533, 0.16428558, 0.64212244, 0.14887692,
0.1695345 , 0.37473279, 0.84383456, 0.21683268, 0.14574318,
0.52955521, 0.07090384, 0.97481163, 0.11185516, 0.03252615,
0.78074755, 0.78886835, 0.28798447, 0.72689408, 0.89413904,
0.126062 , 0.08816954, 0.00359416, 0.28613485, 0.4900384 ,
0.58292759, 0.36027042, 0.1740071 , 0.03629472, 0.01664225,
0.2204286 , 0.27260651, 0.06435568, 0.06385464, 0.24034538,
0.70703781, 0.47914068, 0.93457314, 0.31589777, 0.88503521,
0.73823043, 0.7419924 , 0.45379797, 0.8842874 , 0.39966616,
0.23709192, 0.19581474, 0.02572228, 0.02270205, 0.2507033 ,
0.93189557, 0.53808467, 0.10198623, 0.16130824, 0.49800558,
0.79524968, 0.02711064, 0.14374151, 0.86631409, 0.40096273,
0.1732779 , 0.02524299, 0.10800667, 0.12291532, 0.07481243,
0.06698794, 0.35520047, 0.45438226, 0.55696375, 0.22729112,
0.16050288, 0.87053778, 0.05392314, 0.10434445, 0.69950317,
0.43875402, 0.16831841, 0.25423623, 0.02035405, 0.80177714,
0.10985714, 0.38011268, 0.407979 , 0.07726639, 0.71986109,
0.56397281, 0.26193754, 0.03852141, 0.92485503, 0.74536515,
0.24357333, 0.17669021, 0.64508919, 0.14939804, 0.36135144,
0.38213396, 0.13410298, 0.69811804, 0.02083901, 0.14045542,
0.43403477, 0.05067137, 0.26218364, 0.1594737 , 0.80256321,
0.72903494, 0.20889094, 0.76270388, 0.34238275, 0.12077751,
0.07857036, 0.11447178, 0.04299747, 0.15886438, 0.13399454,
0.83004618, 0.66320473, 0.40155636, 0.01344178, 0.34363317,
0.78456906, 0.07274051, 0.17683959, 0.36965464, 0.21105445,
0.99861792, 0.07374774, 0.1088891 , 0.15343785, 0.13217694,
0.01686346, 0.25329327, 0.13139552, 0.6821291 , 0.24427694,
0.85788147, 0.64129819, 0.063745 , 0.86936699, 0.71054357,
0.26510123, 0.01677697, 0.21705399, 0.06542886, 0.21617549,
0.07610811, 0.0280366 , 0.19400366, 0.53984341, 0.84250849,
0.67998389, 0.2490068 , 0.28605042, 0.39717084, 0.10908288,
0.35820234, 0.17575847, 0.10078122, 0.19188545, 0.48064723,
0.53172722, 0.20207661, 0.05858843, 0.0647947 , 0.93040968,
0.44925233, 0.38625282, 0.97685983, 0.05874444, 0.92364697,
0.1464516 , 0.11134259, 0.13657793, 0.53971592, 0.13219251,
0.76765851, 0.10557028, 0.05450483, 0.78994289, 0.61953438,
0.0506314 , 0.11792312, 0.03371528, 0.32178042, 0.22058151,
0.09780219, 0.72303264, 0.21364686, 0.13154048, 0.38189952,
0.15698393, 0.06936181, 0.1337415 , 0.06388733, 0.04862019,
0.48348392, 0.66934755, 0.58097003, 0.15383146, 0.15617397,
0.01341394, 0.18923678, 0.40935713, 0.67828907, 0.21062703,

```
0.02843519, 0.01655311, 0.05946832, 0.12832783, 0.12582042,
0.23312936, 0.47524526, 0.36099041, 0.30902493, 0.19888525,
0.57414131, 0.11145134, 0.06817841, 0.32255061, 0.50006244,
0.44050972, 0.26060659, 0.47501319, 0.08801539, 0.06799559,
