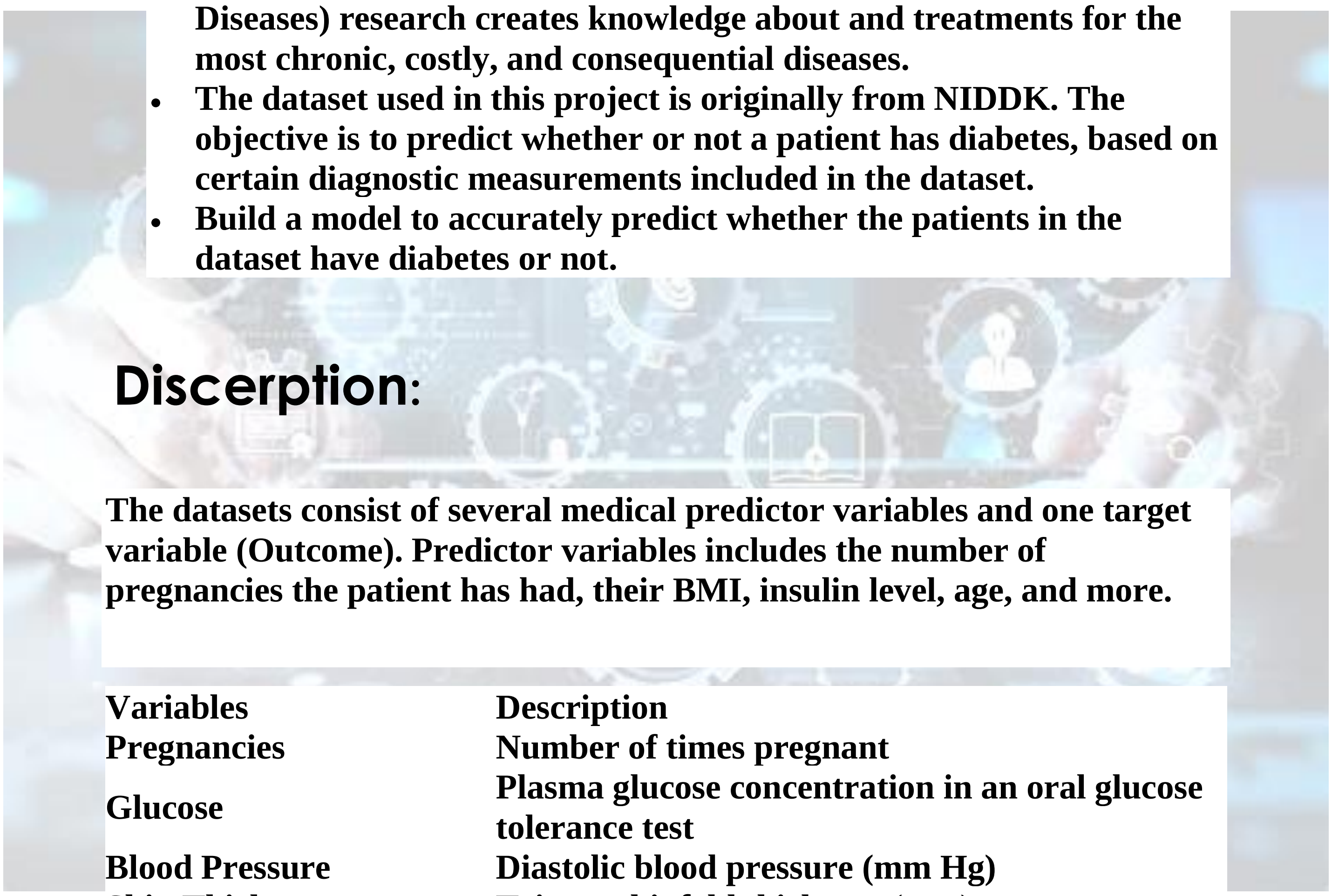
HEALTH CARE

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.**



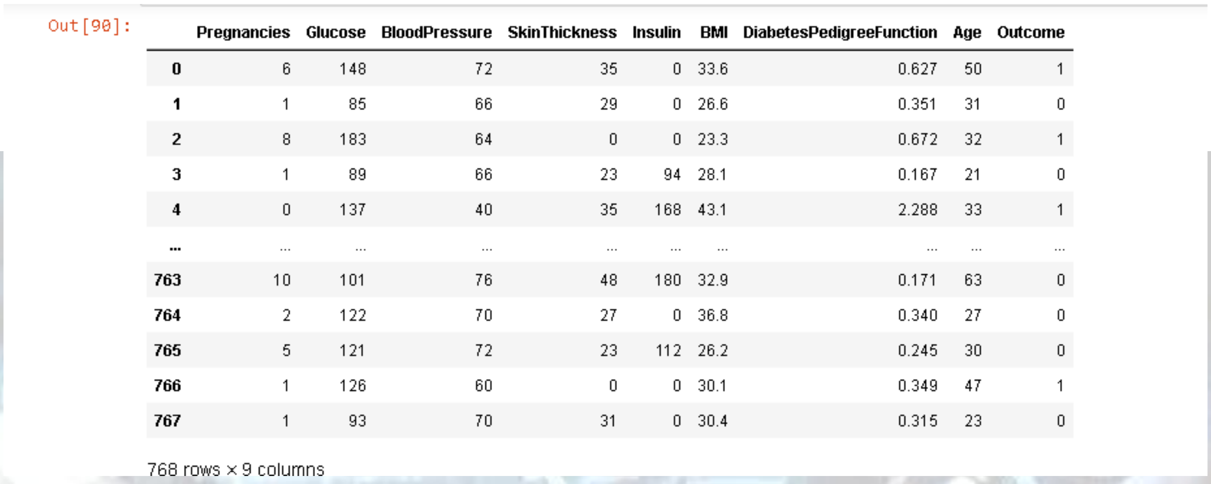
Discerption:

The datasets consist 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**
Blood Pressure	**Diastolic blood pressure (mm Hg)**
Skin Thickness	**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**

ANALYSIS:

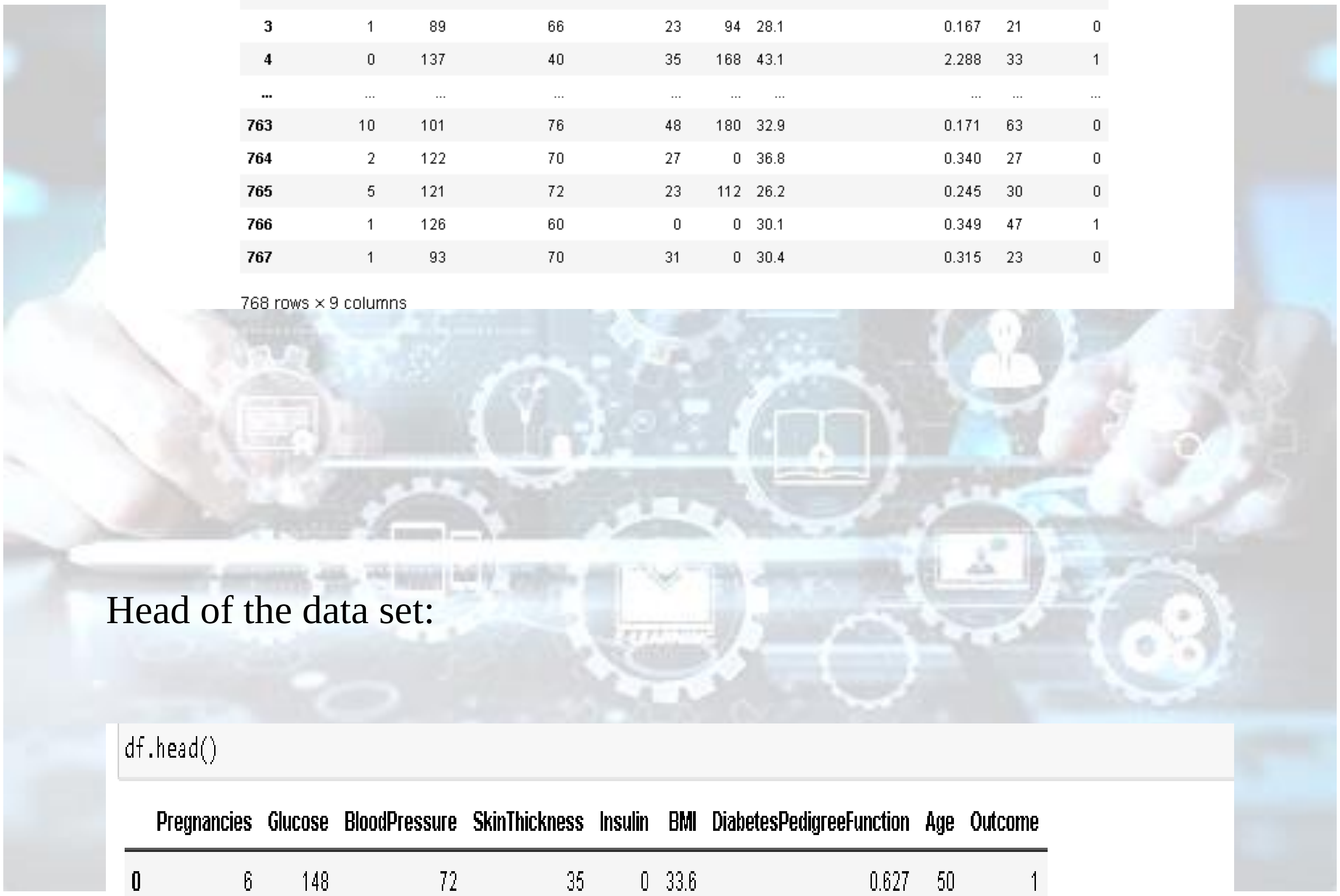
Data Exploration:



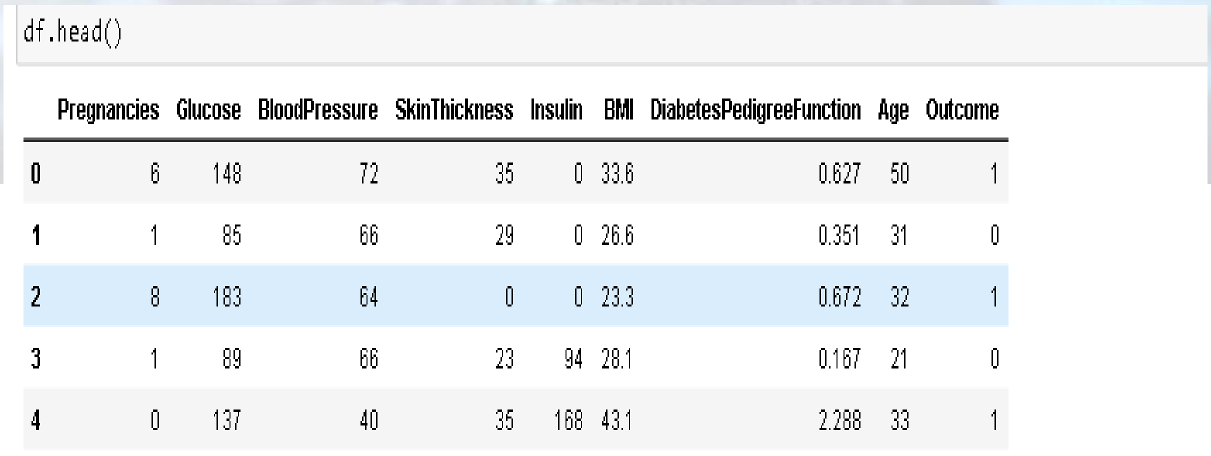
Out[90]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns



Head of the data set:



```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

Descriptive Analysis:

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

• Blood Pressure

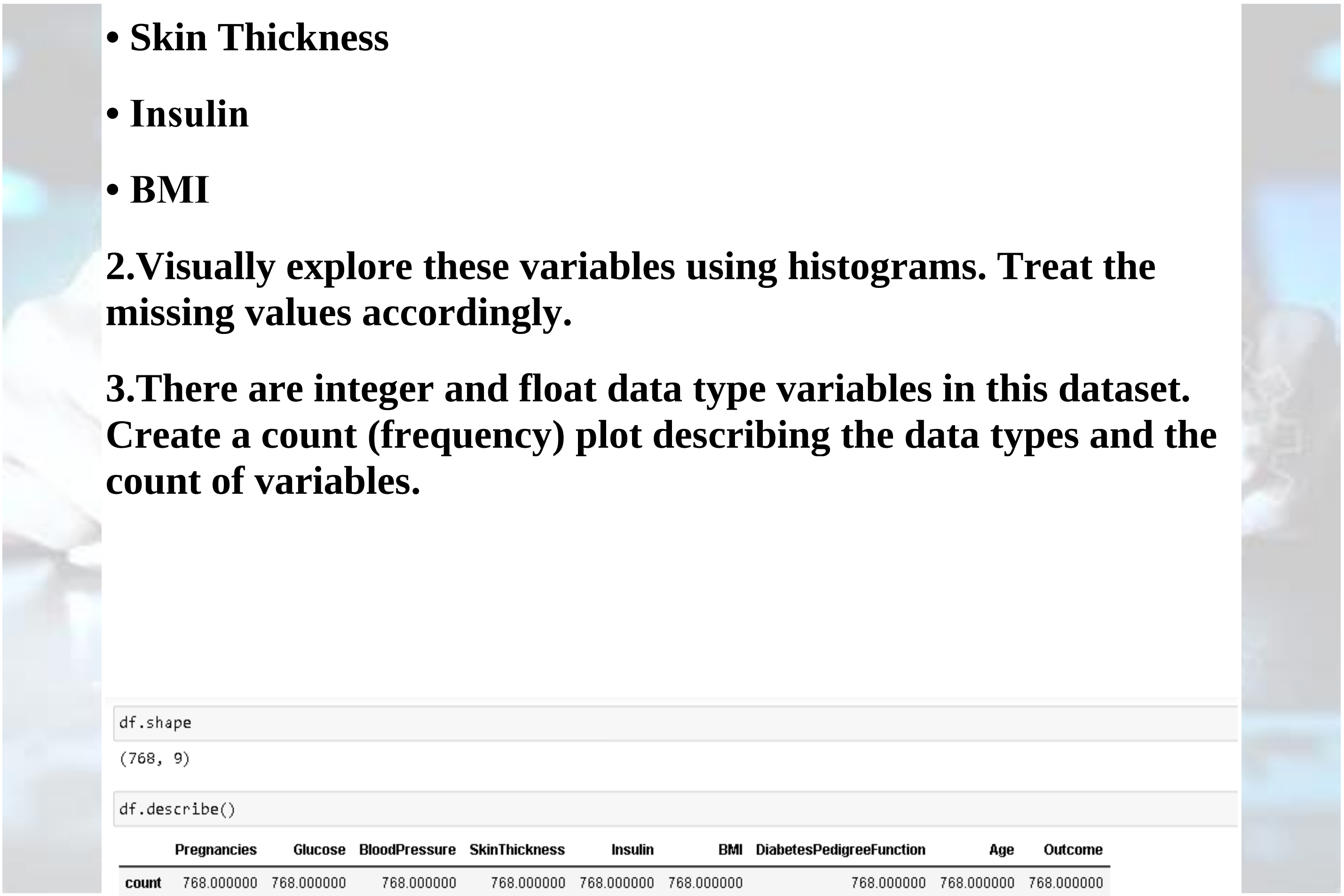
• Skin Thickness

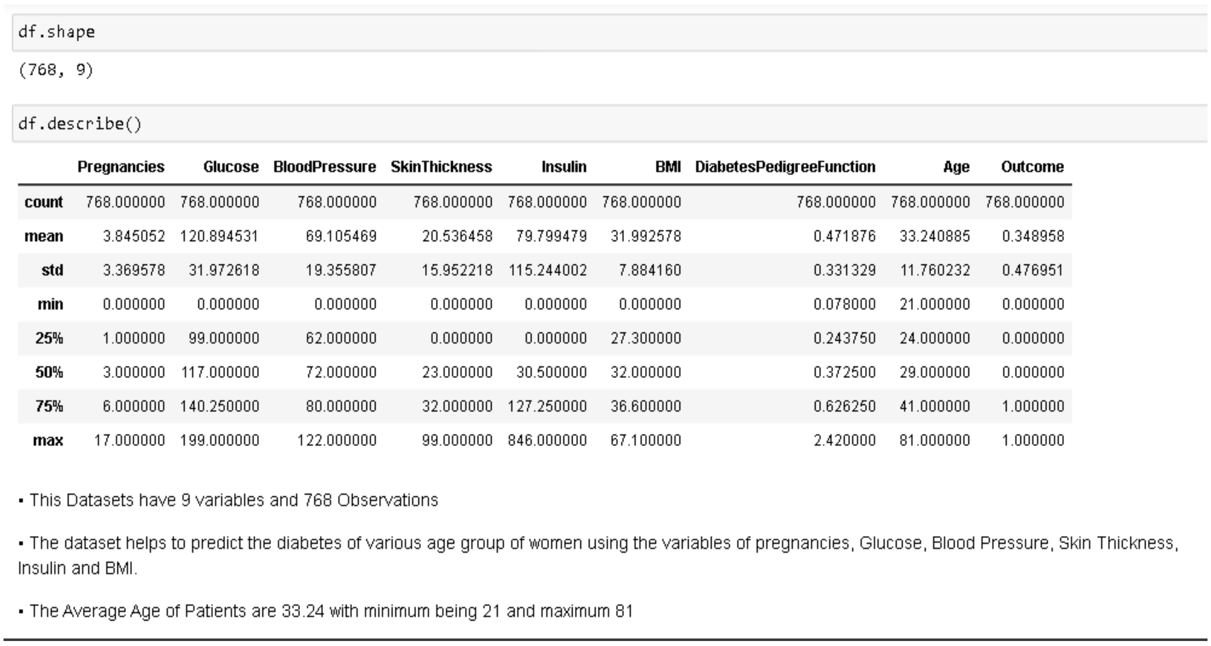
• Insulin

• 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.





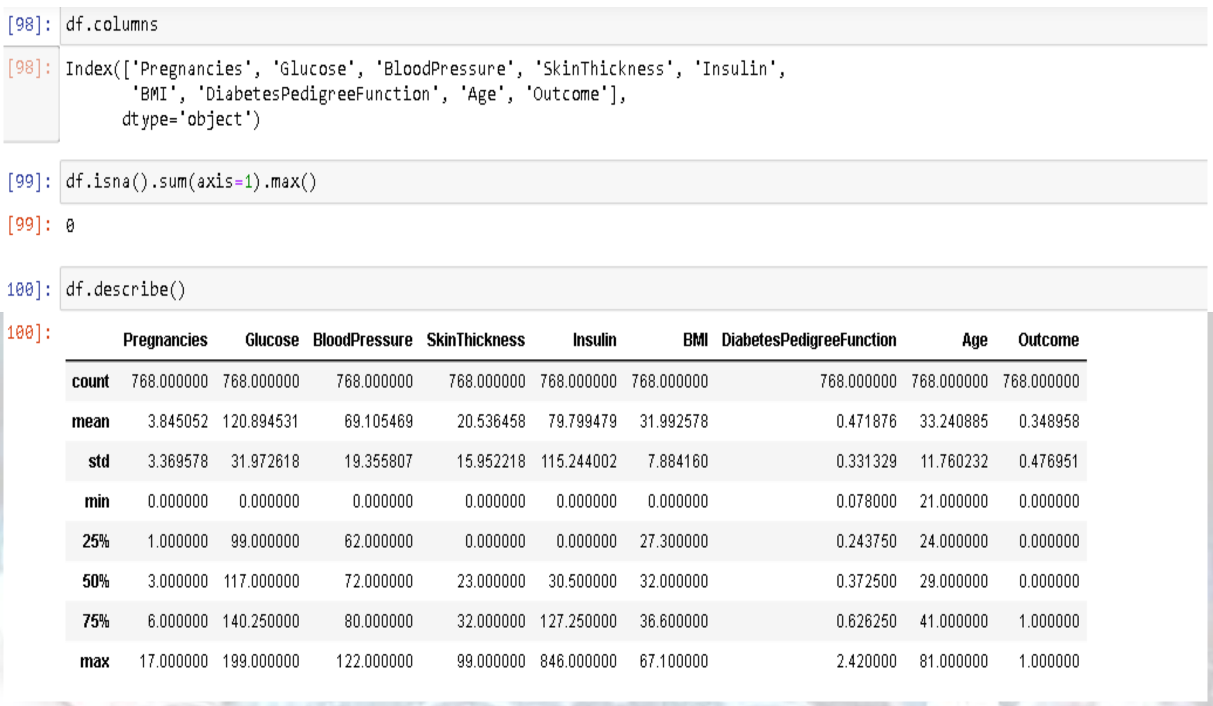
```
df.shape
```

```
(768, 9)
```

```
df.describe()
```

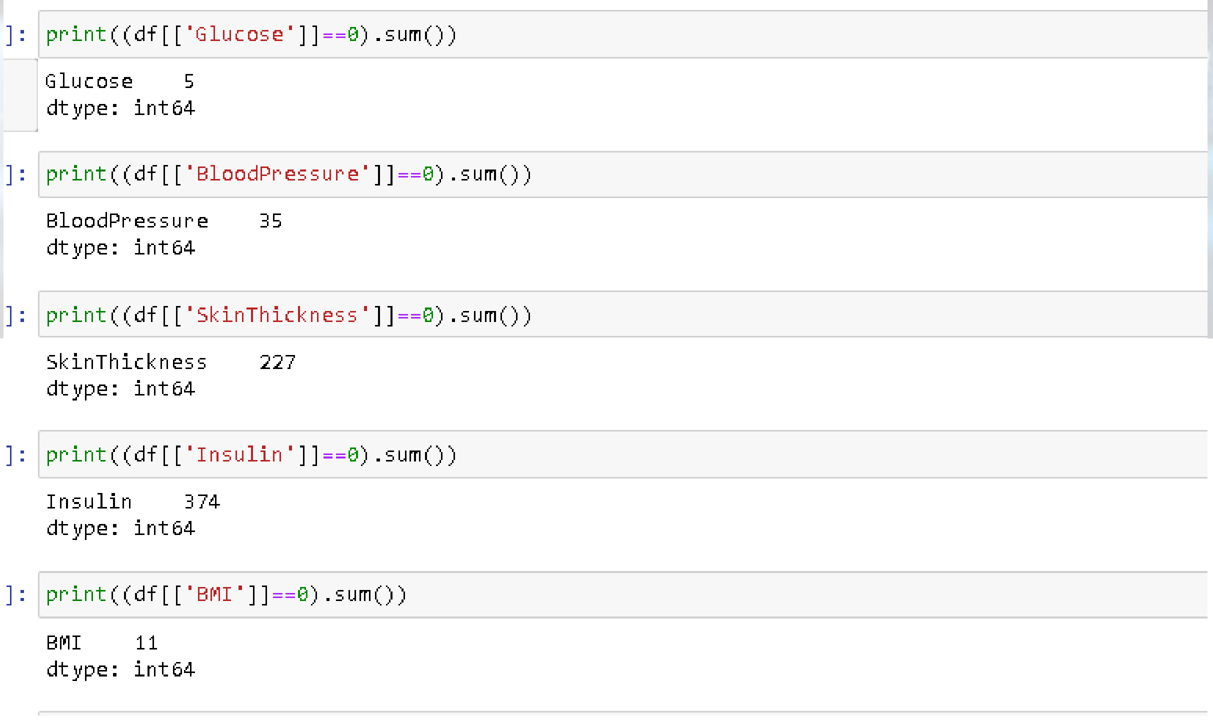
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

- This Datasets have 9 variables and 768 Observations
- The dataset helps to predict the diabetes of various age group of women using the variables of pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin and BMI.
- The Average Age of Patients are 33.24 with minimum being 21 and maximum 81

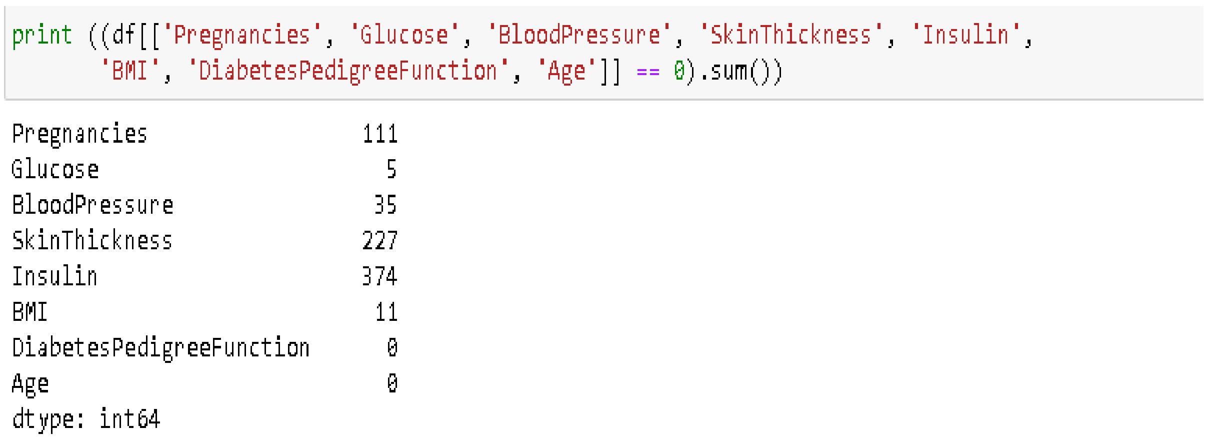