0.78997319, 0.98116097, 0.23332705, 0.7010114 , 0.71796678,
0.0784328 , 0.08404462, 0.31108722, 0.06457504, 0.12603751,
0.12728952, 0.18175407, 0.28248629, 0.63147807, 0.107795  ,
0.46663487, 0.89767783, 0.12335863, 0.12728537, 0.09781482,
0.08456347, 0.18592489, 0.14410365, 0.50448909, 0.21214967,
0.08835803, 0.05468047, 0.1476116 , 0.0993072 , 0.24913833,
0.32145161, 0.21678343, 0.48553972, 0.31193703, 0.96930669,
0.58026596, 0.08972292, 0.41301938, 0.33566535, 0.23826293,
0.03572503, 0.60462556, 0.07010878, 0.91170442, 0.03898747,
0.86076435, 0.19581746, 0.49010578, 0.24208001, 0.3911346 ,
0.65605071, 0.2068757 , 0.11088884, 0.73167601, 0.08326124,
0.06149596, 0.13276898, 0.1195093 , 0.81934739, 0.87158104,
0.35301856, 0.8883053 , 0.02735419, 0.4953253 , 0.03677046,
0.11135546, 0.77543539, 0.82205883, 0.26028245, 0.75086301,
0.06265208, 0.10008469, 0.00958805, 0.56824065, 0.35173615,
0.18553612, 0.1949548 , 0.979958  , 0.19676068, 0.09403409,
0.13974829, 0.0765086 , 0.21893713, 0.40971843, 0.03282449,
0.28834777, 0.09418456, 0.09156802, 0.06432848, 0.07341488,
0.30082035, 0.16210857, 0.09944884, 0.4666542 , 0.02013259,
0.08750859, 0.29974252, 0.49843659, 0.26781396, 0.10654846,
0.42938247, 0.33075092, 0.79373373, 0.33181344, 0.0571485 ,
0.03387383, 0.24914223, 0.26501938, 0.1817275 , 0.09142465,
0.41882151, 0.02891234, 0.64131654, 0.56759885, 0.20023186,
0.64782409, 0.97855098, 0.65179695, 0.75998522, 0.3692774 ,
0.12245704, 0.72009336, 0.17980617, 0.26023778, 0.56364541,
0.83199127, 0.06653953, 0.09131197, 0.79632125, 0.51540377,
0.81655321, 0.5013373 , 0.11802849, 0.30085507, 0.04829117,
0.01162689, 0.8777894 , 0.26311295, 0.2841356 , 0.76111203,
0.25903038, 0.11058535, 0.11113884, 0.18687467, 0.78616359,
0.1790896 , 0.82231903, 0.50043221, 0.65062435, 0.03012703,
0.18471599, 0.48714972, 0.11066679, 0.28744759, 0.7079578 ,
0.2469424 , 0.3907304 , 0.88499324, 0.80389947, 0.07187447,
0.12544609, 0.53320702, 0.18687258, 0.71808053, 0.19363138,
0.28715379, 0.32454781, 0.76657939, 0.10237775, 0.08453648,
0.88159795, 0.79430896, 0.19023064, 0.18155322, 0.33941352,
0.0529662 , 0.18930324, 0.4304263 , 0.39176095, 0.18907439,
0.38125608, 0.20399336, 0.27619553, 0.27000284, 0.06406877,
0.2049692 , 0.11867375, 0.8610195 , 0.0929872 , 0.15759812,
0.17462356, 0.0986799 , 0.11486468, 0.15155489, 0.32687244,
0.79261565, 0.15488316, 0.07450818, 0.64989428, 0.92283892,
0.23344183, 0.7682444 , 0.51174801, 0.83503306, 0.55596606,
0.55125631, 0.30690151, 0.07651115, 0.62083706, 0.71348855,
0.57984924, 0.43462423, 0.48616385, 0.17076424, 0.93778541,
```

```
0.09766349, 0.94277867, 0.0453642 , 0.32827541, 0.33090686,
0.11515914, 0.38485882, 0.06400449])
```

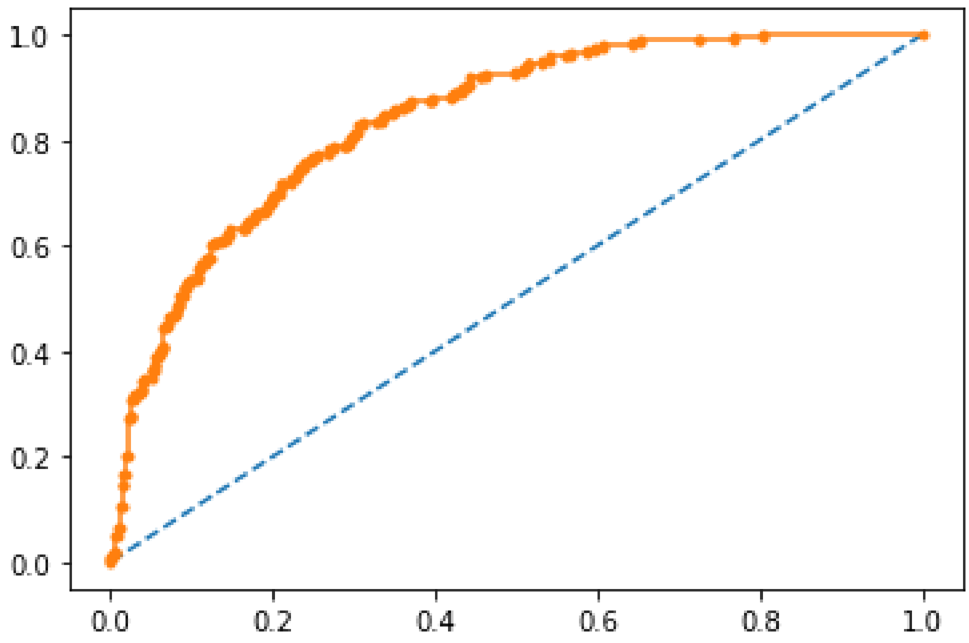
[153]:
```python
#calculate roc_auc_score
auc=roc_auc_score(y,prob)
print('AUC Score is',auc)
```

```
AUC Score is 0.8410671641791044
```