```
[98]: df.columns

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

[99]: df.isna().sum(axis=1).max()

[99]: 0

100]: df.describe()

100]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	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	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

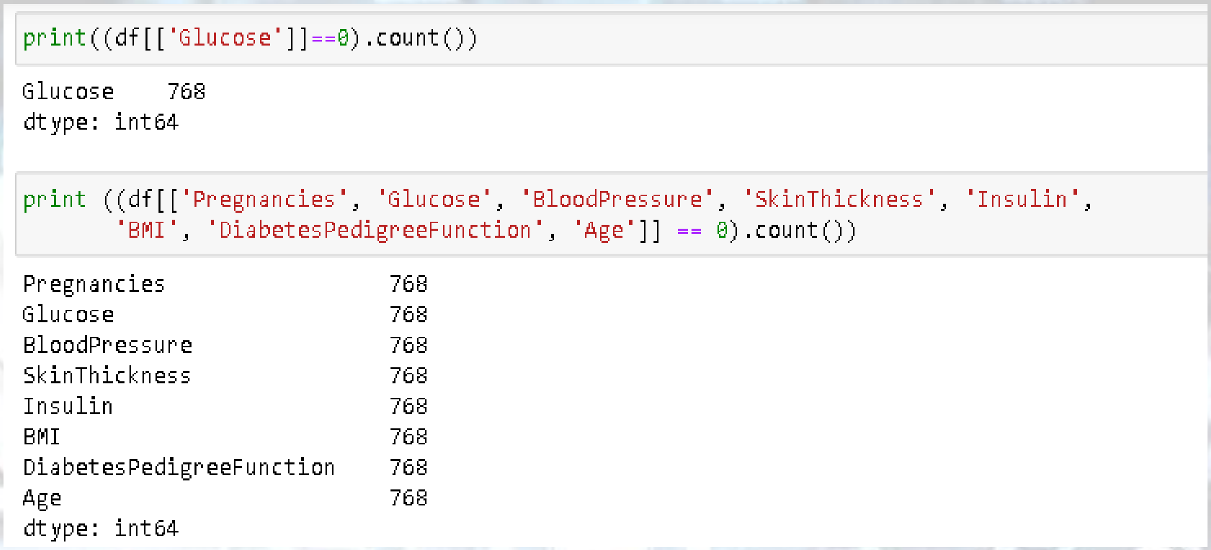


```
]: print((df[['Glucose']]==0).sum())

   Glucose    5
   dtype: int64

]: print((df[['BloodPressure']]==0).sum())

   BloodPressure    35
   dtype: int64

]: print((df[['SkinThickness']]==0).sum())

   SkinThickness    227
   dtype: int64

]: print((df[['Insulin']]==0).sum())

   Insulin    374
   dtype: int64

]: print((df[['BMI']]==0).sum())

   BMI    11
   dtype: int64
```



```
print ((df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']] == 0).sum())
```

```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
dtype: int64
```

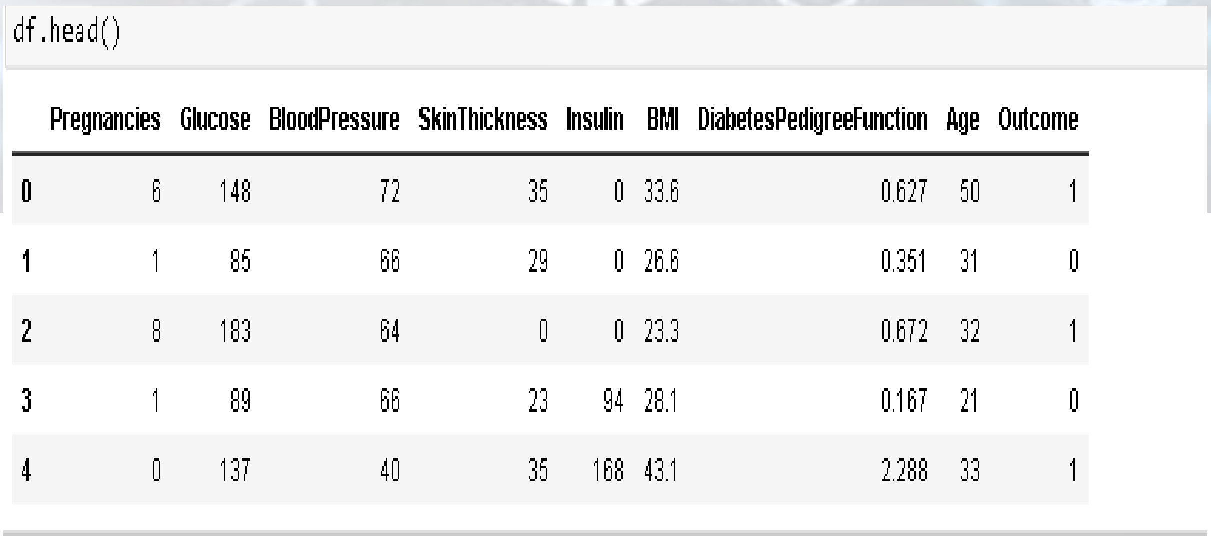


```
print((df[['Glucose']]==0).count())
```

```
Glucose    768
dtype: int64
```

```
print ((df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']] == 0).count())
```

```
Pregnancies                 768
Glucose                     768
BloodPressure               768
SkinThickness               768
Insulin                     768
BMI                         768
DiabetesPedigreeFunction    768
Age                         768
dtype: int64
```



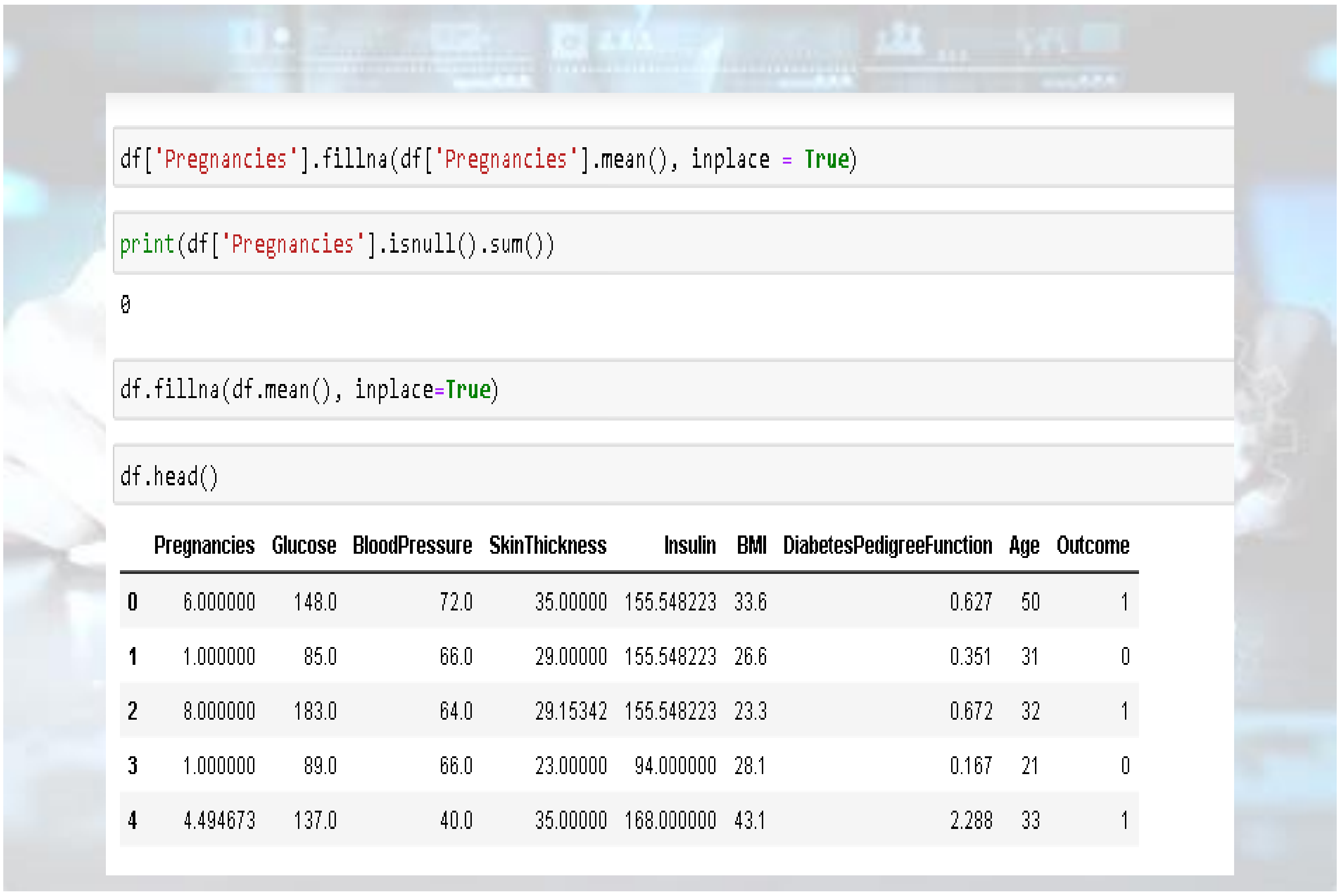
```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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

```
df [['Pregnancies','Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', ]] = df [['Pregnancies','Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', ]].replace(0,np.NaN)
```

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.0	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1.0	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8.0	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1.0	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	NaN	137.0	40.0	35.0	168.0	43.1	2.288	33	1



```
df['Pregnancies'].fillna(df['Pregnancies'].mean(), inplace = True)
```

```
print(df['Pregnancies'].isnull().sum())
```

```
0
```

```
df.fillna(df.mean(), inplace=True)
```

```
df.head()
```

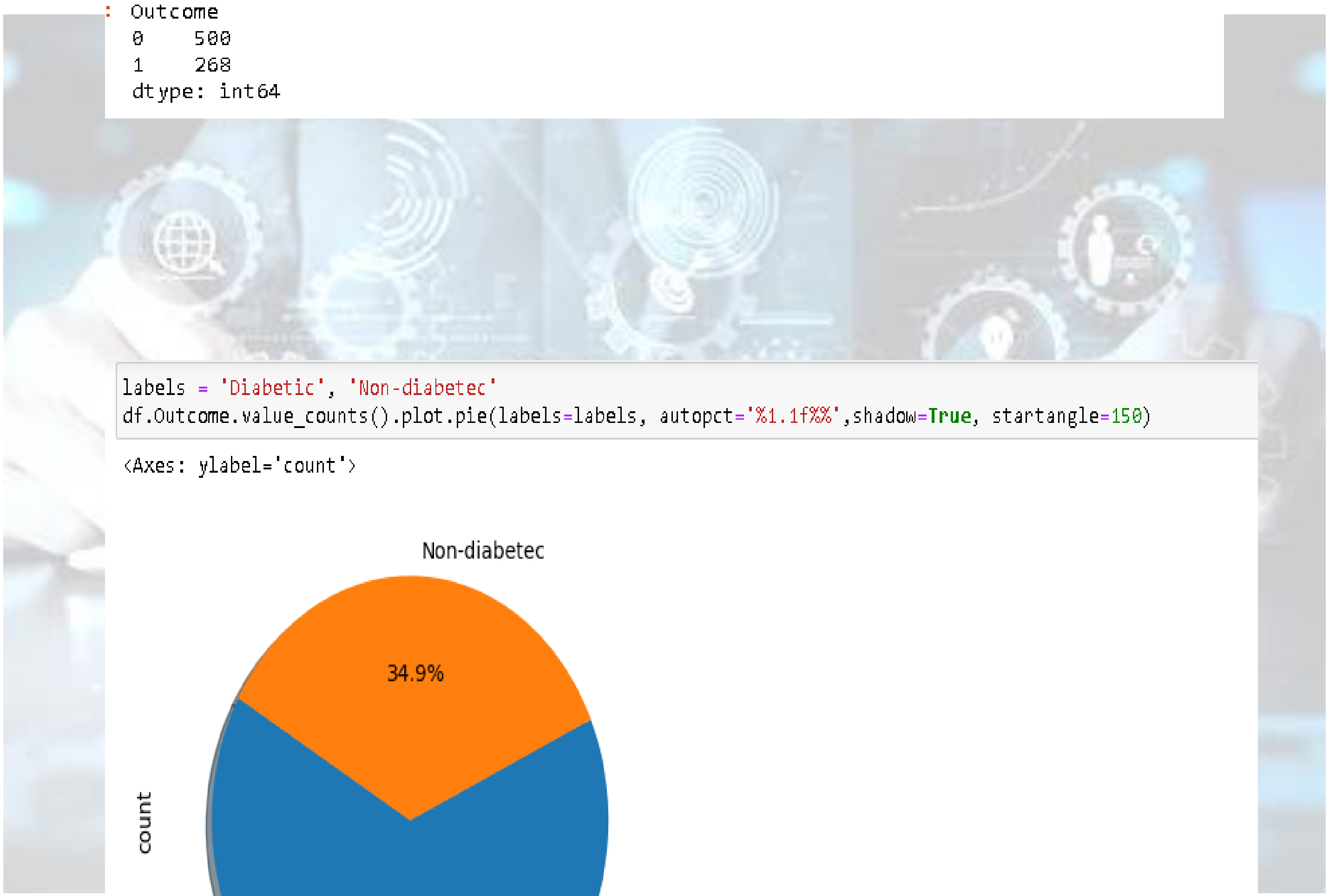
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6.000000	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1
1	1.000000	85.0	66.0	29.00000	155.548223	26.6	0.351	31	0
2	8.000000	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1
3	1.000000	89.0	66.0	23.00000	94.000000	28.1	0.167	21	0
4	4.494673	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1

```python
print (df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']].isnull().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
dtype: int64
```

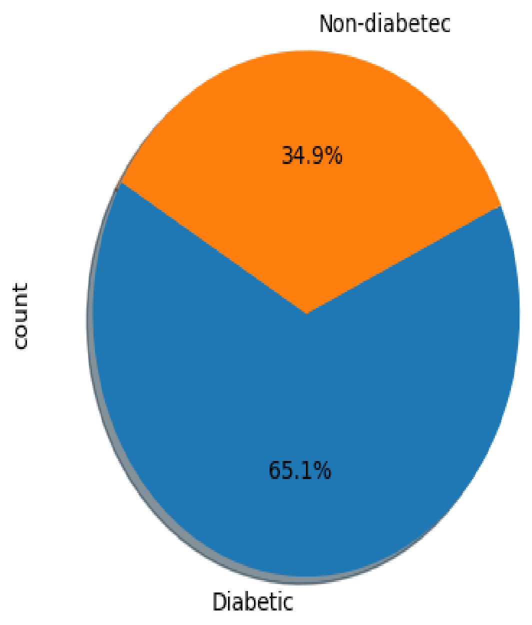
```python
df.groupby('Outcome').size()
```

```
Outcome
0    500
1    268
dtype: int64
```



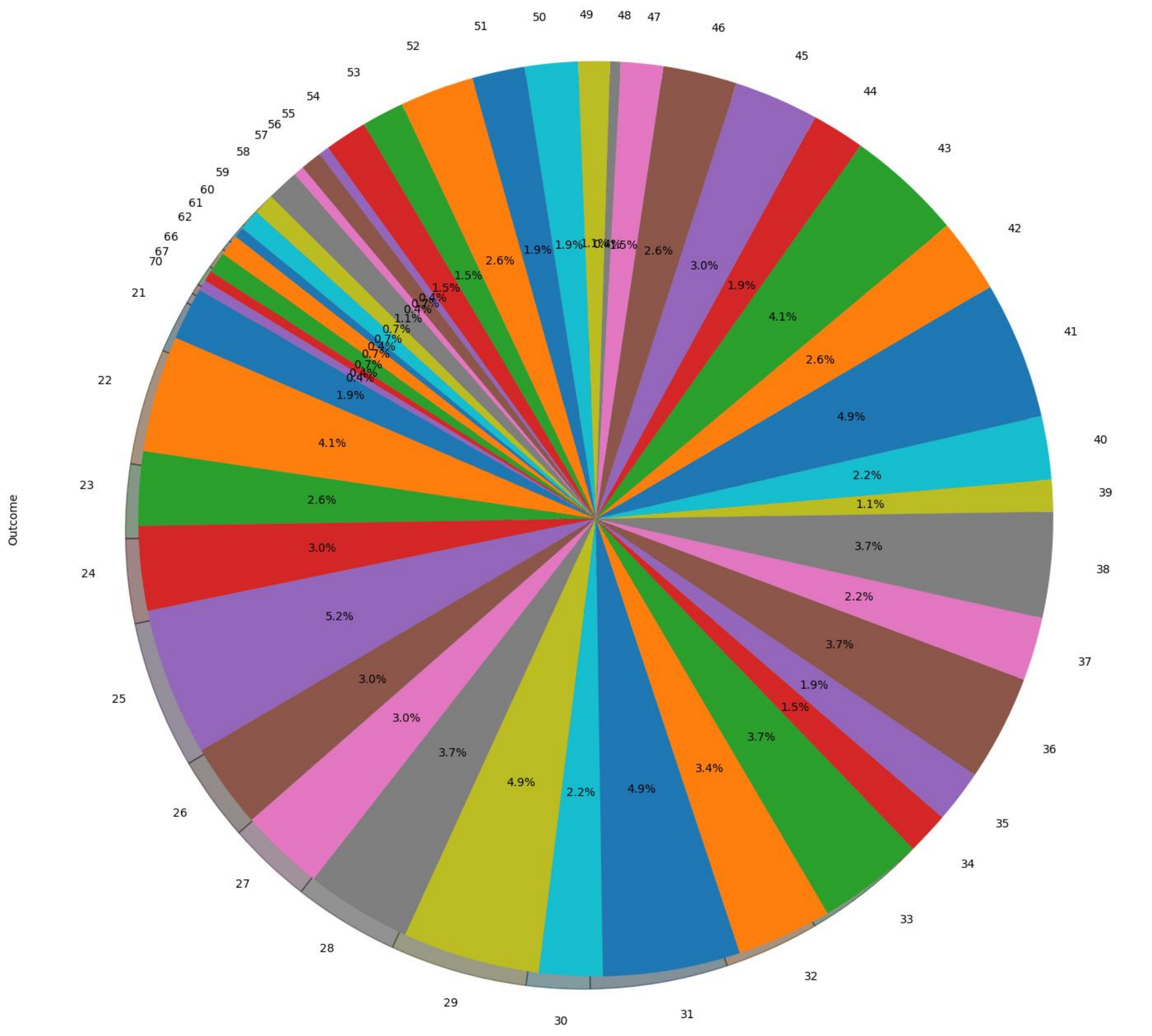
```python
labels = 'Diabetic', 'Non-diabetec'
df.Outcome.value_counts().plot.pie(labels=labels, autopct='%1.1f%%',shadow=True, startangle=150)
```

```
<Axes: ylabel='count'>
```



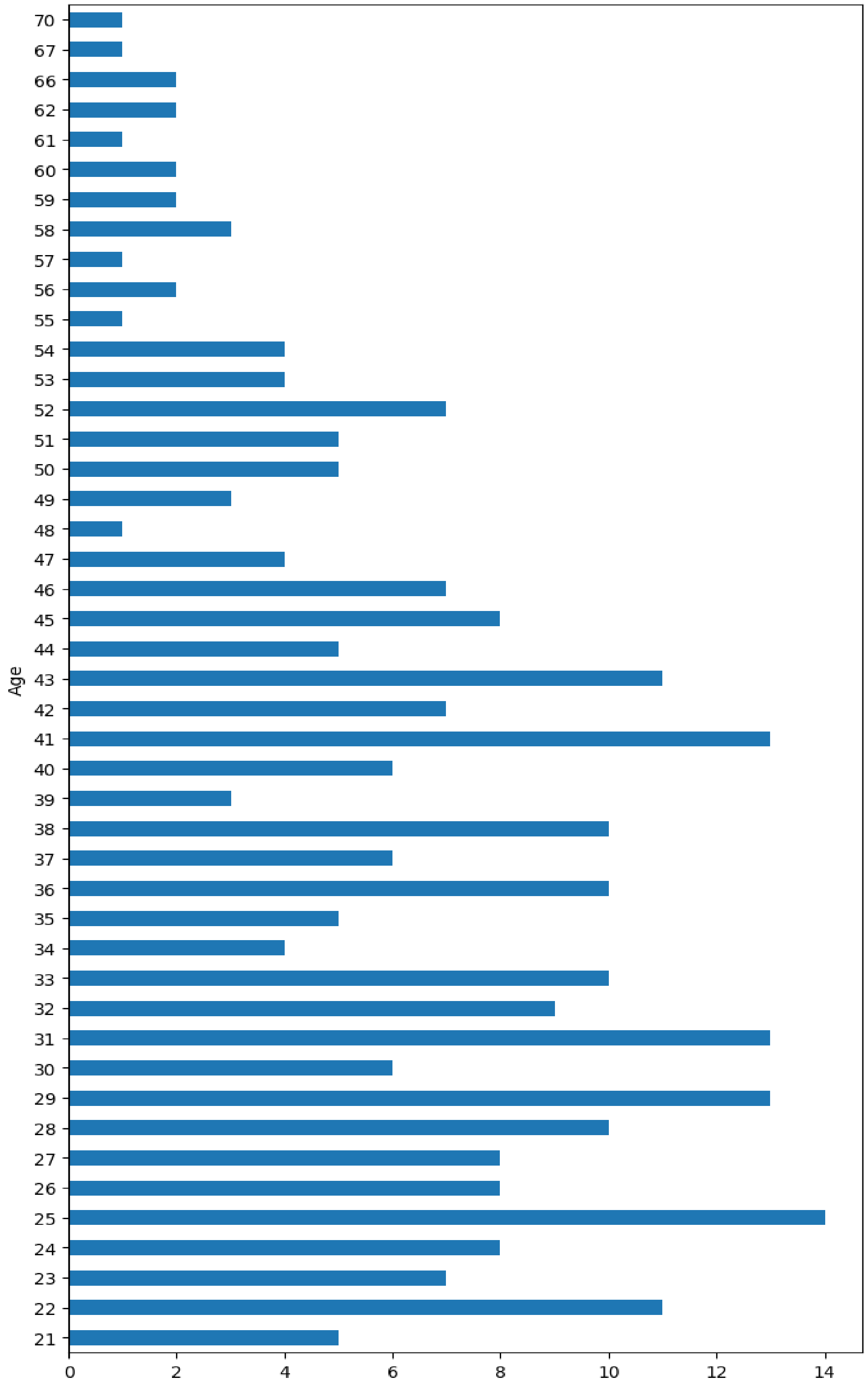
```
diabetes_agewise.groupby('Age')['Outcome'].count().plot.pie(autopct='%1.1f%%',shadow=True, startangle=150, figsize=(35,18))
```

```
<Axes: ylabel='Outcome'>
```



```
diabetes_agewise.groupby('Age')['Outcome'].count().plot(kind= 'barh',  figsize=(8,15))
```

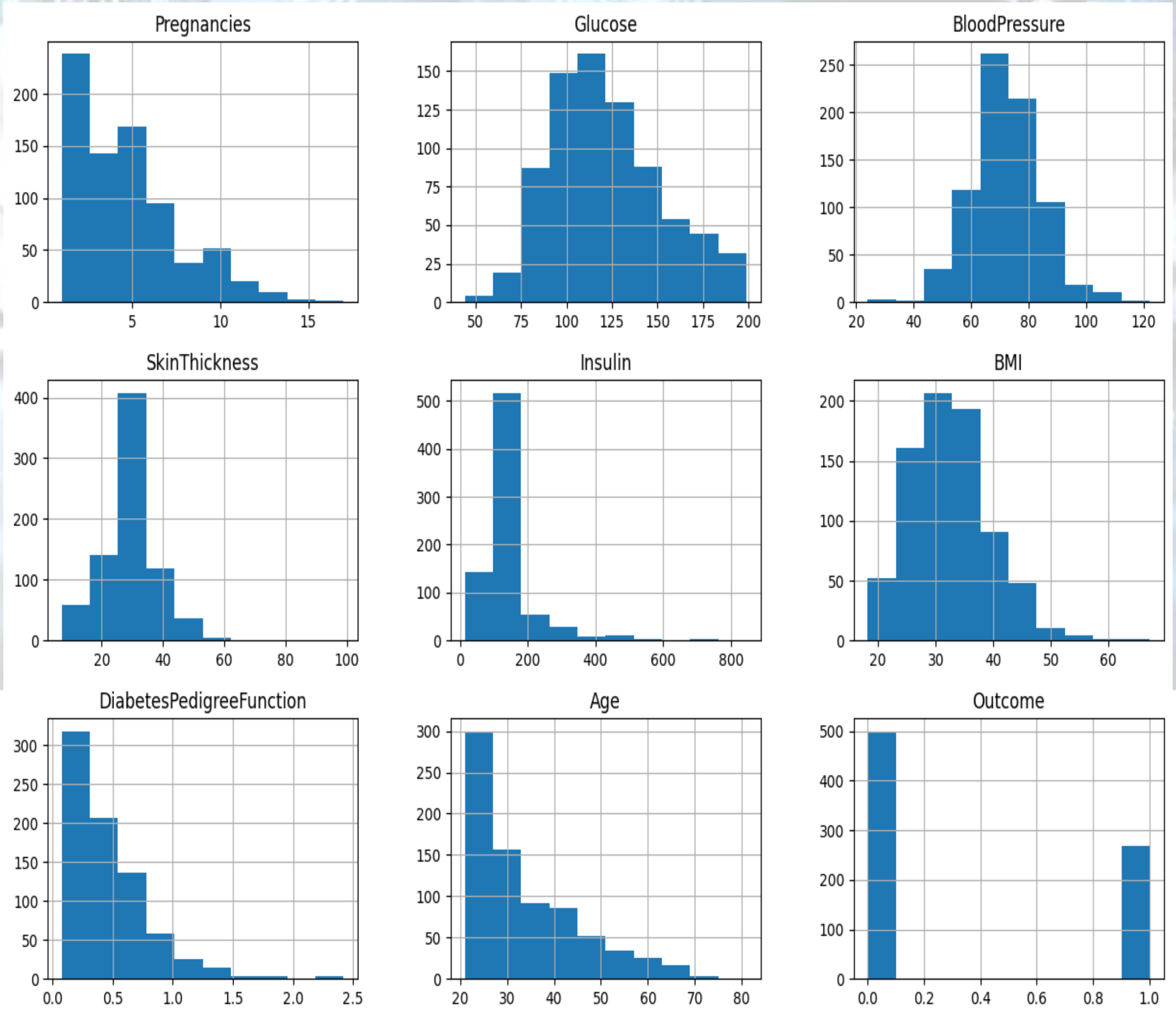
```
<Axes: ylabel='Age'>
```



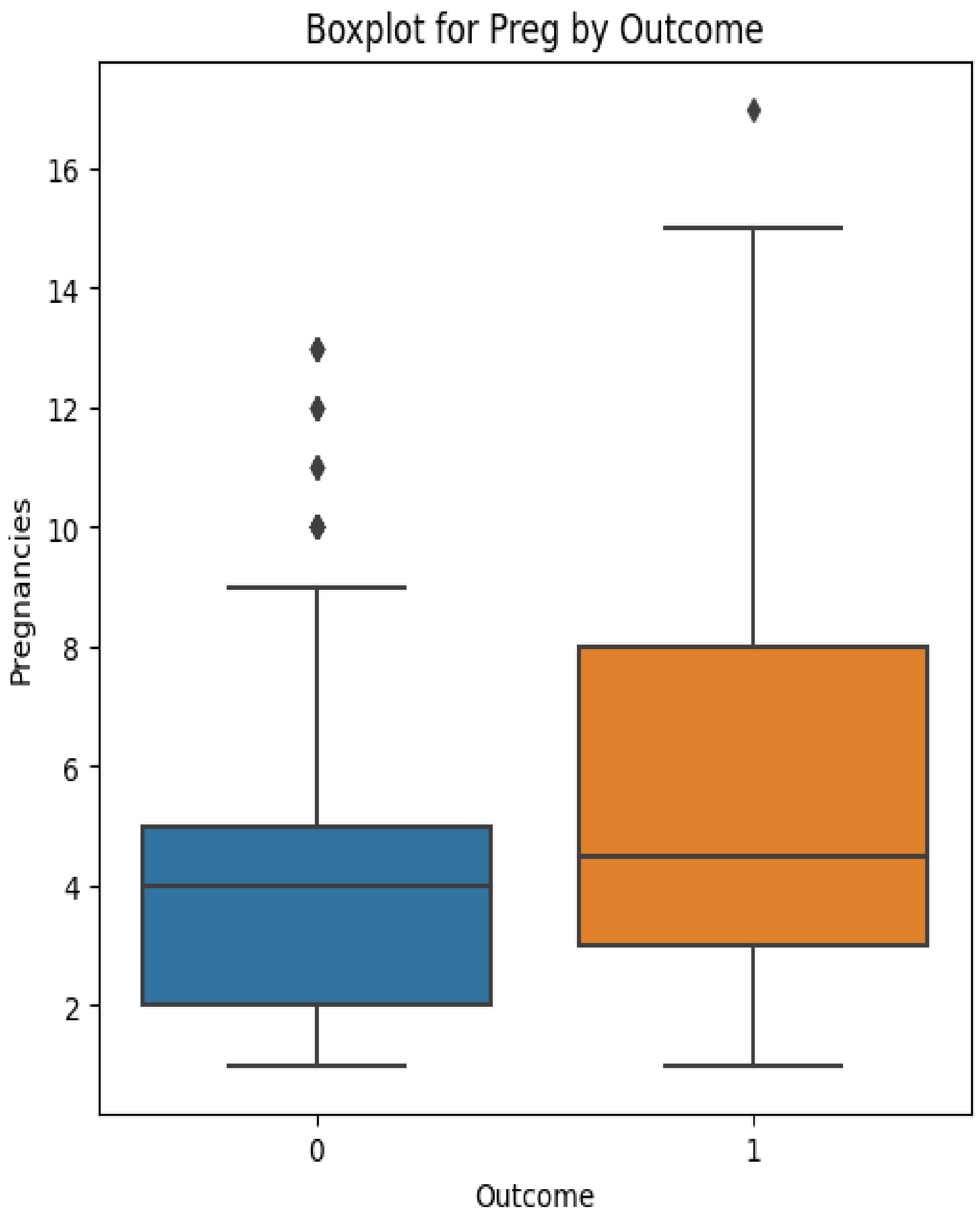
Age
70
67
66
62
61
60
59
58
57
56
55
54
53
52
51
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
0
2
4
6
8
10
12
14

```
df.hist(figsize=(15,10))
```

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

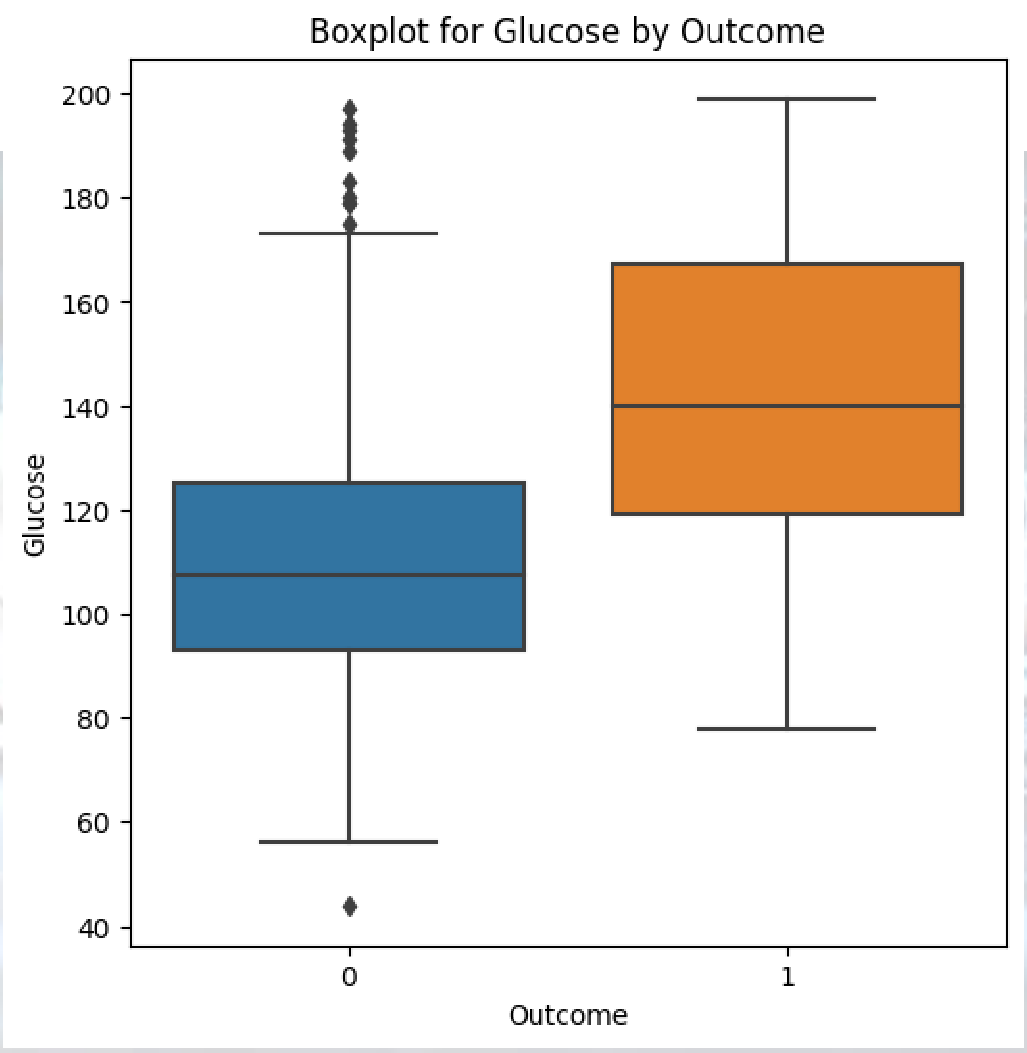


```
# Plots for count of outcome by values
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Pregnancies)
plt.title("Boxplot for Preg by Outcome")
```



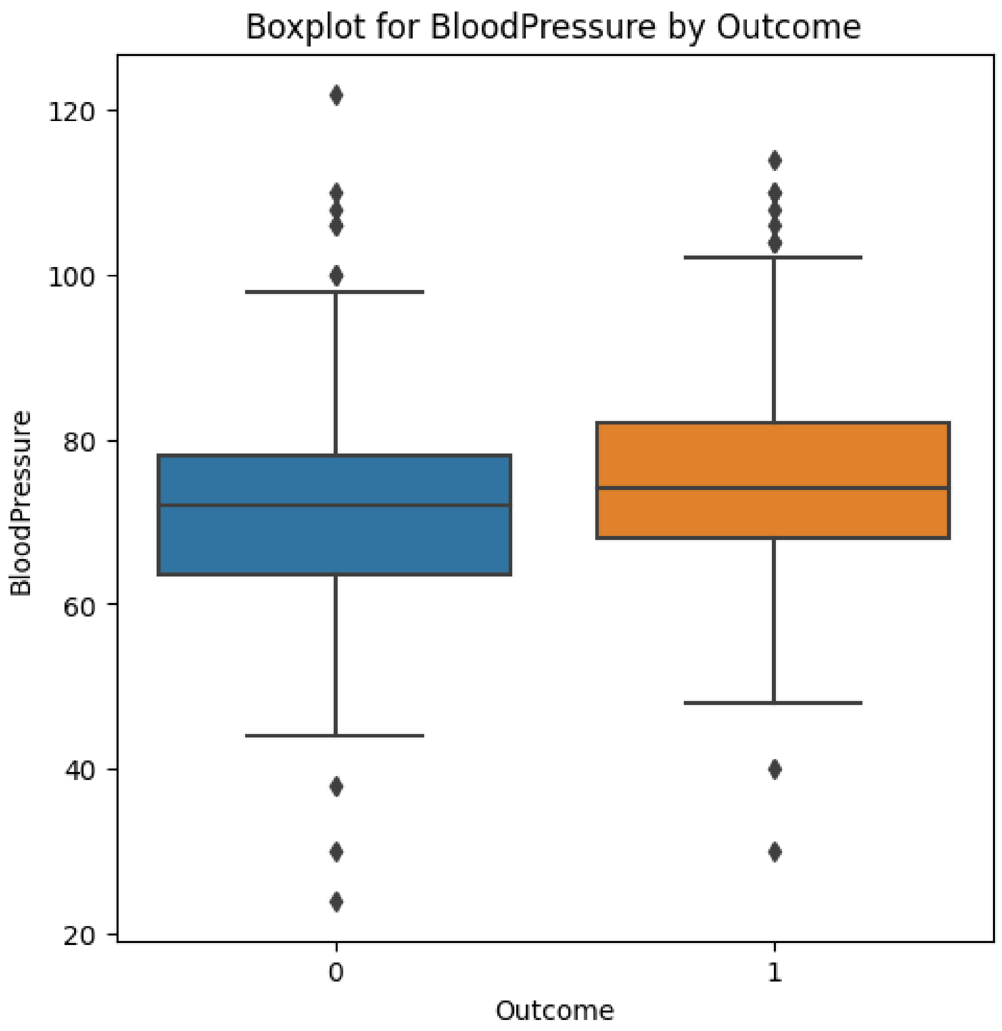
```
# Plot for glucose
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Glucose)
plt.title("Boxplot for Glucose by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for Glucose by Outcome')
```



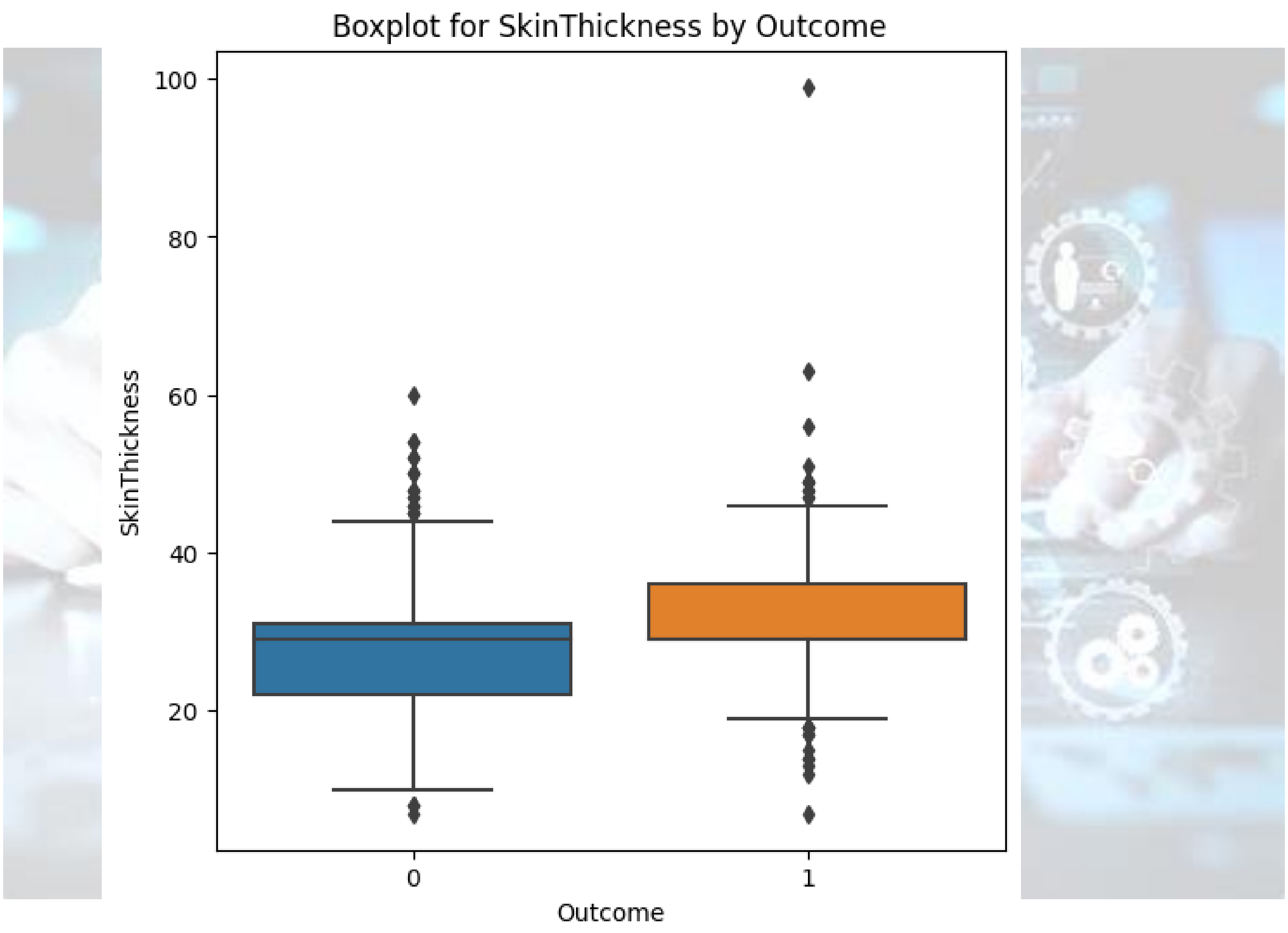
```
: # Plot for BloodPressure
  plt.figure(figsize=(20, 6))
  plt.subplot(1,3,3)
  sns.boxplot(x=df.Outcome,y=df.BloodPressure)
  plt.title("Boxplot for BloodPressure by Outcome")
```

```
: Text(0.5, 1.0, 'Boxplot for BloodPressure by Outcome')
```



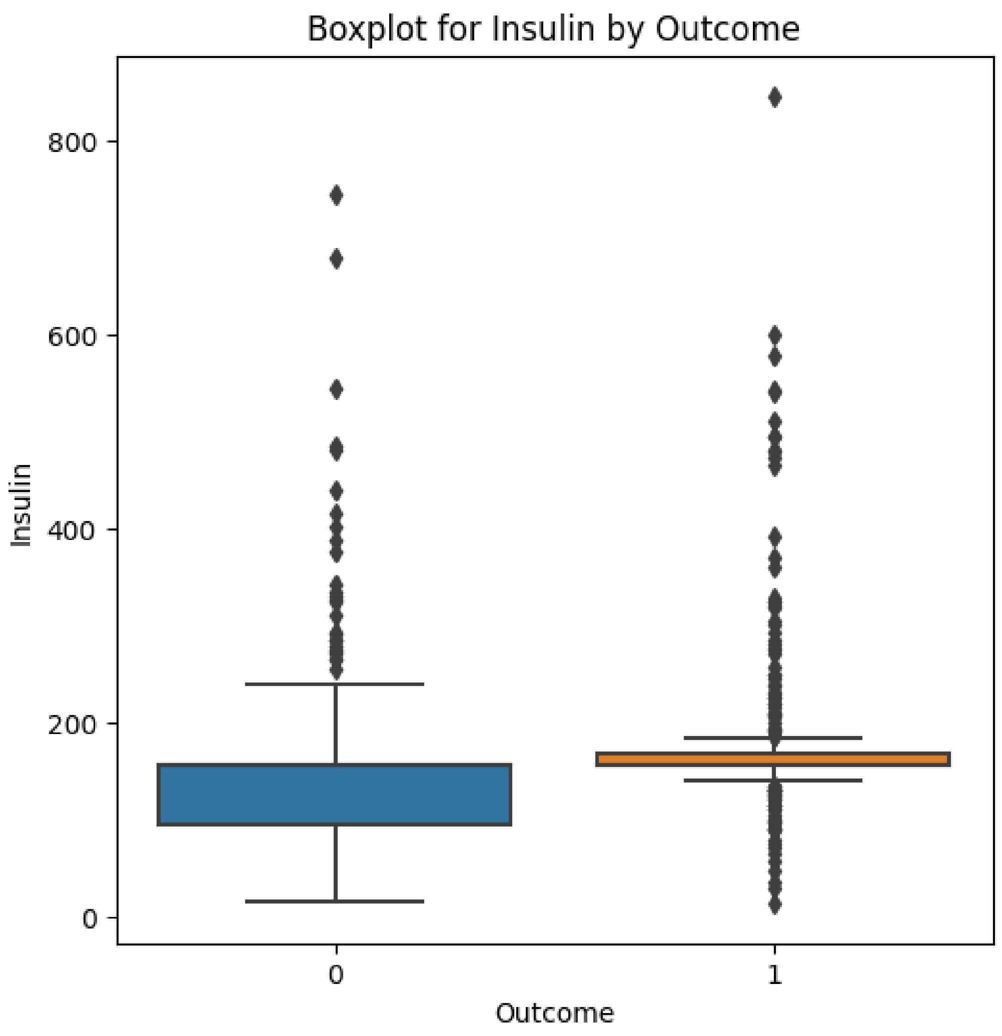
```
: # Plot for SkinThickness
  plt.figure(figsize=(20, 6))
  plt.subplot(1,3,3)
  sns.boxplot(x=df.Outcome,y=df.SkinThickness)
  plt.title("Boxplot for SkinThickness by Outcome")
```