[72]:
```python
y.shape
```

[72]:
```
(768,)
```

[154]:
```python
fpr,tpr,thresholds=roc_curve(y,prob)

plt.plot([0,1],[0,1],linestyle='--')
plt.plot(fpr,tpr,marker='.')
plt.show()
```



[159]:
```python
#APPLY DECISION TREE CLASSIFIER
from sklearn.tree import DecisionTreeClassifier
decision_tree=DecisionTreeClassifier()
```

[160]:
```python
decision_tree.fit(x_train,y_train)
```

[160]:
```
DecisionTreeClassifier()
```

[161]:
```
y_pred=decision_tree.predict(x_test)
```

[167]:
```
print('Accuracy score:',accuracy_score(y_test,y_pred))
```

```
Accuracy score: 0.7012987012987013
```

[22]:
```
from sklearn.neighbors import KNeighborsClassifier
clf=KNeighborsClassifier()
```

[23]:
```
clf.fit(x_train,y_train)
```

[23]:
```
KNeighborsClassifier()
```

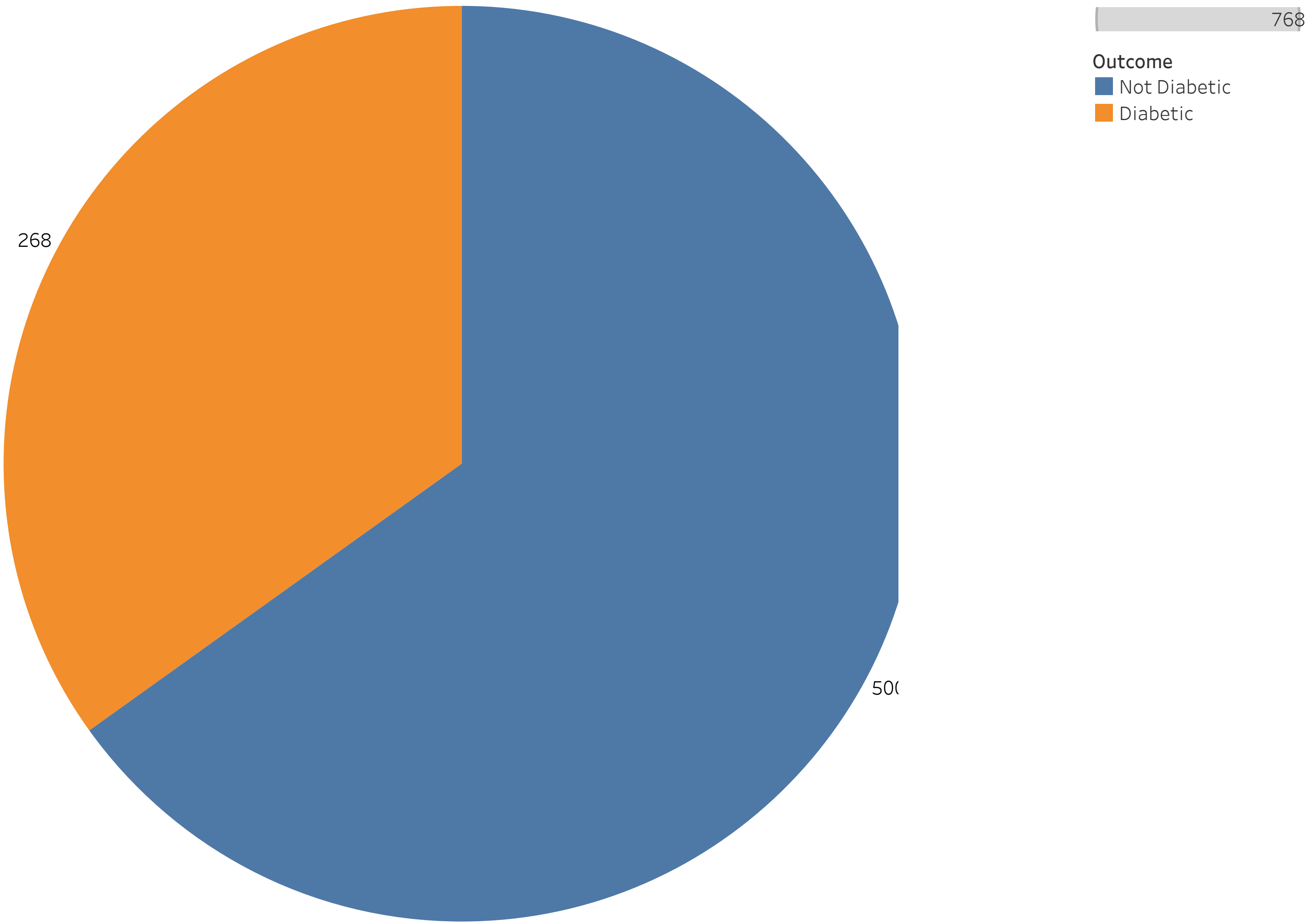
[24]:
```
y_pred=clf.predict(x_test)
```