```
: Text(0.5, 1.0, 'Boxplot for SkinThickness by Outcome')
```



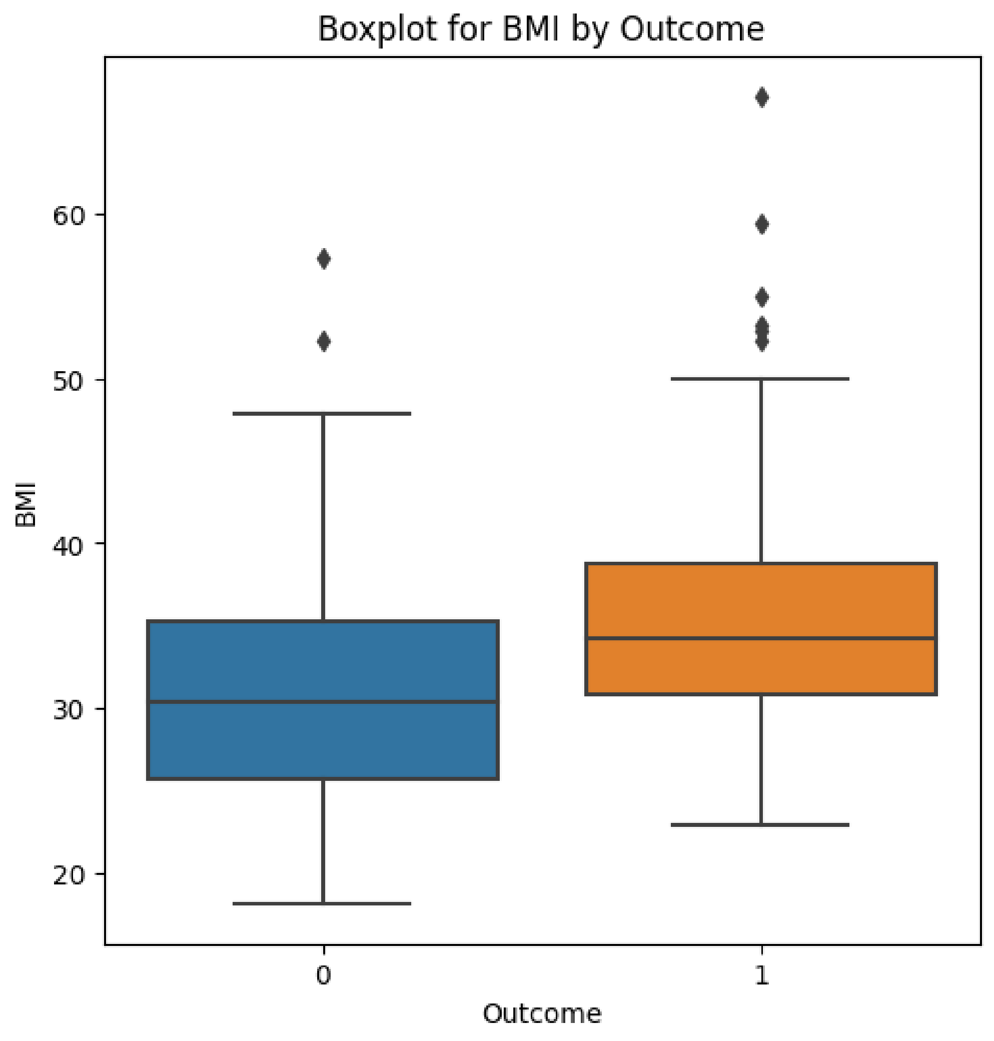
```
# plot for Insulin
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.Insulin)
plt.title("Boxplot for Insulin by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for Insulin by Outcome')
```



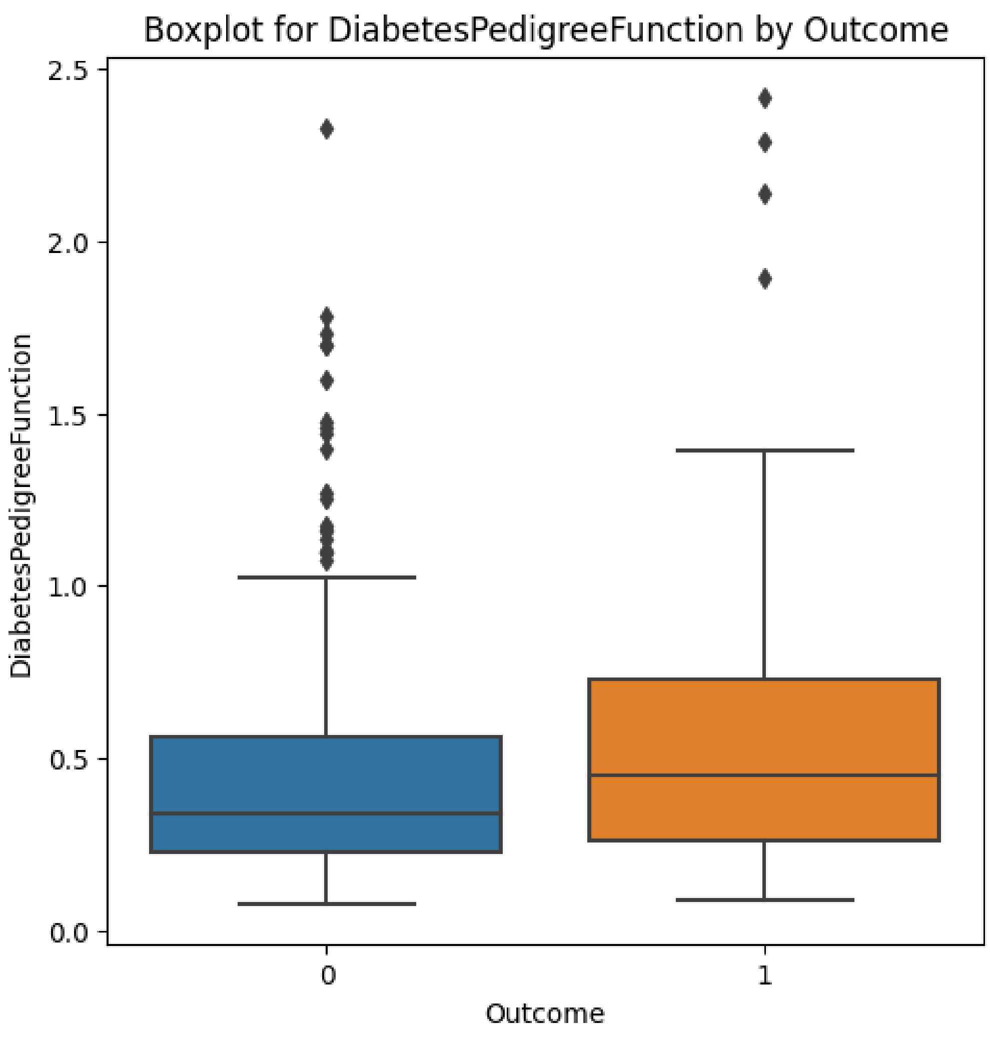
```
# Plot for BMI
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.BMI)
plt.title("Boxplot for BMI by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for BMI by Outcome')
```



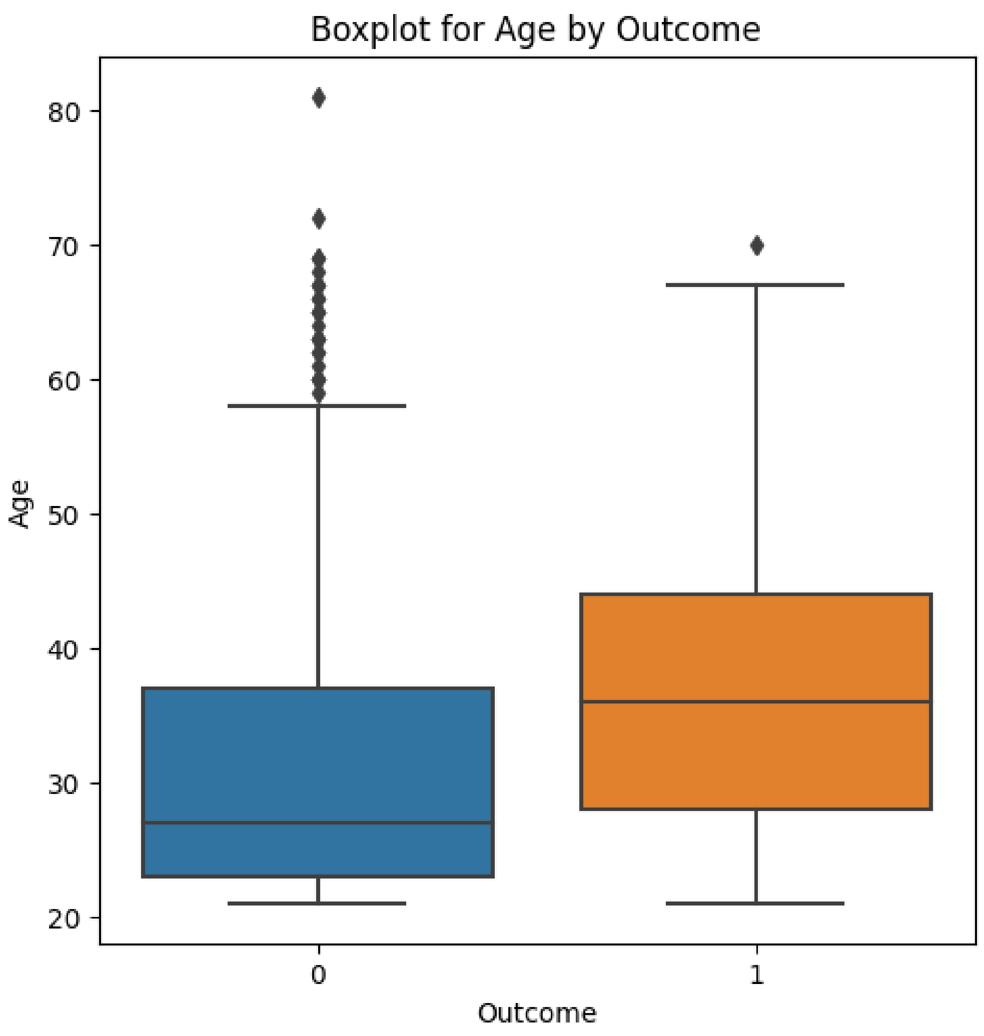
```
# Plot for Diabetes Pedigree Function
plt.figure(figsize=(20, 6))
plt.subplot(1,3,3)
sns.boxplot(x=df.Outcome,y=df.DiabetesPedigreeFunction)
plt.title("Boxplot for DiabetesPedigreeFunction by Outcome")
```

```
Text(0.5, 1.0, 'Boxplot for DiabetesPedigreeFunction by Outcome')
```



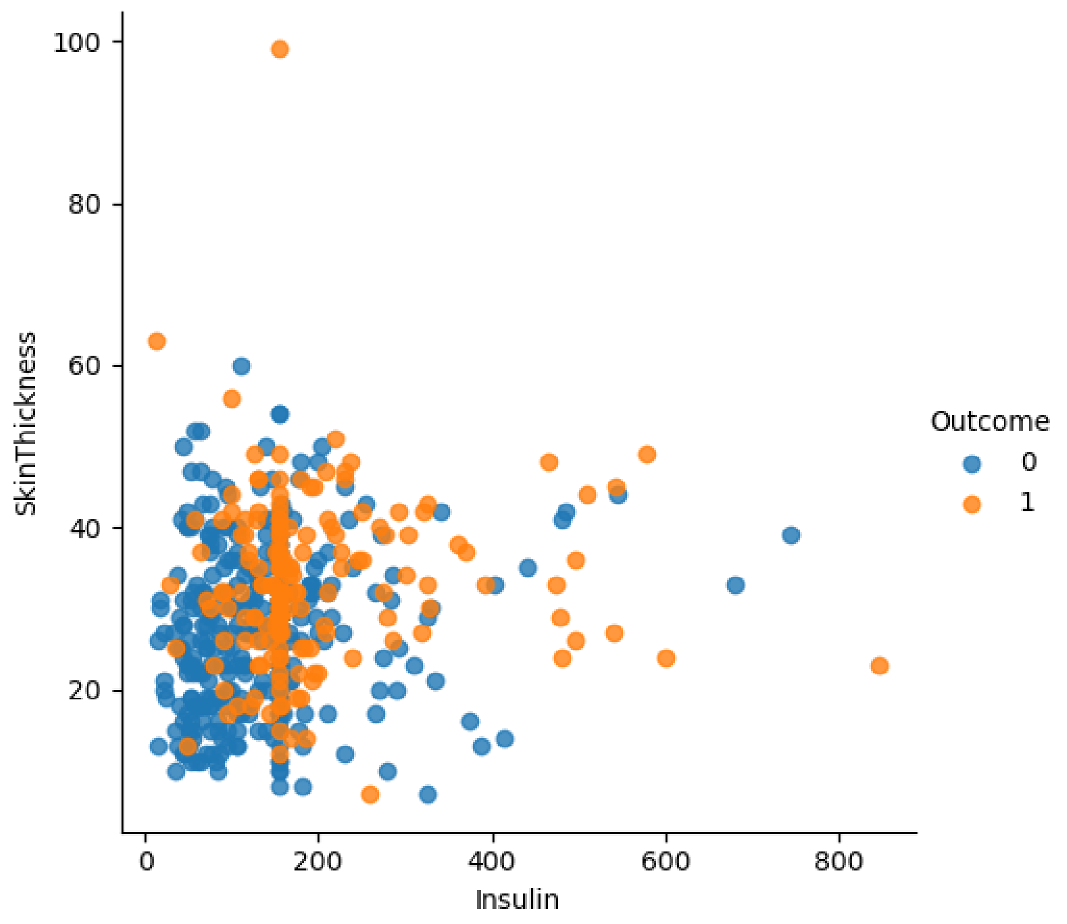
```
: # Plot for Age
  plt.figure(figsize=(20, 6))
  plt.subplot(1,3,3)
  sns.boxplot(x=df.Outcome,y=df.Age)
  plt.title("Boxplot for Age by Outcome")
```

```
: Text(0.5, 1.0, 'Boxplot for Age by Outcome')
```



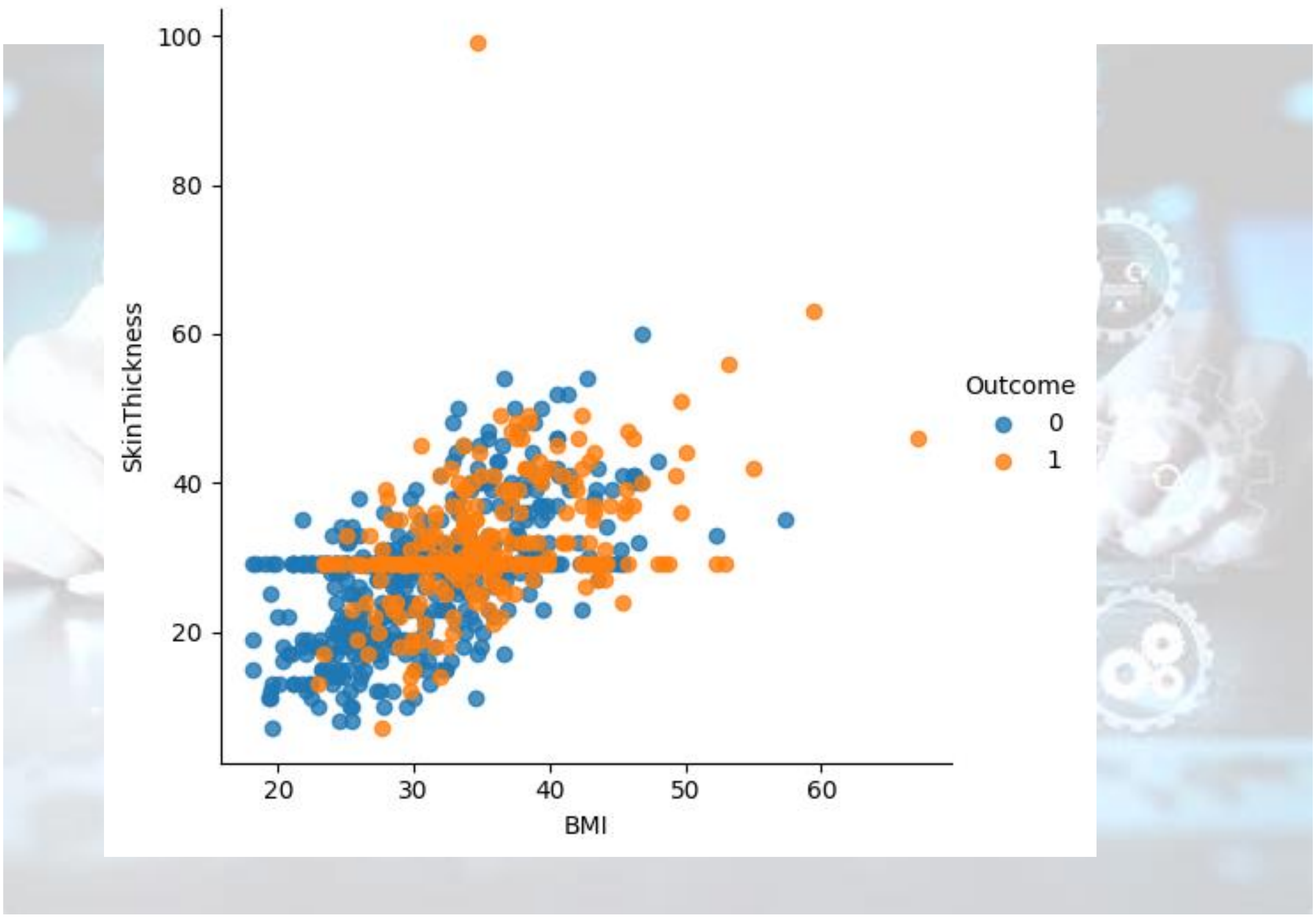
```
# Plot with outcome and variables
sns.lmplot(x='Insulin',y='SkinThickness',data=df,fit_reg=False,hue='Outcome')
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae273e0d90>
```



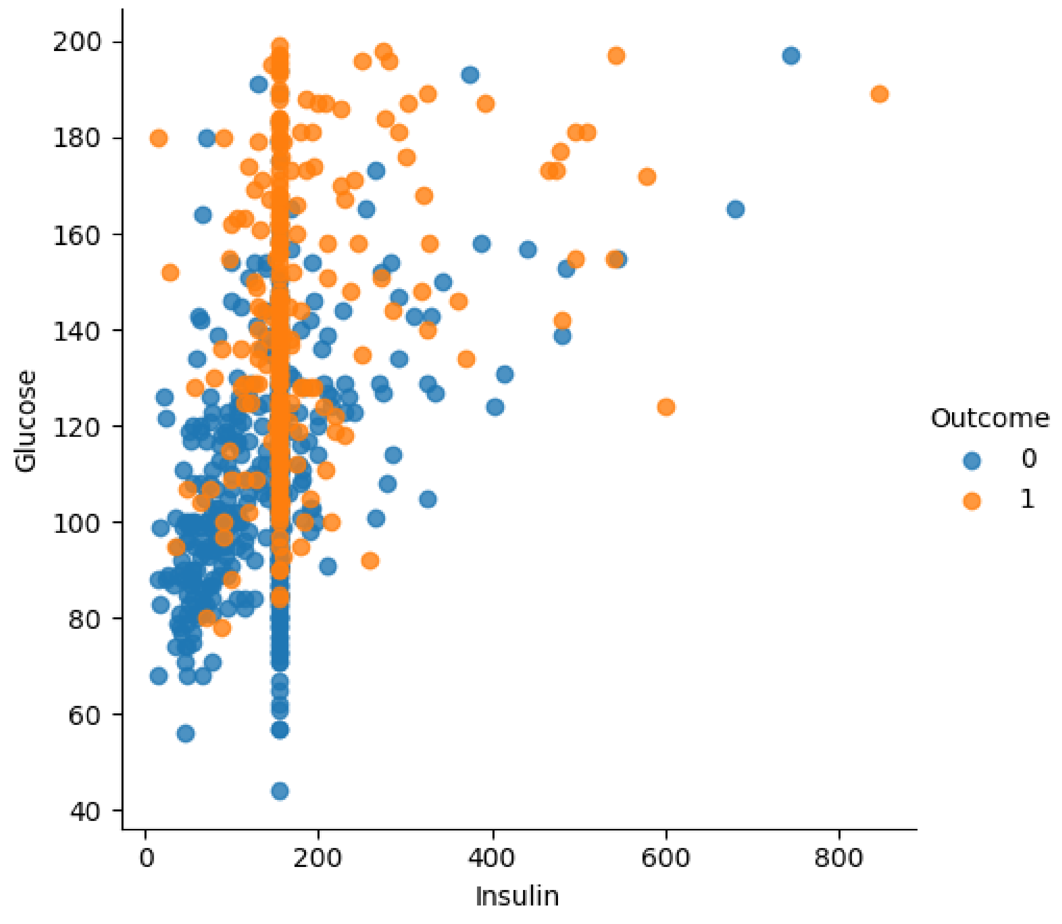
```
sns.lmplot(x='BMI',y='SkinThickness',data=df,fit_reg=False,hue='Outcome')
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae24fefe50>
```



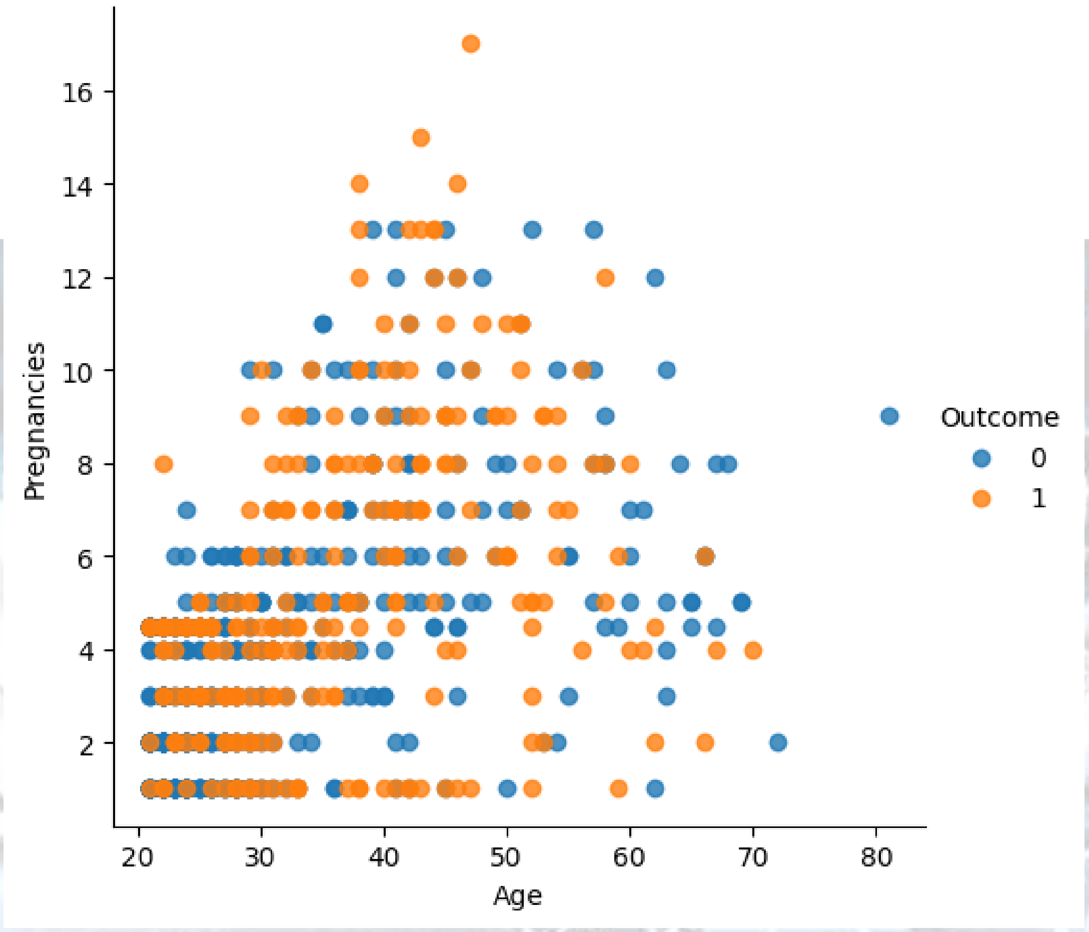
```
sns.lmplot(x='Insulin',y='Glucose',data=df,fit_reg=False,hue='Outcome')
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae27bcaac0>
```



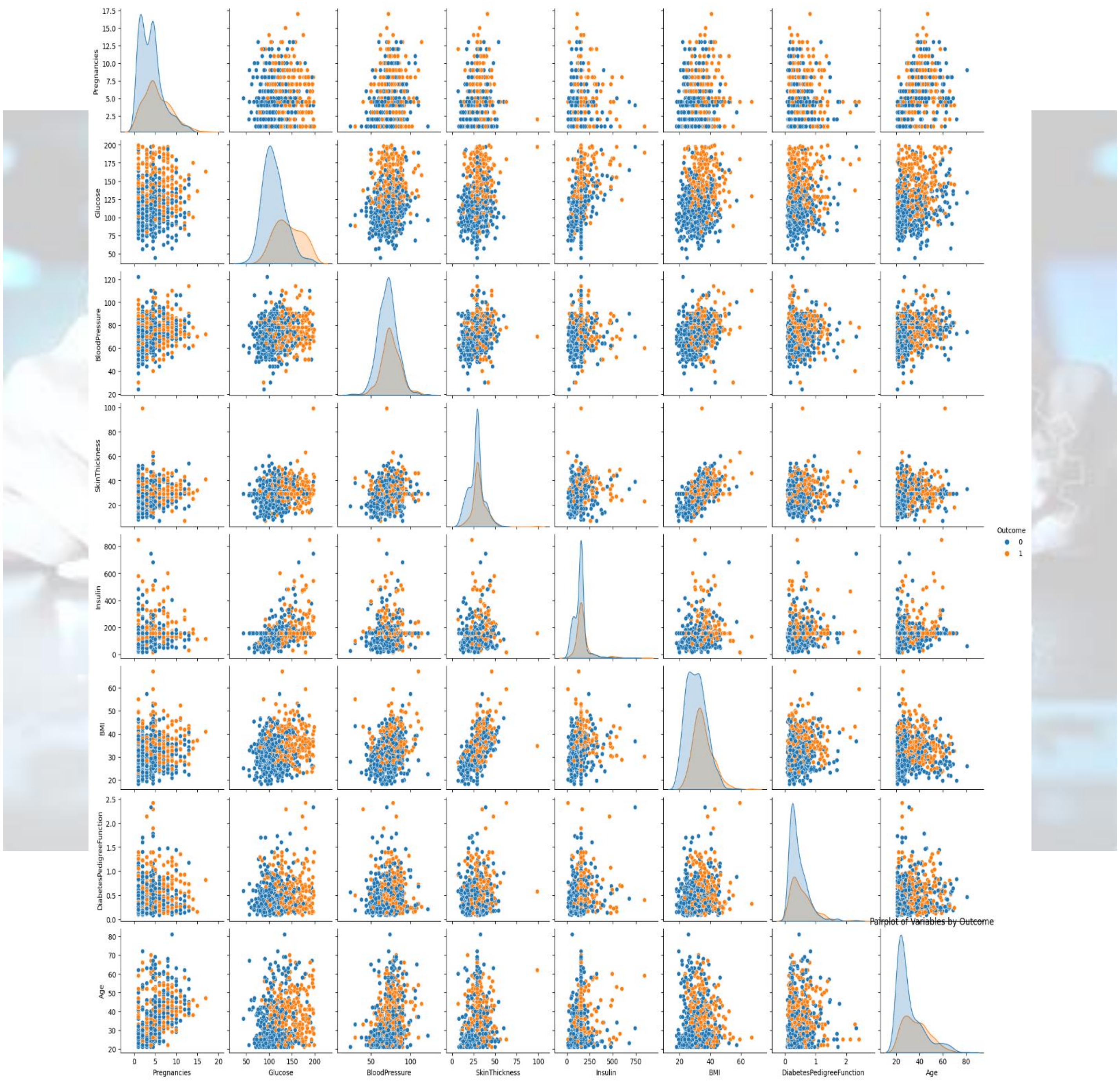
```
sns.lmplot(x='Age',y='Pregnancies',data=df,fit_reg=False,hue='Outcome')
```