[25]:
```
print('Accuracy score:',accuracy_score(y_test,y_pred))
```

```
Accuracy score: 0.6753246753246753
```

[ ]:
```

```

Proportion

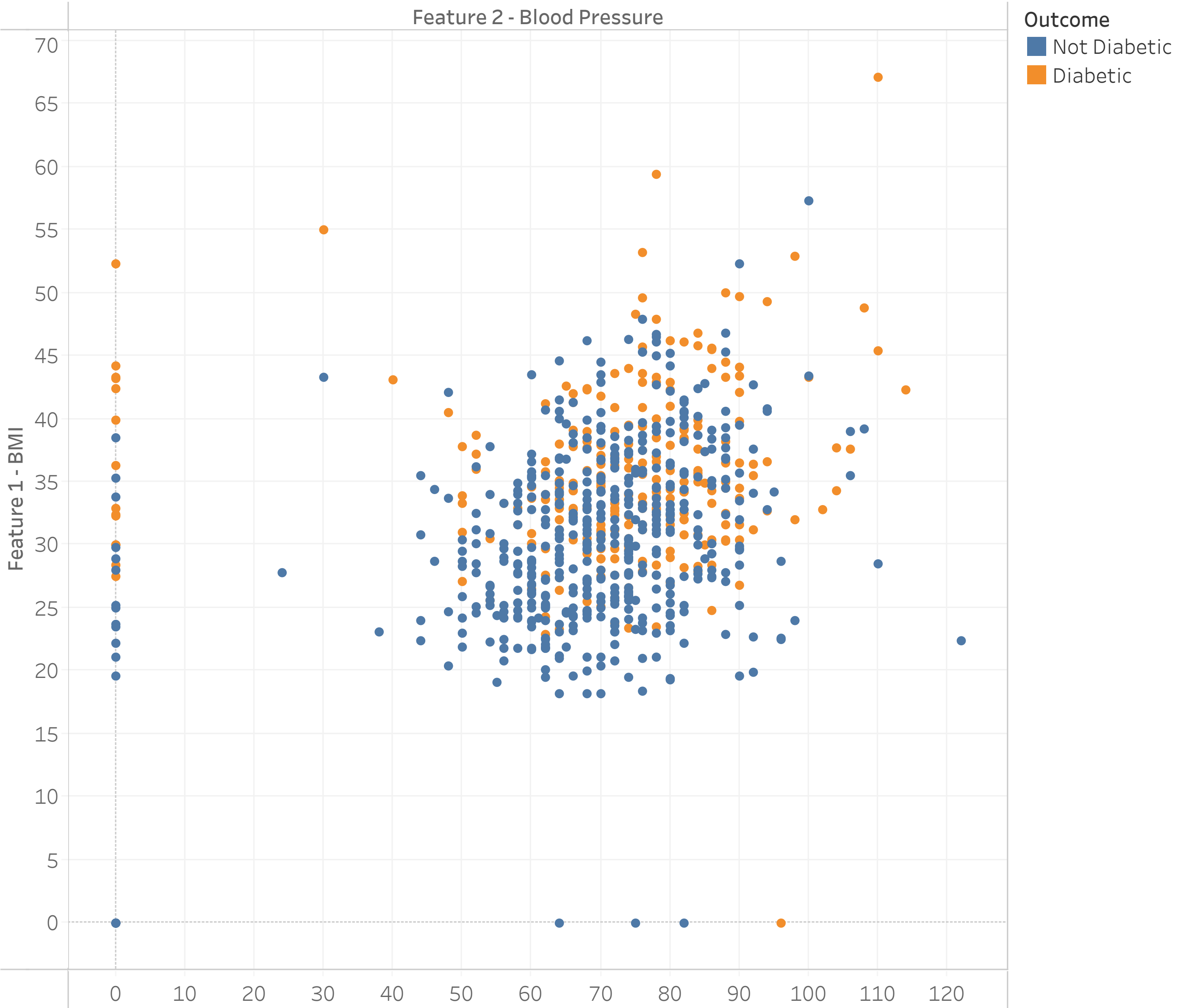


Outcome (color) and count of Outcome (size). The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Proportion

Outcome (color) and count of Outcome (size). The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Scatter Chart : BMI with Blood Pressure



Select Var 2 vs. Select Var 1 broken down by Axis Var 2 vs. Axis Var1. Color shows details about Outcome. Details are shown for Outcome. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Hist - Age

The trend of count of Age for Age (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