```
<seaborn.axisgrid.FacetGrid at 0x1ae28027d00>
```



```
sns.pairplot(df, vars=["Pregnancies", "Glucose","BloodPressure","SkinThickness","Insulin", "BMI","DiabetesPedigreeFunction"
plt.title("Pairplot of Variables by Outcome")
```

```
Text(0.5, 1.0, 'Pairplot of Variables by Outcome')
```

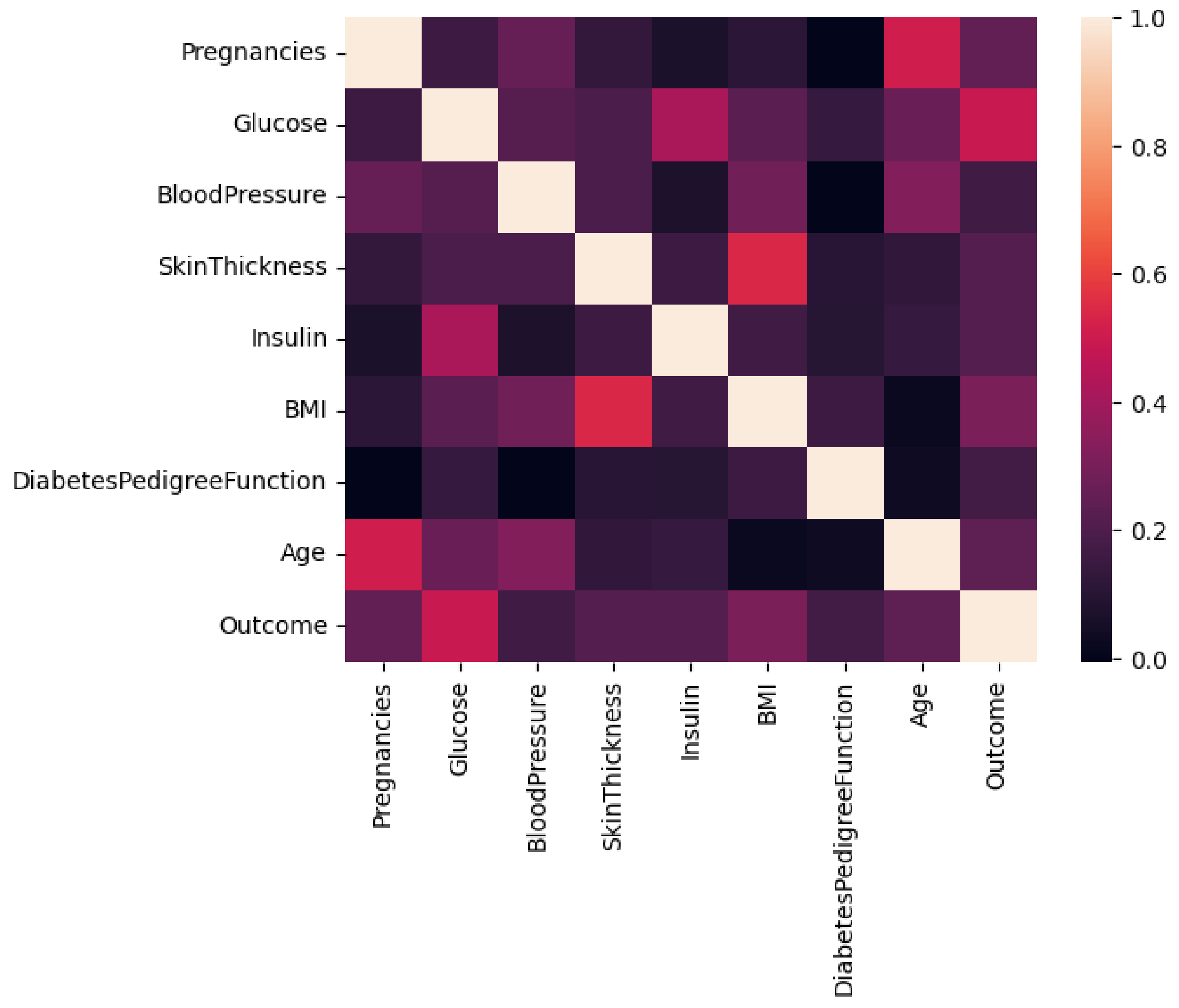


```
cor = df.corr()
cor
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.154290	0.259117	0.131819	0.068077	0.110590	-0.005658	0.511662	0.248263
Glucose	0.154290	1.000000	0.218367	0.192991	0.420157	0.230941	0.137060	0.266534	0.492928
BloodPressure	0.259117	0.218367	1.000000	0.192816	0.072517	0.281268	-0.002763	0.324595	0.166074
SkinThickness	0.131819	0.192991	0.192816	1.000000	0.158139	0.542398	0.100966	0.127872	0.215299
Insulin	0.068077	0.420157	0.072517	0.158139	1.000000	0.166586	0.098634	0.136734	0.214411
BMI	0.110590	0.230941	0.281268	0.542398	0.166586	1.000000	0.153400	0.025519	0.311924
DiabetesPedigreeFunction	-0.005658	0.137060	-0.002763	0.100966	0.098634	0.153400	1.000000	0.033561	0.173844
Age	0.511662	0.266534	0.324595	0.127872	0.136734	0.025519	0.033561	1.000000	0.238356
Outcome	0.248263	0.492928	0.166074	0.215299	0.214411	0.311924	0.173844	0.238356	1.000000

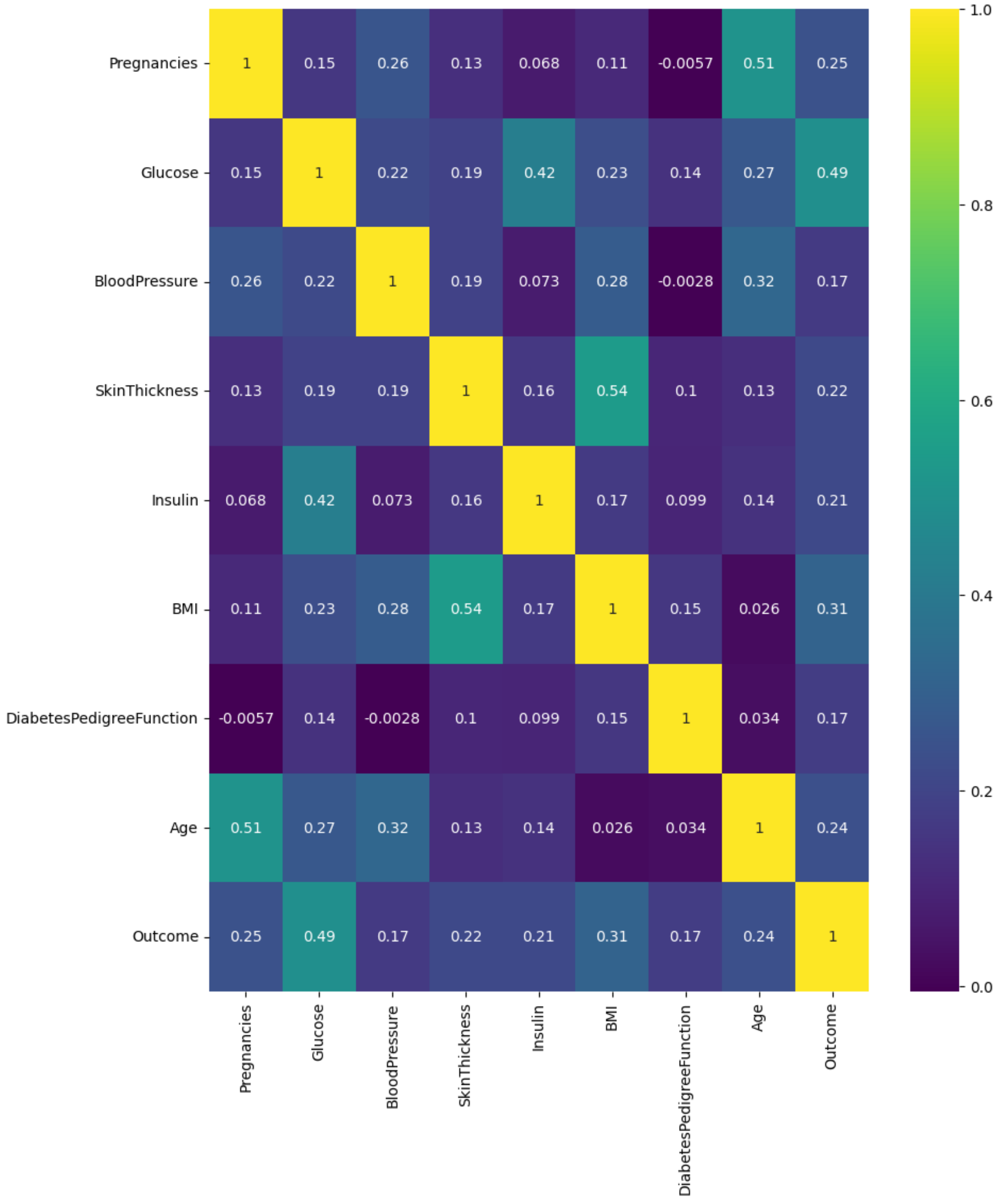
```
sns.heatmap(cor)
```

```
<Axes: >
```



```
plt.subplots(figsize=(10,12))
sns.heatmap(cor,annot=True,cmap='viridis')
```

```
<Axes: >
```

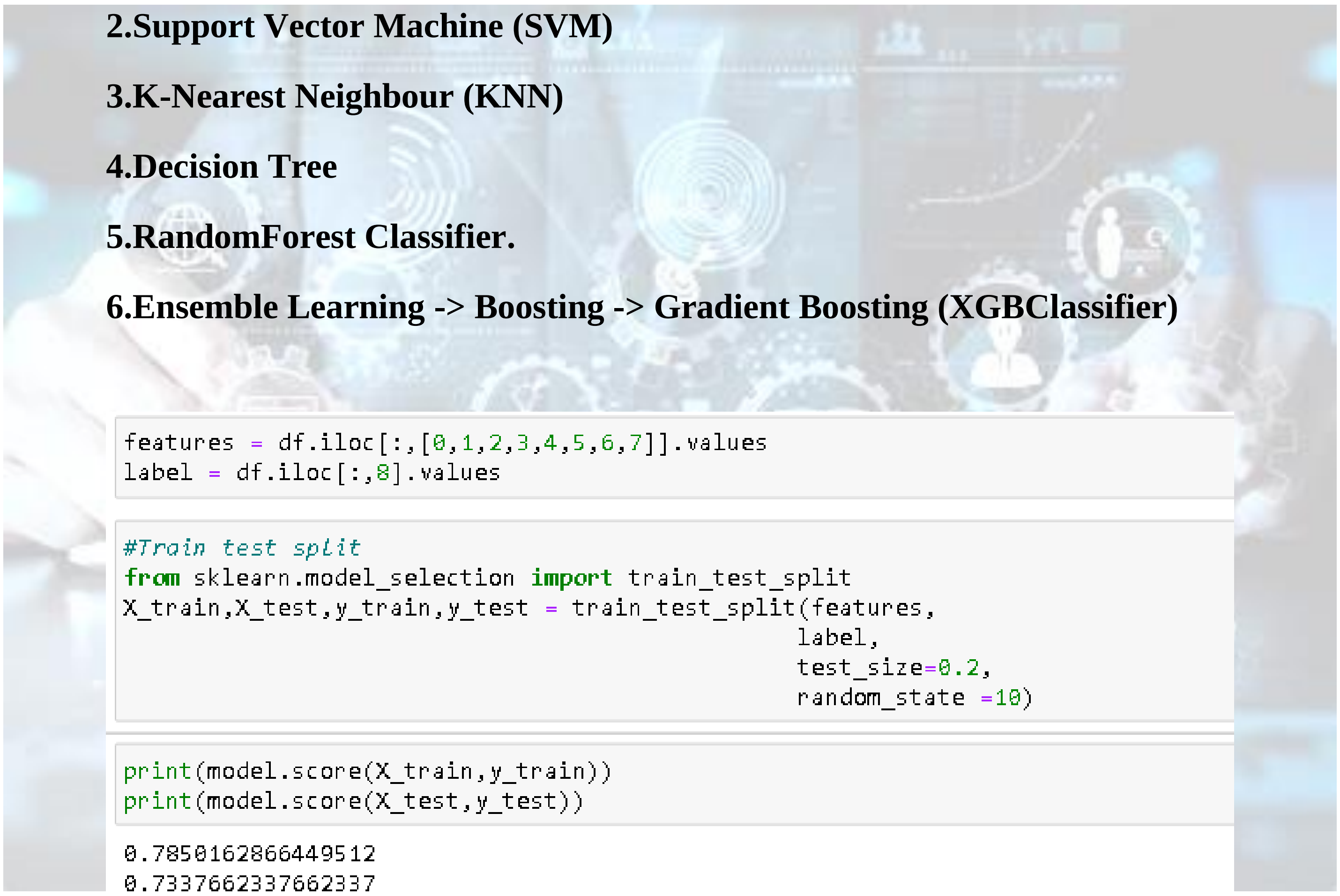


	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.15	0.26	0.13	0.068	0.11	-0.0057	0.51	0.25
Glucose	0.15	1	0.22	0.19	0.42	0.23	0.14	0.27	0.49
BloodPressure	0.26	0.22	1	0.19	0.073	0.28	-0.0028	0.32	0.17
SkinThickness	0.13	0.19	0.19	1	0.16	0.54	0.1	0.13	0.22
Insulin	0.068	0.42	0.073	0.16	1	0.17	0.099	0.14	0.21
BMI	0.11	0.23	0.28	0.54	0.17	1	0.15	0.026	0.31
DiabetesPedigreeFunction	-0.0057	0.14	-0.0028	0.1	0.099	0.15	1	0.034	0.17
Age	0.51	0.27	0.32	0.13	0.14	0.026	0.034	1	0.24
Outcome	0.25	0.49	0.17	0.22	0.21	0.31	0.17	0.24	1

Data Modelling:

Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.¶
Since it's a classification problem, we'll be building models using following classification algorithms for our training data and then compare performance of each model on test data to accurately predict target variable (Outcome):

1.Logistic Regression



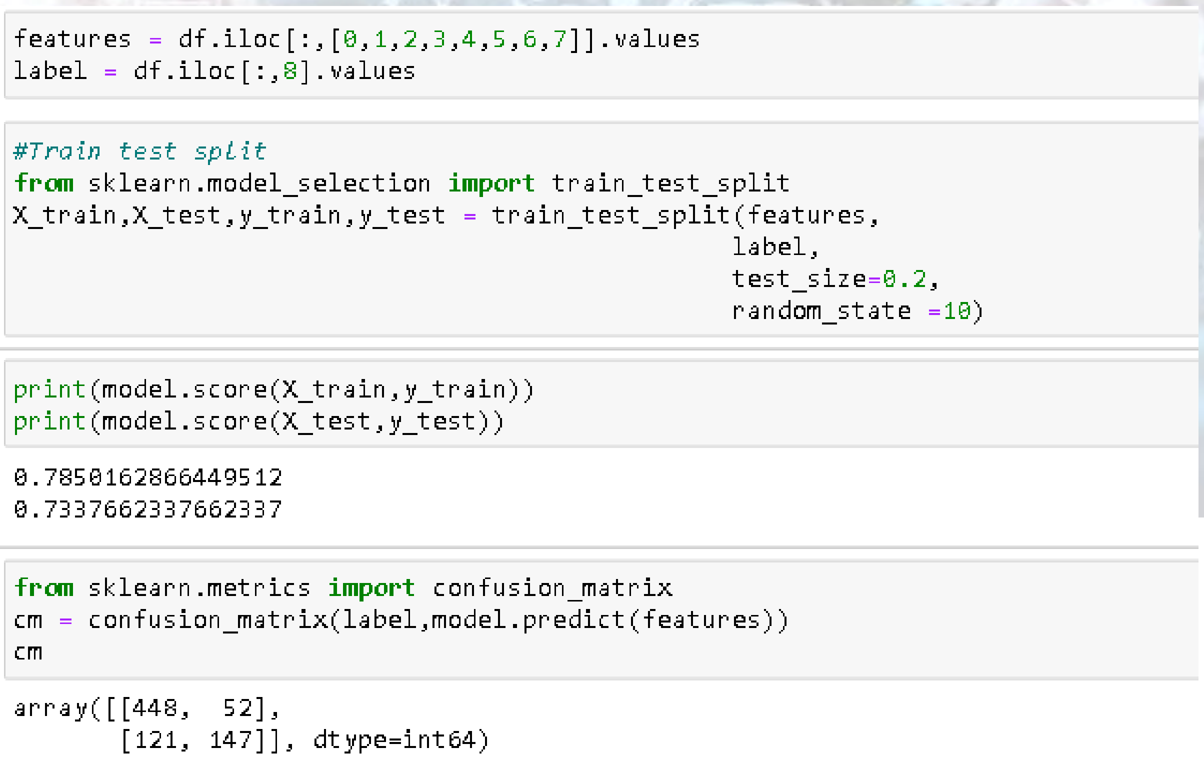
2.Support Vector Machine (SVM)

3.K-Nearest Neighbour (KNN)

4.Decision Tree

5.RandomForest Classifier.

6.Ensemble Learning -> Boosting -> Gradient Boosting (XGBClassifier)



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

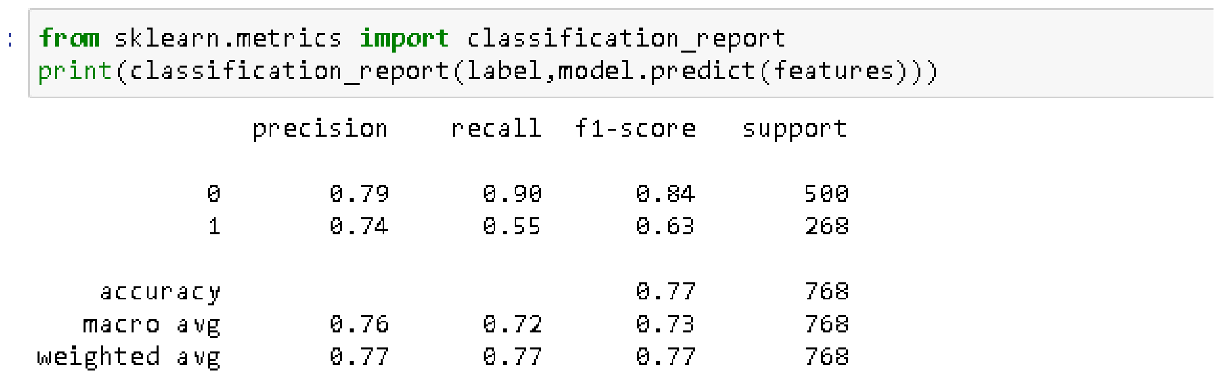
```
#Train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(features,
                                                 label,
                                                 test_size=0.2,
                                                 random_state =10)
```

```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7850162866449512
0.7337662337662337
```

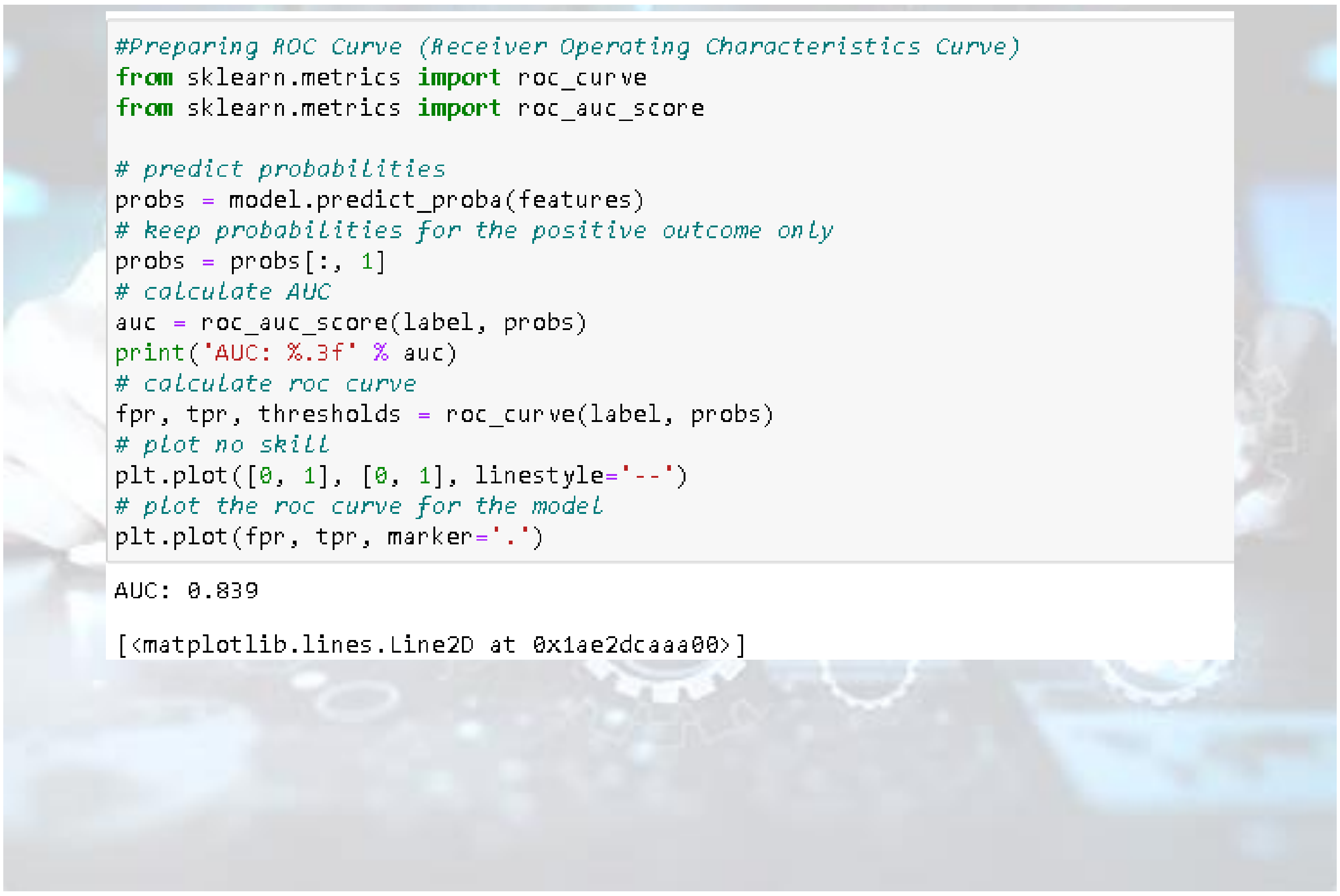
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

```
array([[448,  52],
       [121, 147]], dtype=int64)
```



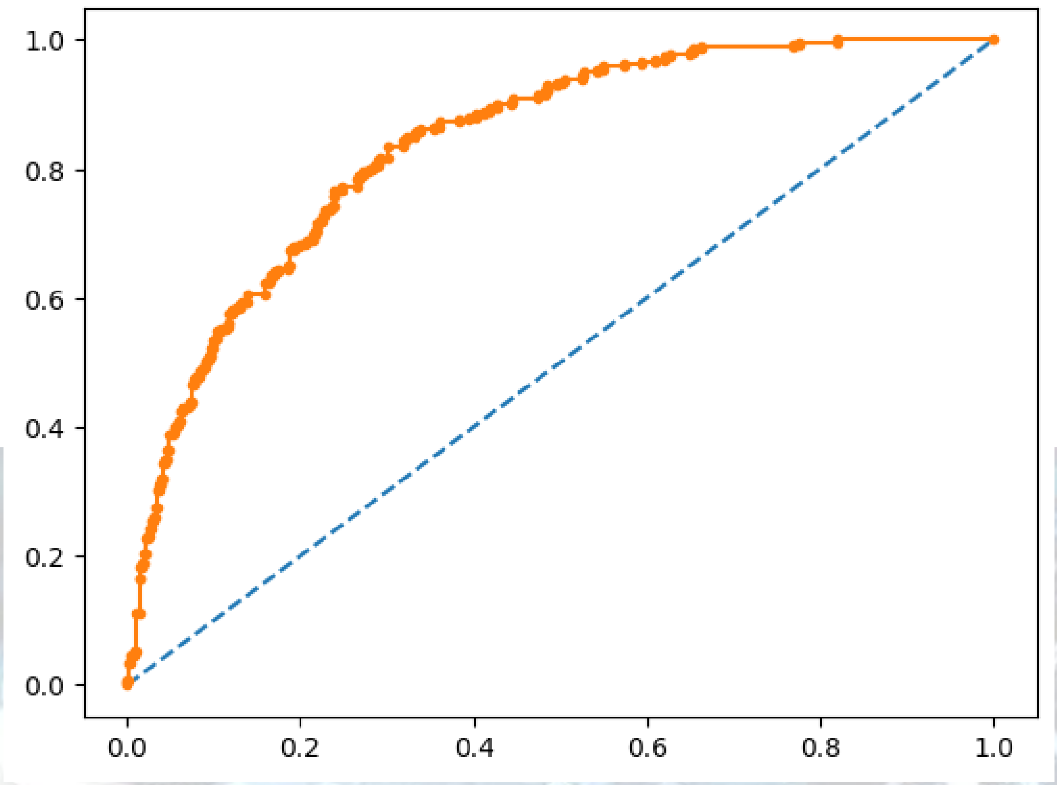
```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

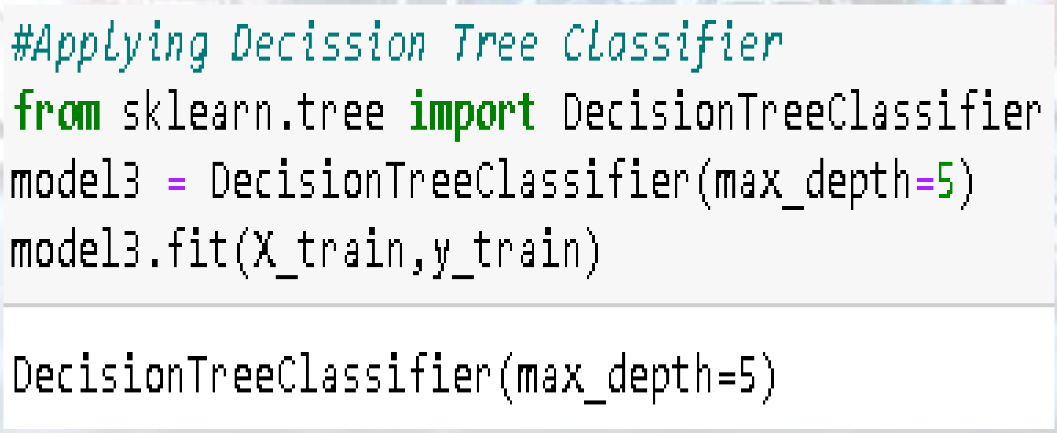
```
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       500
           1       0.74      0.55      0.63       268

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



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

# 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='.')
```

```
AUC: 0.839

[<matplotlib.lines.Line2D at 0x1ae2dcaaa00>]
```





```
#Applying Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=5)
```

```
model3.score(X_train,y_train)
```

```
0.8208469055374593
```

```
model3.score(X_test,y_test)
```

```
0.7532467532467533
```

```
#Applying Random Forest
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(X_train,y_train)
```

```
RandomForestClassifier(n_estimators=11)
```

```
model4.score(X_train,y_train)
```

```
0.993485342019544
```

```
model4.score(X_test,y_test)
```

```
0.7727272727272727
```

```
#Support Vector Classifier

from sklearn.svm import SVC
model5 = SVC(kernel='rbf',
            gamma='auto')
model5.fit(X_train,y_train)
```

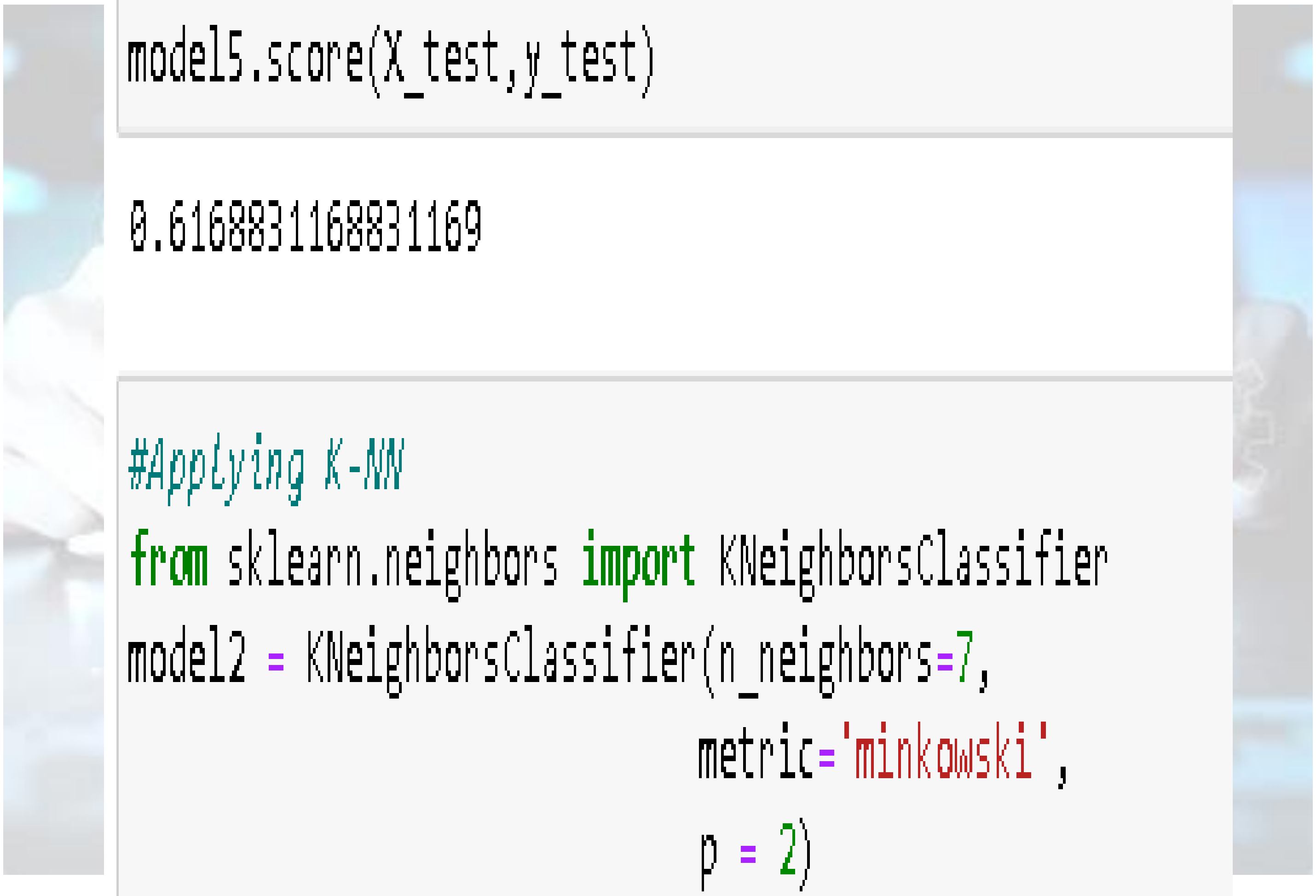
```
SVC(gamma='auto')
```

```
model5.score(X_test,y_test)
```

```
0.6168831168831169
```

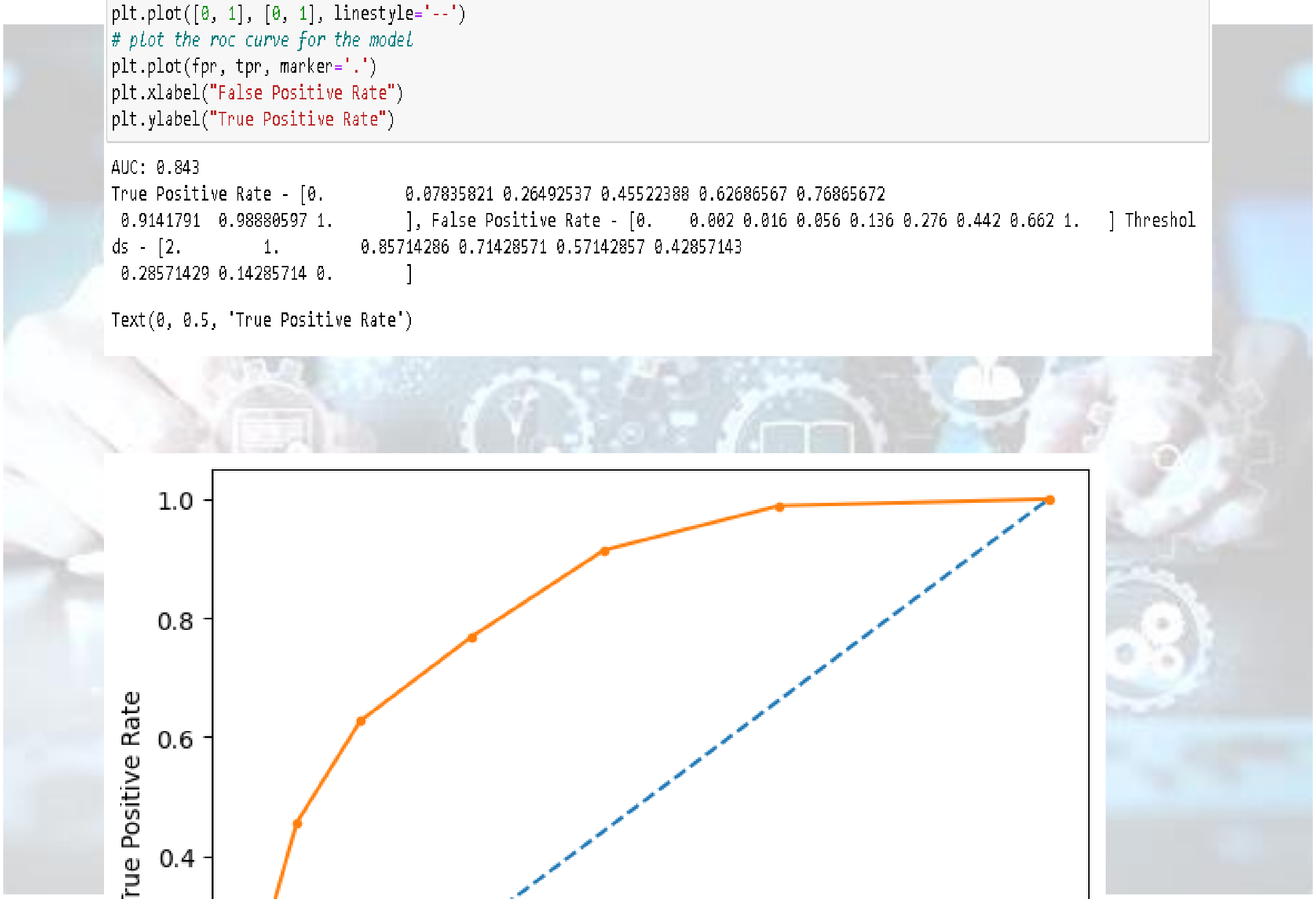
```
model5.score(X_test,y_test)
```

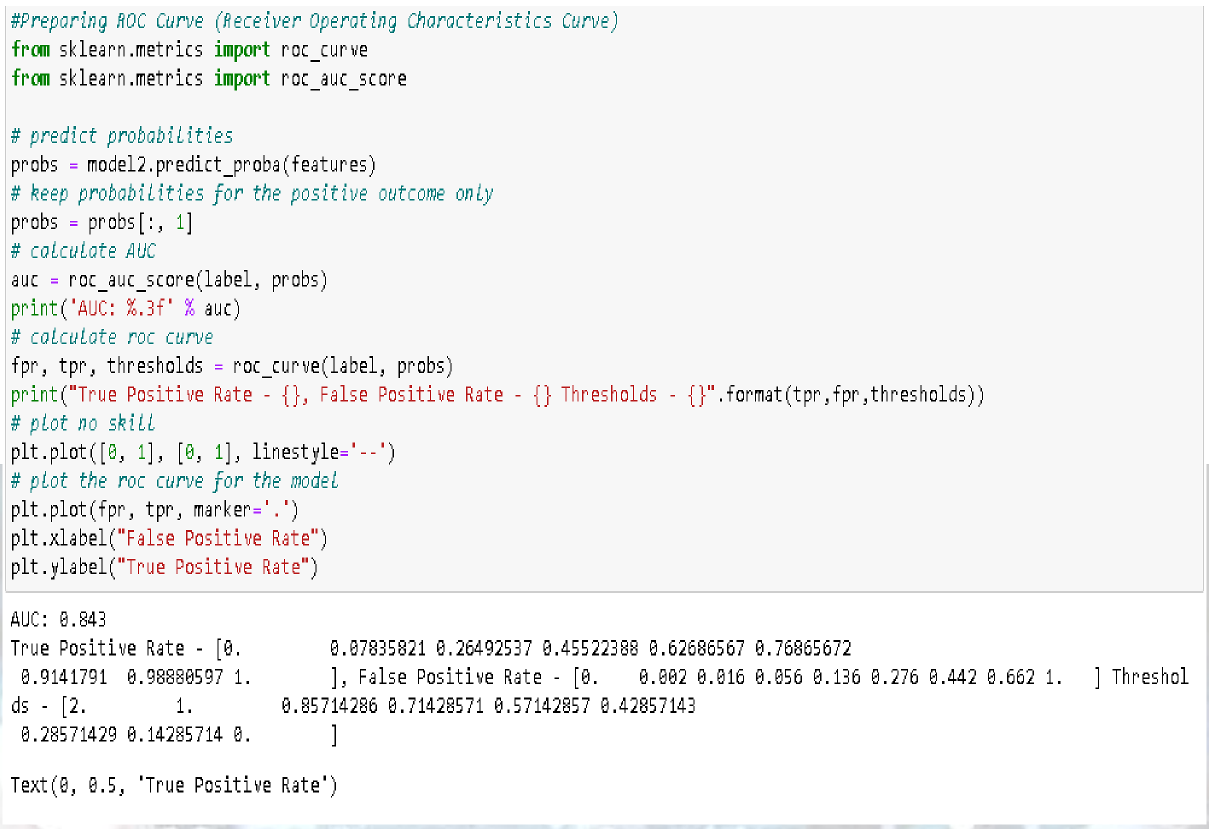
```
0.6168831168831169
```



```
#Applying K-NN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=7,
                              metric='minkowski',
                              p = 2)
model2.fit(X_train,y_train)
```

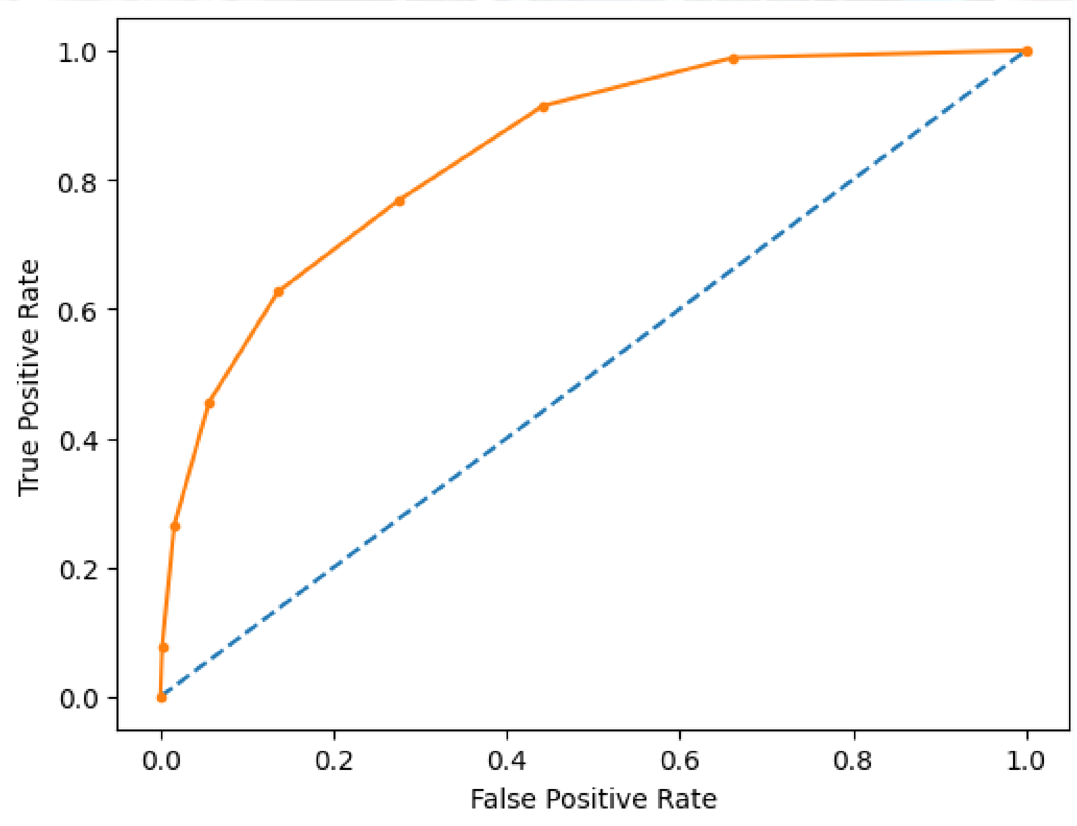
```
KNeighborsClassifier(n_neighbors=7)
```





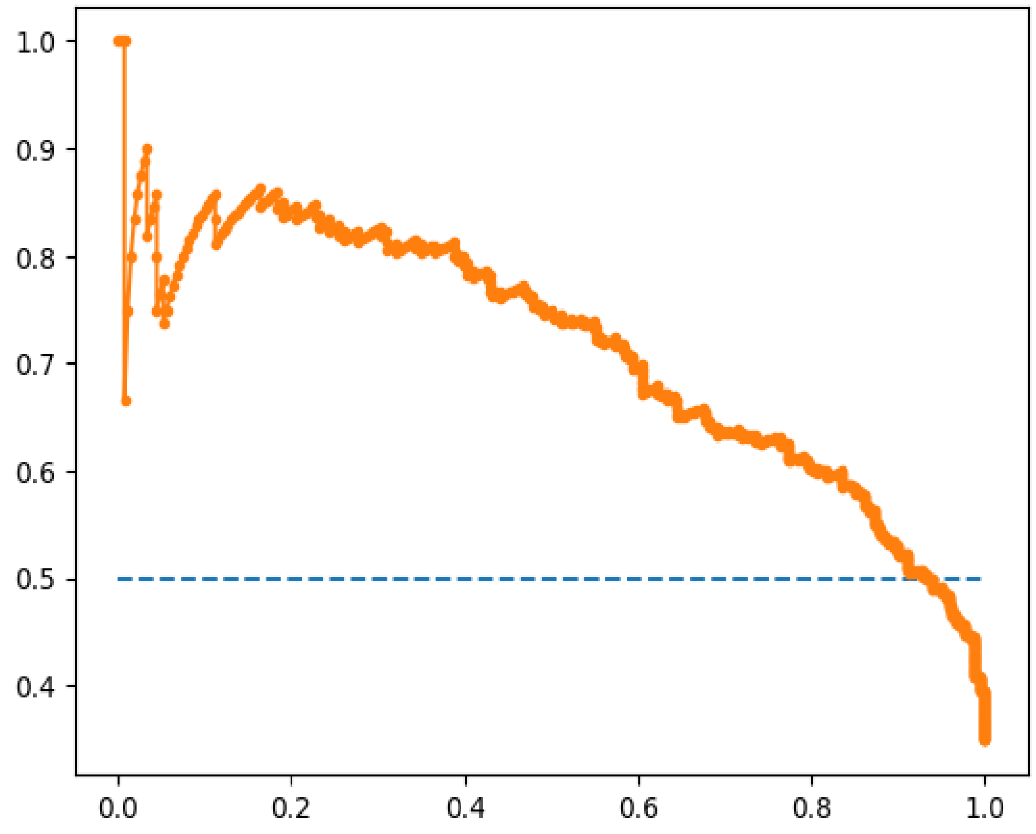
```python
#Preparing ROC Curve (Receiver Operating Characteristics Curve)
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