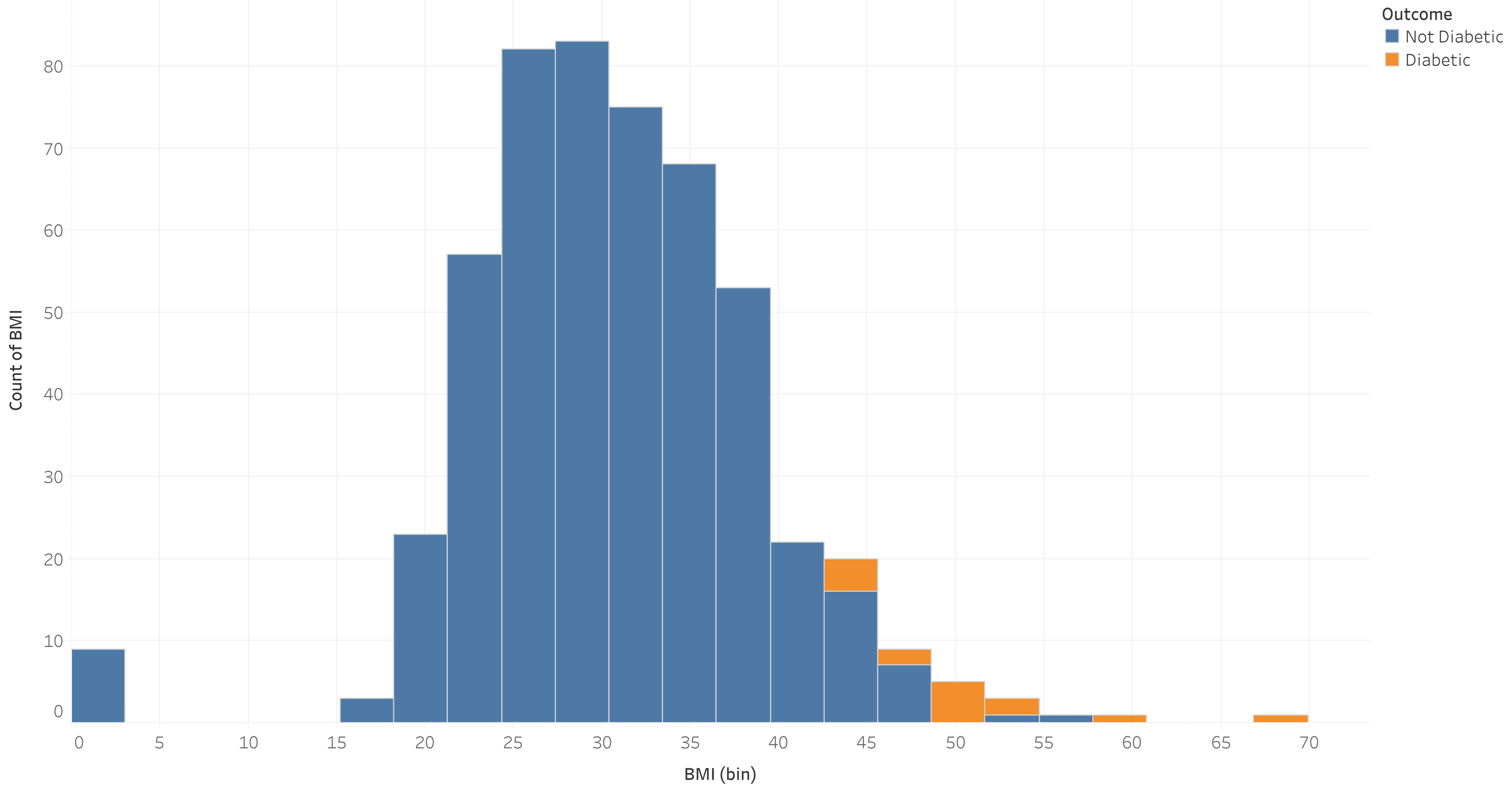
Outcome

Hist - Blood Pressure

The trend of count of Blood Pressure for Blood Pressure (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Outcome

Hist - BMI



The trend of count of BMI for BMI (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which keeps BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Hist - DPF

The trend of count of Diabetes Pedigree Function for Diabetes Pedigree Function (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Outcome

Hist - Glucose

The trend of count of Glucose for Glucose (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Outcome

Hist - Insulin

The trend of count of Insulin for Insulin (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

Outcome

Hist - Pregnancies

The trend of count of Pregnancies for Pregnancies (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

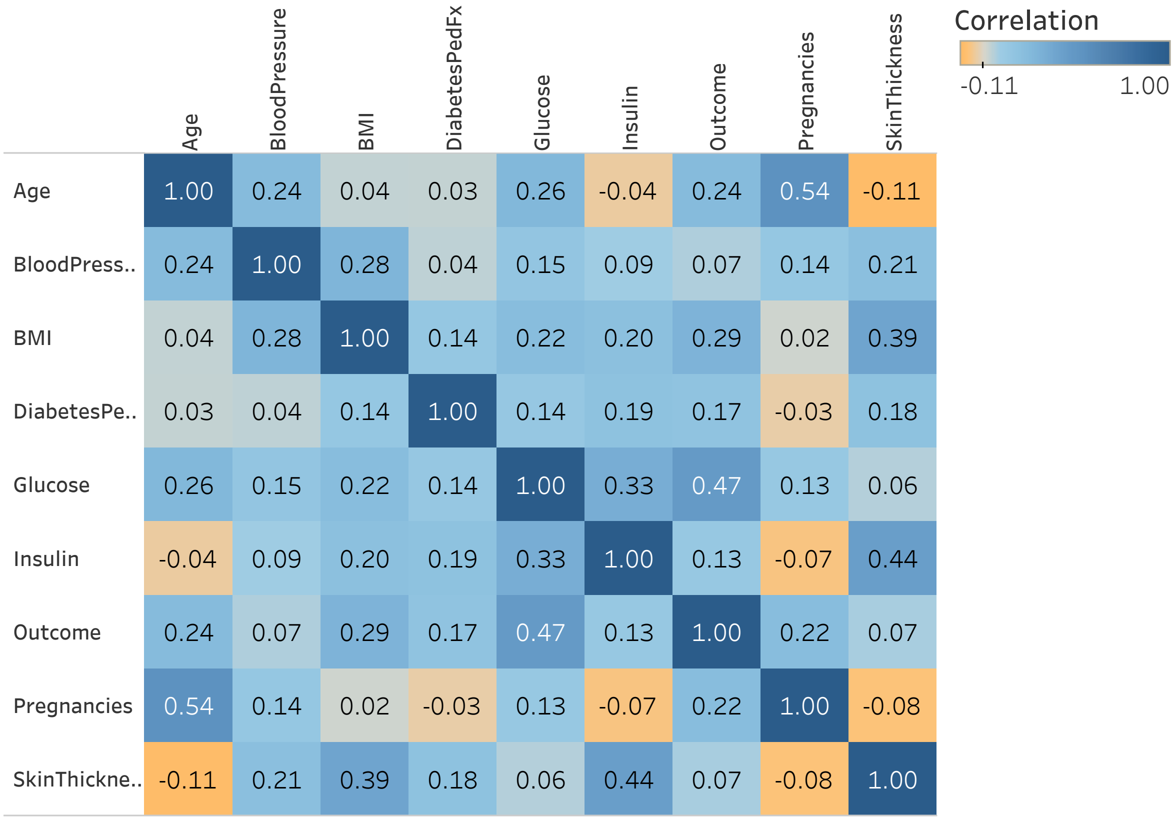
Outcome

Hist - Skin Thickness

The trend of count of Skin Thickness for Skin Thickness (bin). Color shows details about Outcome. The data is filtered on Histogram Select, which excludes BMI. The view is filtered on Outcome, which keeps Not Diabetic and Diabetic.