# predict probabilities
probs = model2.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 - {}".format(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: 0.843
True Positive Rate - [0.         0.07835821 0.26492537 0.45522388 0.62686567 0.76865672
 0.9141791  0.98880597 1.        ], False Positive Rate - [0.    0.002 0.016 0.056 0.136 0.276 0.442 0.662 1.   ] Threshol
ds - [2.         1.         0.85714286 0.71428571 0.57142857 0.42857143
 0.28571429 0.14285714 0.        ]

Text(0, 0.5, 'True Positive Rate')
```



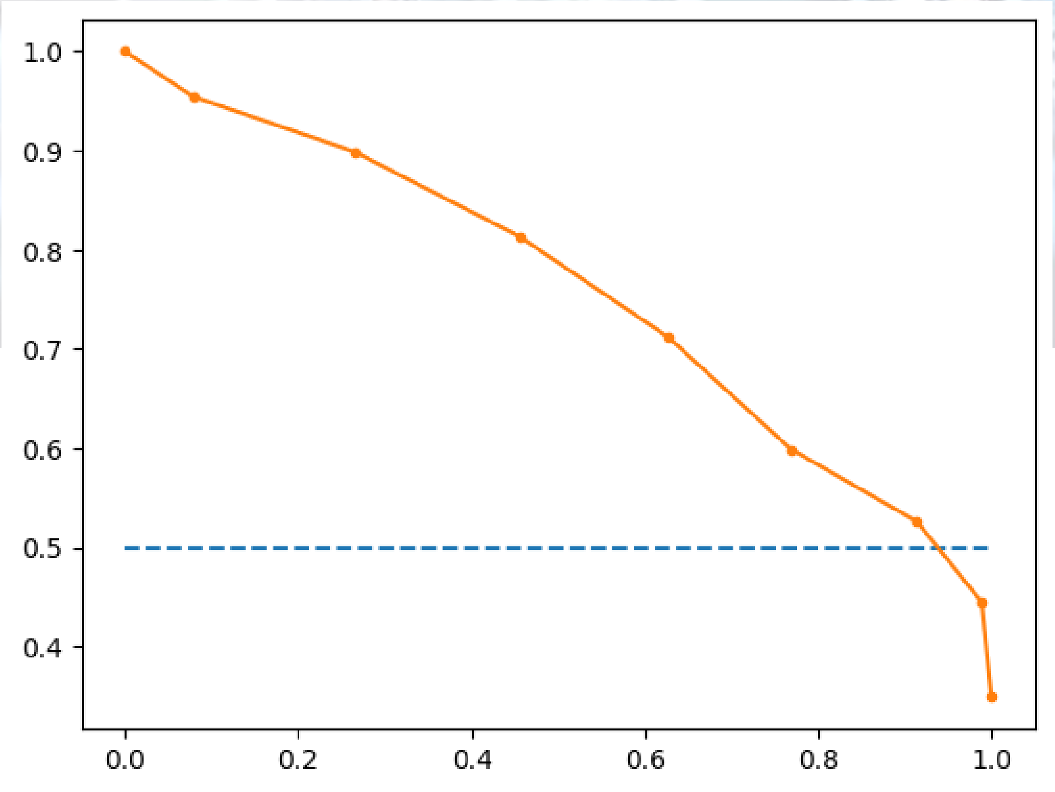
```
#Precision Recall Curve for Logistic Regression

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.630 auc=0.713 ap=0.715

[<matplotlib.lines.Line2D at 0x1ae2dd80ac0>]
```



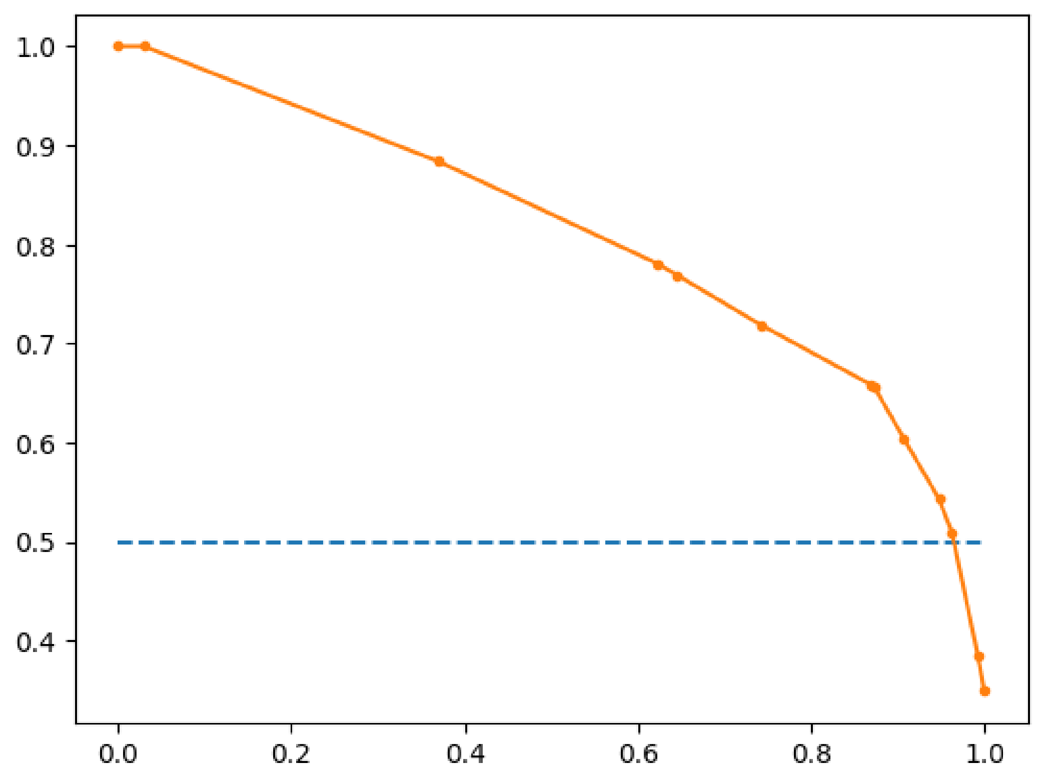
```
#Precision Recall Curve for KNN

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model2.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model2.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.667 auc=0.759 ap=0.718
```



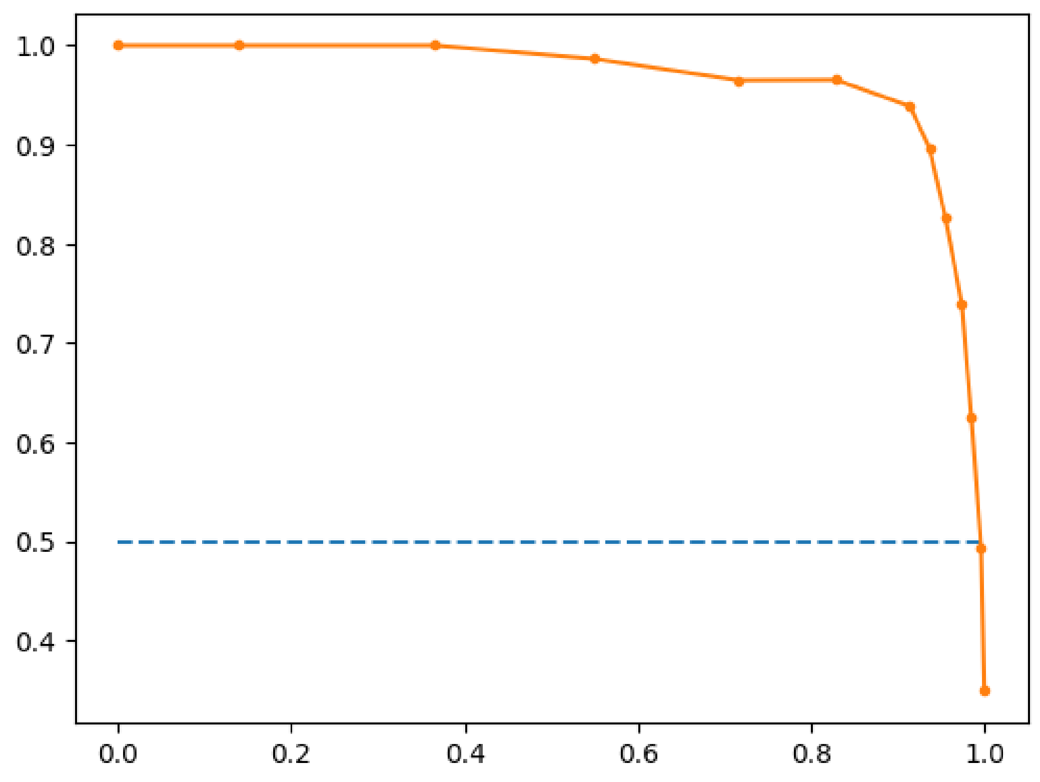
```
#Precision Recall Curve for Decission Tree Classifier

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model3.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model3.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

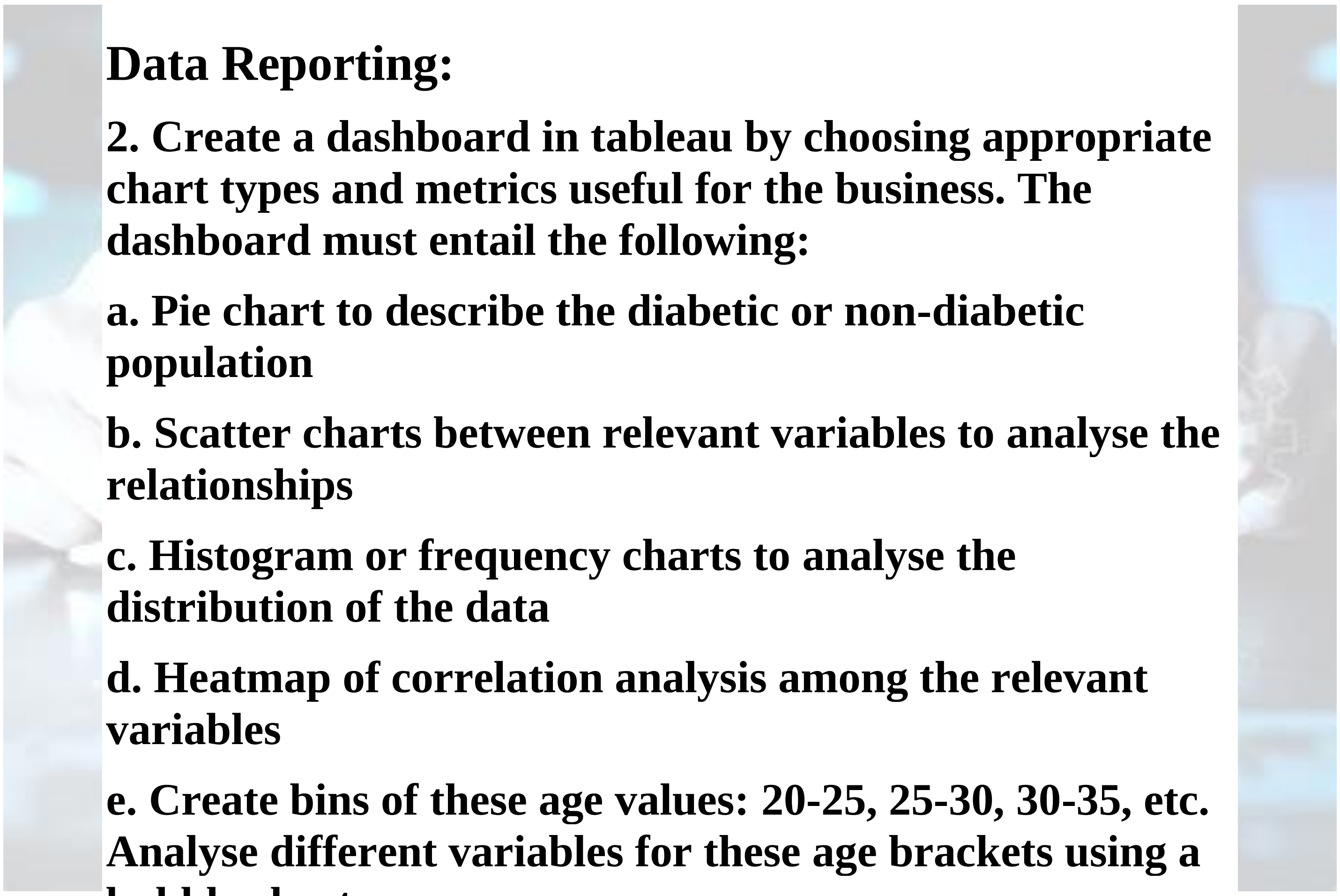
```
f1=0.693 auc=0.809 ap=0.765

[<matplotlib.lines.Line2D at 0x1ae2de7b6a0>]
```



```
#Precision Recall Curve for Random Forest

from sklearn.metrics import precision_recall_curve
from sklearn.metrics import f1_score
from sklearn.metrics import auc
from sklearn.metrics import average_precision_score
# predict probabilities
probs = model4.predict_proba(features)
# keep probabilities for the positive outcome only
probs = probs[:, 1]
# predict class values
yhat = model4.predict(features)
# calculate precision-recall curve
precision, recall, thresholds = precision_recall_curve(label, probs)
# calculate F1 score
f1 = f1_score(label, yhat)
# calculate precision-recall AUC
auc = auc(recall, precision)
# calculate average precision score
ap = average_precision_score(label, probs)
print('f1=%.3f auc=%.3f ap=%.3f' % (f1, auc, ap))
# plot no skill
plt.plot([0, 1], [0.5, 0.5], linestyle='--')
# plot the precision-recall curve for the model
plt.plot(recall, precision, marker='.')
```

```
f1=0.926 auc=0.968 ap=0.960

[<matplotlib.lines.Line2D at 0x1ae2deb27c0>]
```





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

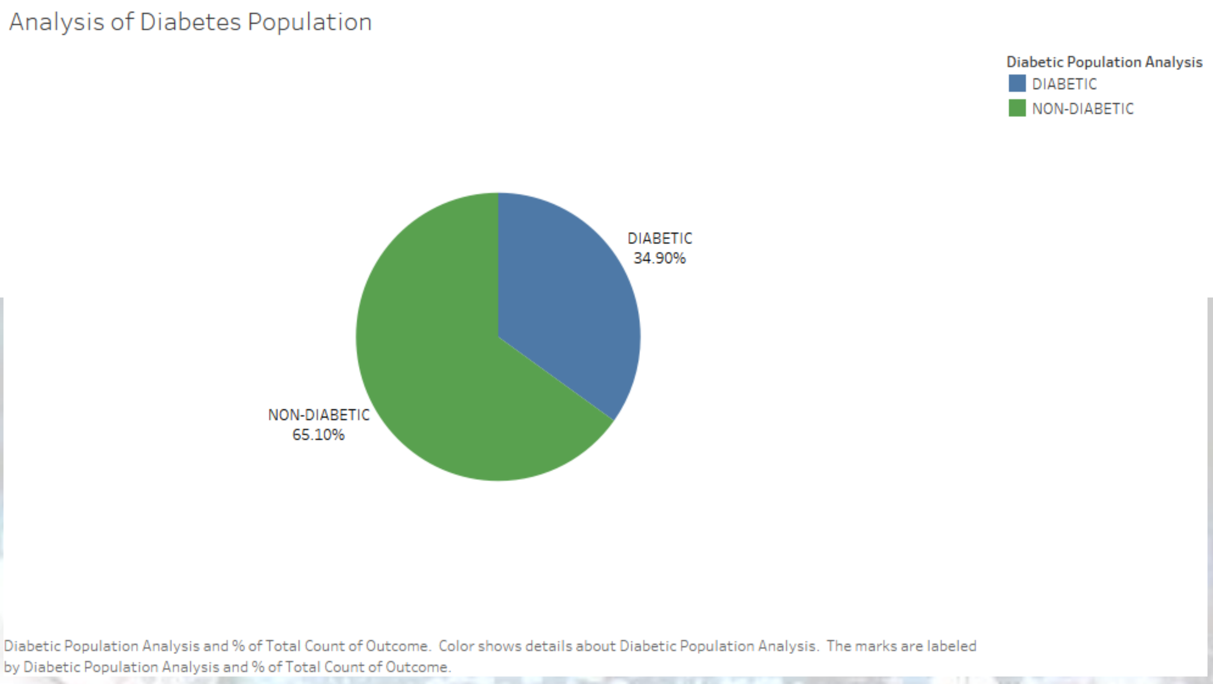
b. Scatter charts between relevant variables to analyse the relationships

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

d. Heatmap of correlation analysis among the relevant variables

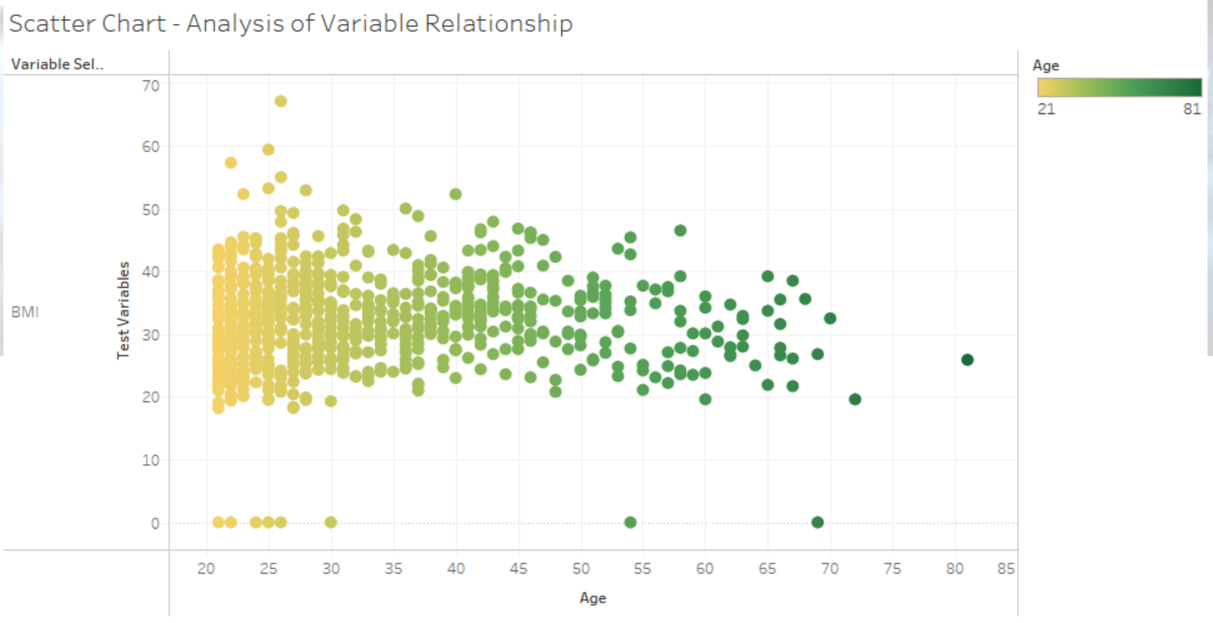
e. Create bins of these age values: 20-25, 25-30, 30-35, etc. Analyse different variables for these age brackets using a bubble chart.

Analysis of Diabetes Population:



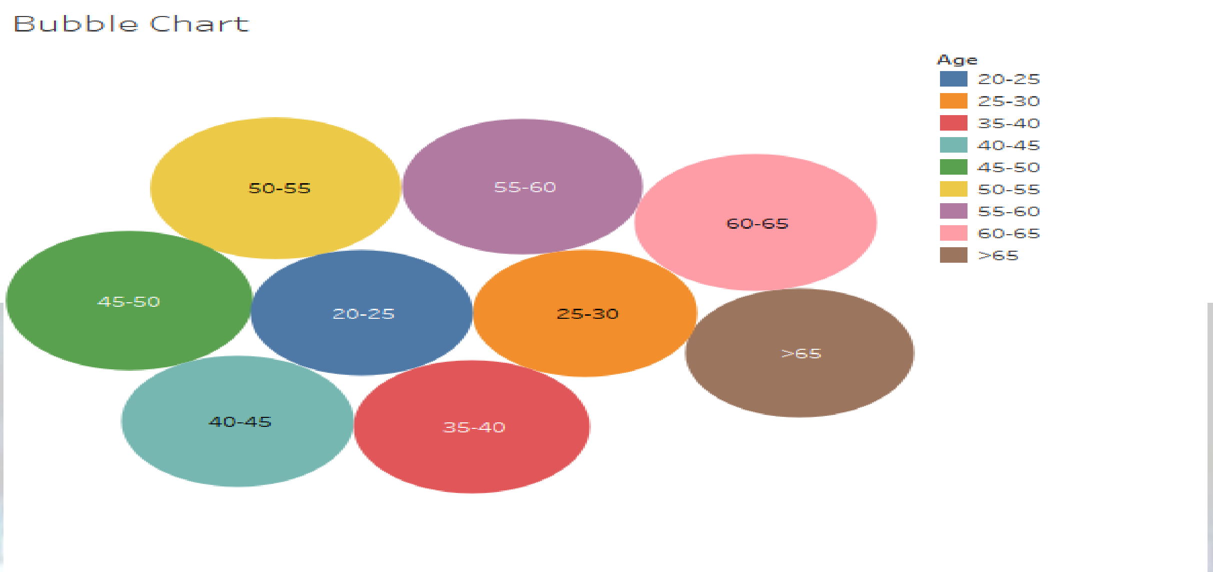
Diabetic Population Analysis and % of Total Count of Outcome. Color shows details about Diabetic Population Analysis. The marks are labeled by Diabetic Population Analysis and % of Total Count of Outcome.