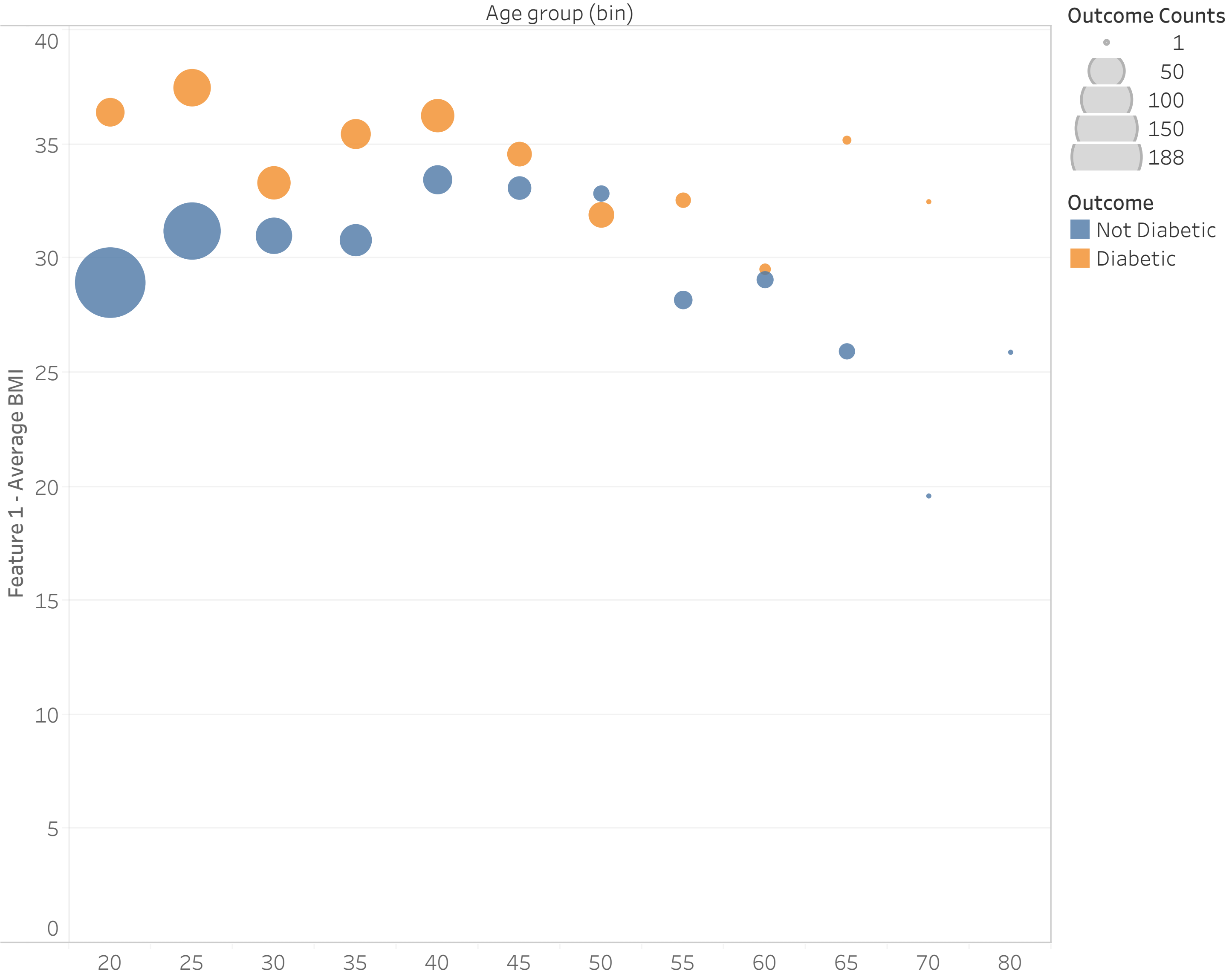
Outcome

Correlation Heatmap



Correlation Coeff broken down by Feature 1 vs. Feature 2. Color shows Correlation Coeff. The marks are labeled by Correlation Coeff.