Analysis of Variable Relationship -Scatter chart



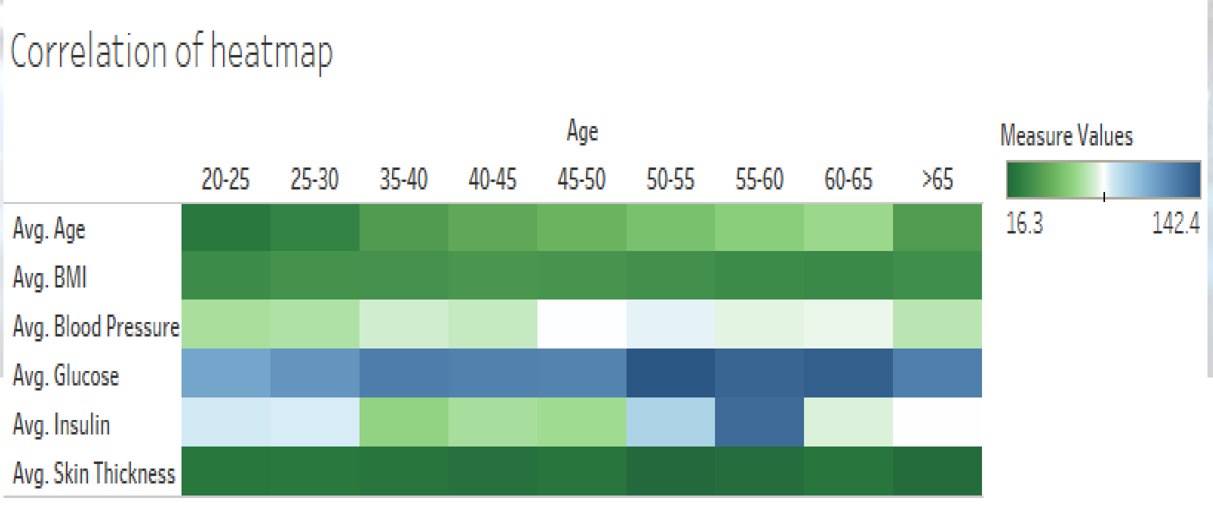
Age vs. Test Variables broken down by Variable Selector. Color shows details about Age.

Bubble Chart



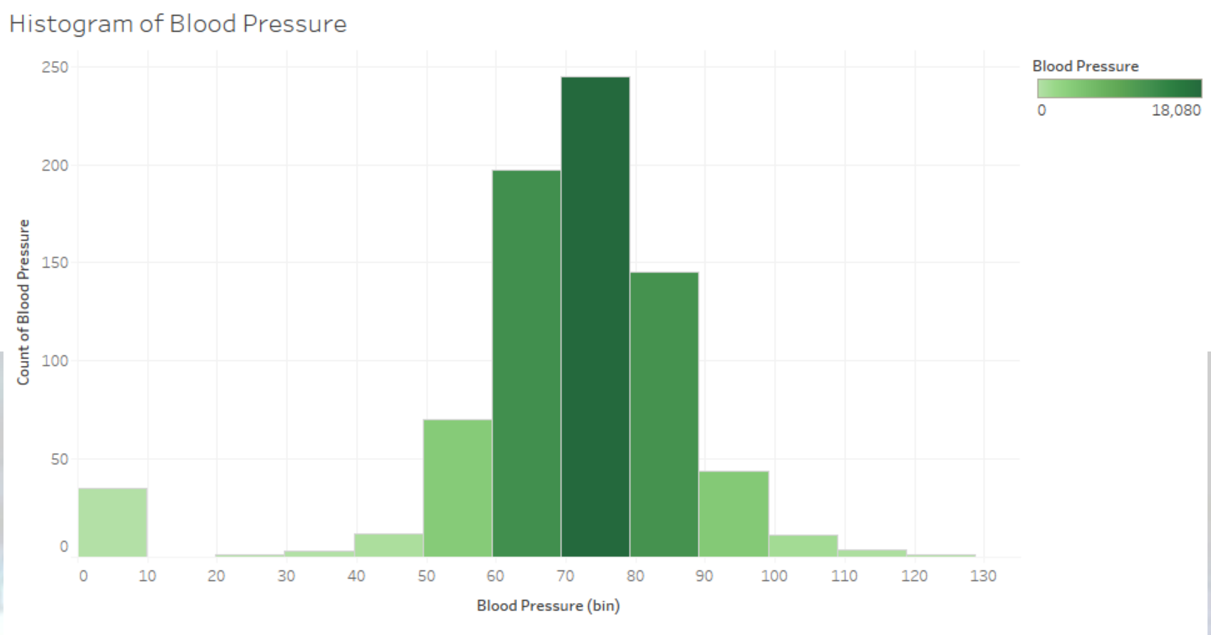
Age . Color shows details about Age . Size shows average of Blood Pressure. The marks are labeled by Age .

Correlation of Heatmap



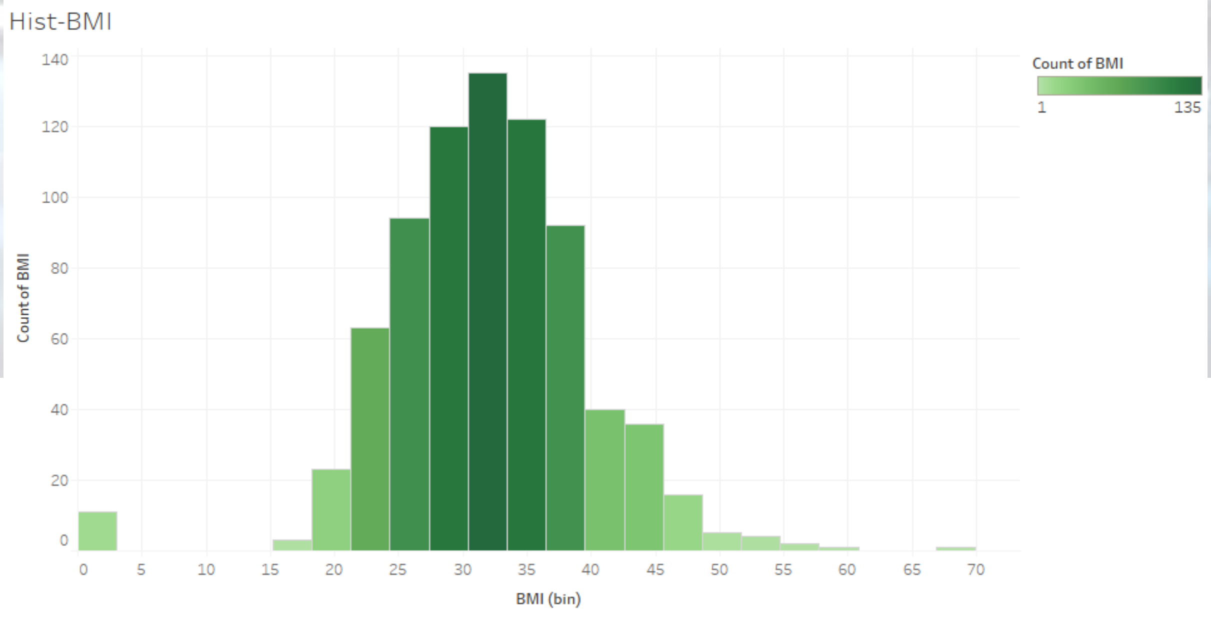
Avg. Age, Avg. Blood Pressure, Avg. BMI, Avg. Glucose, Avg. Insulin and Avg. Skin Thickness (color) broken down by Age . The view is filtered on Age , which keeps 9 of 9 members.

Histogram Of Blood Pressure



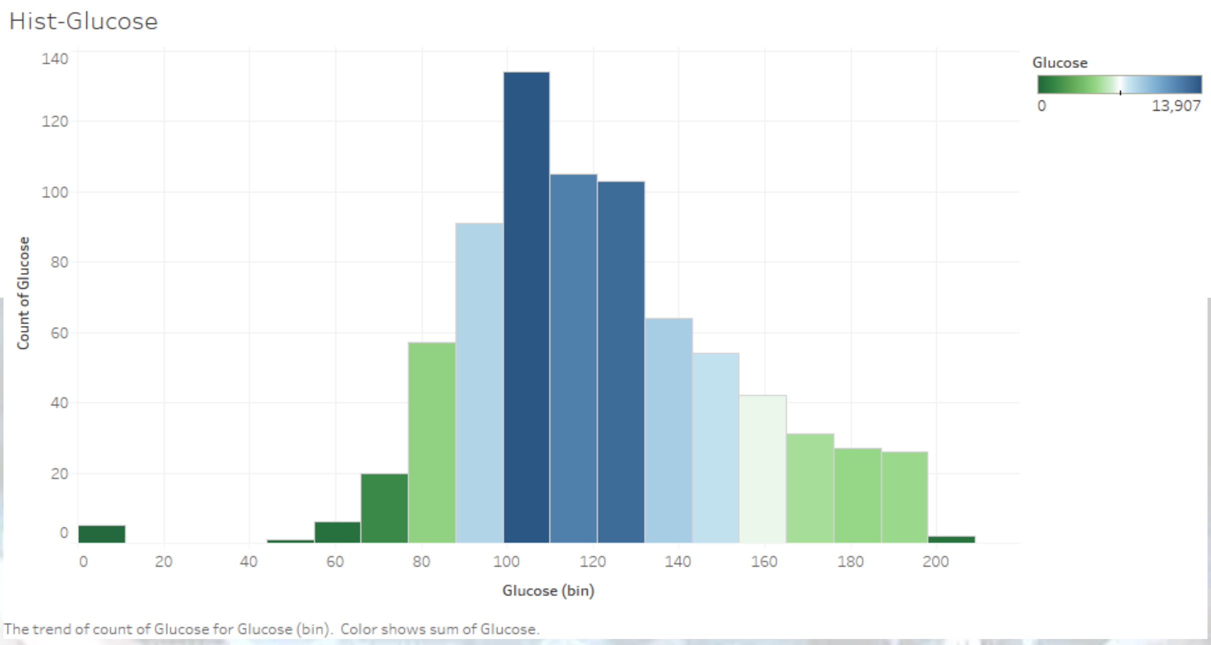
The trend of count of Blood Pressure for Blood Pressure (bin). Color shows sum of Blood Pressure.

Histogram Of BMI



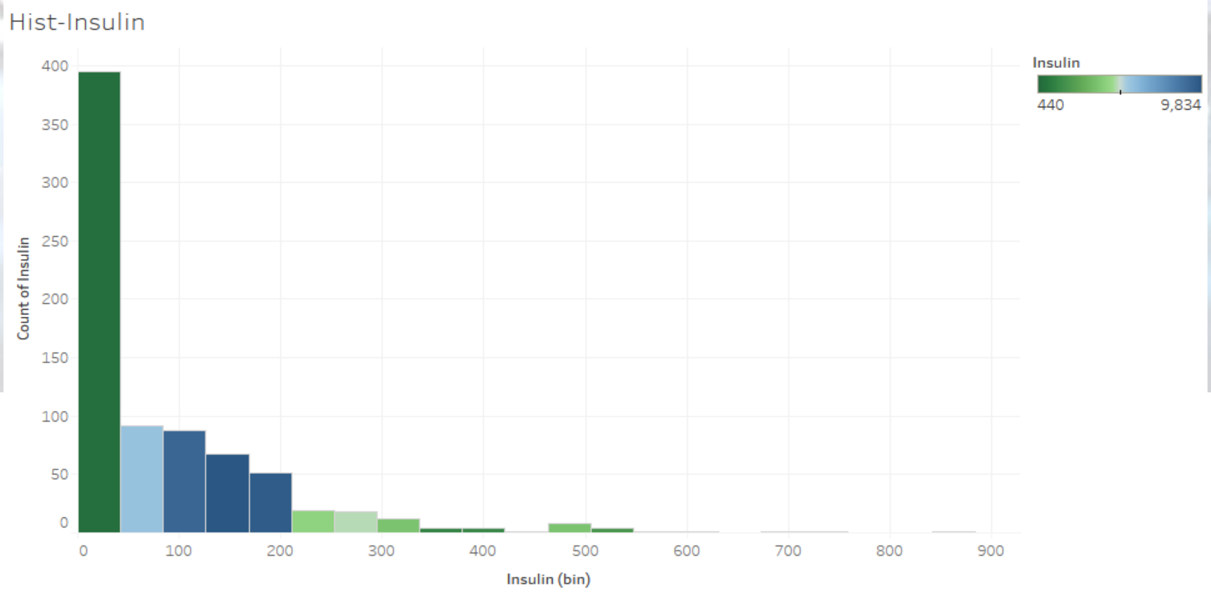
The trend of count of BMI for BMI (bin). Color shows count of BMI.

Histogram Of Glucose



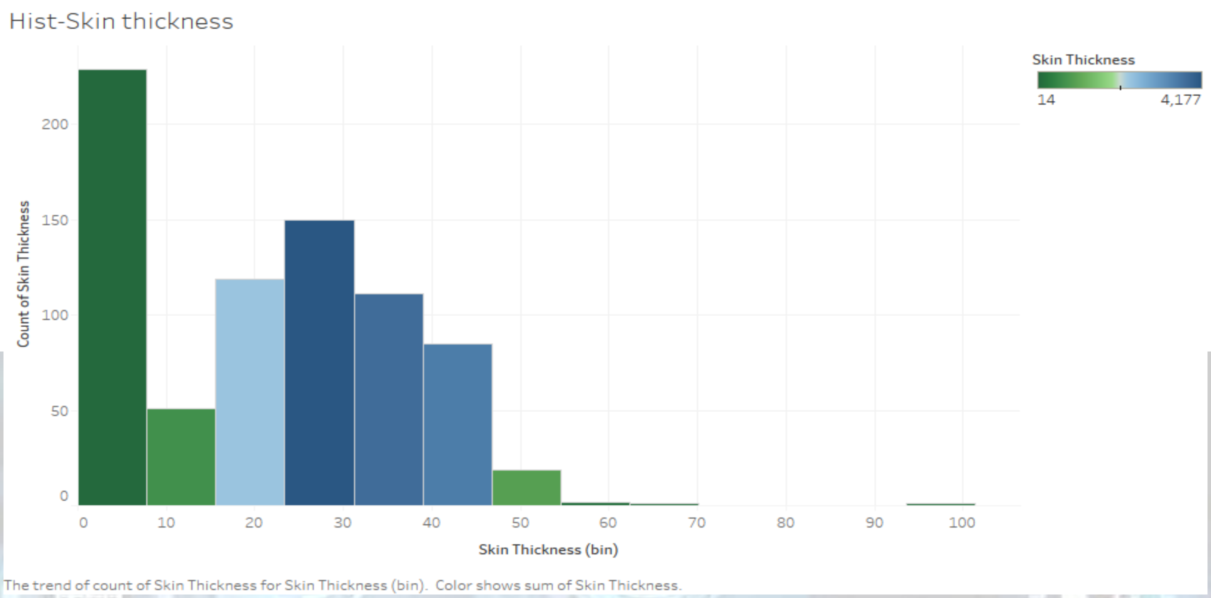
The trend of count of Glucose for Glucose (bin). Color shows sum of Glucose.

Histogram Of Insulin



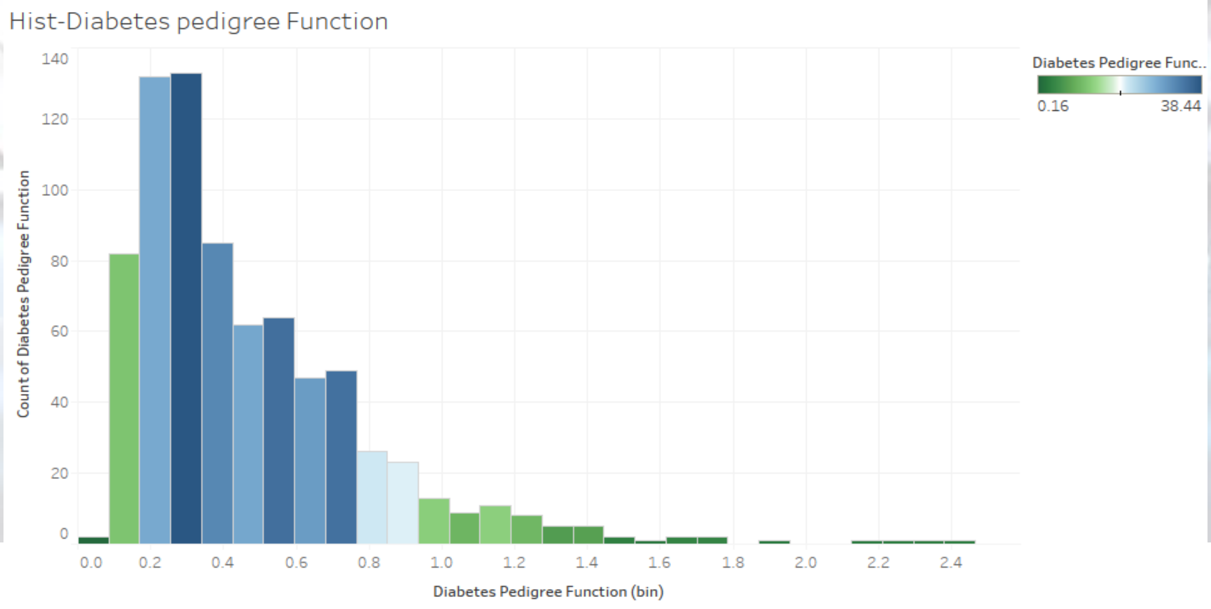
The trend of count of Insulin for Insulin (bin). Color shows sum of Insulin.

Histogram of Skin Thickness



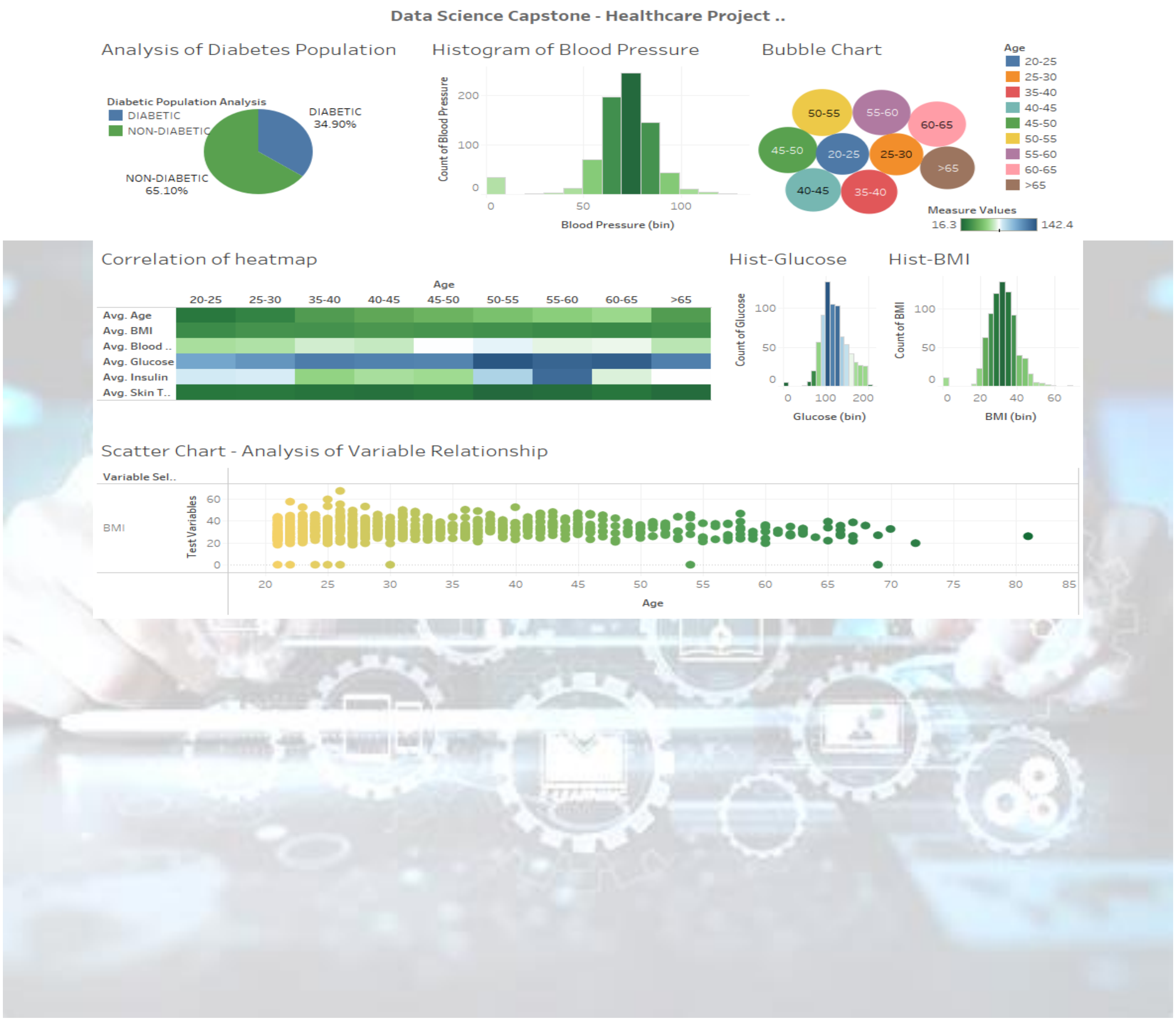
The trend of count of Skin Thickness for Skin Thickness (bin). Color shows sum of Skin Thickness.

Histogram of Diabetes Pedigree Function



The trend of count of Diabetes Pedigree Function for Diabetes Pedigree Function (bin). Color shows sum of Diabetes Pedigree Function.

DASH BOARD :


Data Science Capstone - Healthcare Project ..
Analysis of Diabetes Population
Diabetic Population Analysis
DIABETIC
NON-DIABETIC
DIABETIC 34.90%
NON-DIABETIC 65.10%
Histogram of Blood Pressure
Count of Blood Pressure
200
100
0
0
50
100
Blood Pressure (bin)
Bubble Chart
50-55
55-60
60-65
45-50
20-25
25-30
>65
40-45
35-40
Age
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
>65
Measure Values
16.3
142.4
Correlation of heatmap
Age
20-25
25-30
35-40
40-45
45-50
50-55
55-60
60-65
>65
Avg. Age
Avg. BMI
Avg. Blood ..
Avg. Glucose
Avg. Insulin
Avg. Skin T..
Hist-Glucose
Count of Glucose
100
50
0
0
100
200
Glucose (bin)
Hist-BMI
Count of BMI
100
50
0
0
20
40
60
BMI (bin)
Scatter Chart - Analysis of Variable Relationship
Variable Sel..
BMI
Test Variables
60
40
20
0
20
25
30
35
40
45
50
55
60
65
70
75
80
85
Age

Thank You