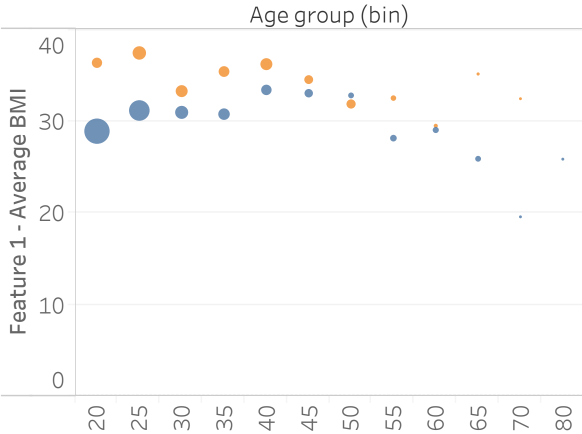
Bubble Chart



Bubble Chart of each variable v/s Age groups. X-axis is age group . Y-axis indicates average values of the other Y-axis variable which can be changed from Feature 1 for the corresponding age group . Color shows details about Outcome and Size shows Outcome Counts(no of samples in the age group on X-axis).

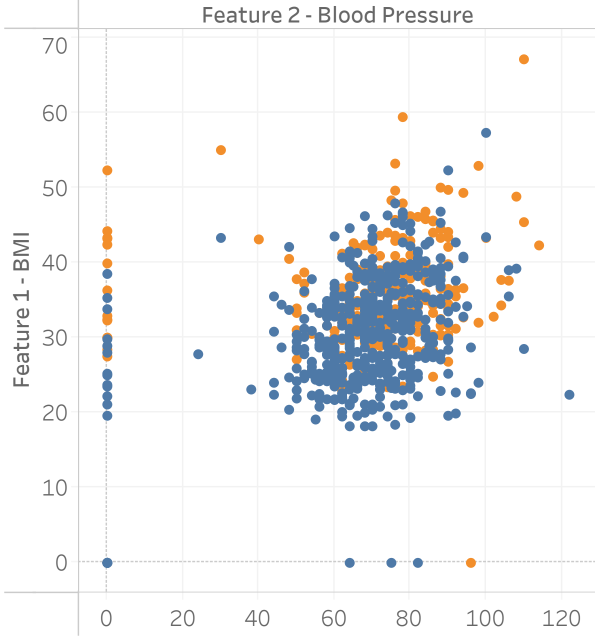
Diabetes Dataset Dashboard

Bubble Chart



Bubble Chart of each variable v/s Age groups. X-axis is age group . Y-axis indicates average values of the other Y-axis variable which can be changed from Feature 1 for the corresponding age group . Color shows details about Outcome and Size shows Outcome Counts(no of samples in the age group on X-axis).

Scatter Chart : BMI with Blood Pressure

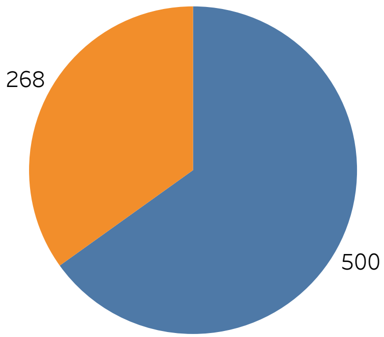


Feature 1
BMI

Feature 2 Scatter Plot
Blood Pressure

Outcome
All

Outcome
Not Diabetic   Diabetic



Correlati..

-0.11   1.00

	Age	BloodPress..	BMI	DiabetesPe..	Glucose	Insulin	Outcome	Pregnancies	SkinThickn..
Age	1.00	0.24	0.04	0.03	0.26	-0.04	0.24	0.54	-0.11
BloodPress..	0.24	1.00	0.28	0.04	0.15	0.09	0.07	0.14	0.21
BMI	0.04	0.28	1.00	0.14	0.22	0.20	0.29	0.02	0.39
DiabetesPe..	0.03	0.04	0.14	1.00	0.14	0.19	0.17	-0.03	0.18
Glucose	0.26	0.15	0.22	0.14	1.00	0.33	0.47	0.13	0.06
Insulin	-0.04	0.09	0.20	0.19	0.33	1.00	0.13	-0.07	0.44
Outcome	0.24	0.07	0.29	0.17	0.47	0.13	1.00	0.22	0.07
Pregnancies	0.54	0.14	0.02	-0.03	0.13	-0.07	0.22	1.00	-0.08
SkinThickne..	-0.11	0.21	0.39	0.18	0.06	0.44	0.07	-0.08	1.00

Histogram for Feature 1 - BMI