**ASSIGNMENT 10**

**Source Code:**

import pandas as pd

importmatplotlib.pyplot as plt

frompandas.plotting import scatter\_matrix

fromsklearn.model\_selection import train\_test\_split

fromsklearn.naive\_bayes import GaussianNB

fromsklearn.metrics import confusion\_matrix

fromsklearn.metrics import precision\_recall\_fscore\_support

fromsklearn.preprocessing import StandardScaler

df=pd.read\_csv("diabetes.csv")

print("----------------------------------------------Dataset Values-------------------------------------------------")

print(df.head(5))

print("----------------------------------------------Dataset Values-------------------------------------------------")

print(df.tail(5))

x=df.iloc[:,:-1]

y=df.iloc[:,-1]

#x=x.values() Not applicable on ndarrays

#y=y.values()

print("--------------------------------------------------X values-----------------------------------------------------")

#independent

print(x.head(5))

print("--------------------------------------------------Y values-----------------------------------------------------")

#dependent

print(y.head(5))

print("--------------------------------------------------------------------------------------------------------")

print(x.isnull().any())

print("--------------------------------------------------------------------------------------------------------")

print(y.isnull().any())

print("--------------------------------------------------------------------------------------------------------")

print("Information about X dataframe: ")

print(x.info())

#print(y.info()) Gives error because info() is not avaiable on series

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.33,random\_state=0)

#random\_state=0 means that every time you will get the same output as when you had made the split; also called "seed"

#if we write, random\_state=any\_no then the results will vary but would be same for the same number anytime.

print("---------------------------------------------x\_train--------------------------------------------------------")

print(x\_train.head(5))

print("---------------------------------------------x\_test---------------------------------------------------------")

print(x\_test.head(5))

print("---------------------------------------------y\_train--------------------------------------------------------")

print(y\_train.head(5))

print("---------------------------------------------y\_test---------------------------------------------------------")

print(y\_test.head(5))

df.hist(figsize=(20,30))

plt.show()

df.boxplot(figsize=(20,30))

plt.show()

df.plot(kind='hist', subplots=True, layout=(3,3), sharex=False, sharey=False,title="Pima Indian Diabetes dataset Histogram",figsize=(20,30))

plt.show()

df.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False,title="Pima Indian Diabetes dataset Boxplot",figsize=(20,30))

plt.show()

scatter\_matrix(df,figsize=(20,30))

plt.show()

#Scaling(Preprocessing)

scalar\_x = StandardScaler()

x\_train = scalar\_x.fit\_transform(x\_train)

x\_test = scalar\_x.transform(x\_test) #dont use fit\_transform as we want same std dev as whole dataset

#Training

classifier=GaussianNB()

classifier.fit(x\_train,y\_train) #train dataset

#Testing

y\_pred=classifier.predict(x\_test) #testing

cm=confusion\_matrix(y\_test,y\_pred)

print("confusion matrix:\n",cm)

tp = cm[0][0]

tn = cm[1][1]

fp = cm[1][0]

fn = cm[0][1]

correct\_predictions = tp + tn

print("\nCorrect Predictions")

print(correct\_predictions)

total\_predictions = correct\_predictions + fp + fn

print("\nTotal Predictions")

print(total\_predictions)

'''accuracy = (correct\_predictions/total\_predictions)\*100

print("\nAccuracy")

print(accuracy)

'''

precision = tp/(tp + fp)

print("\nPrecision")

print(precision)

recall = tp/(tp + fn)

print("\nRecall")

print(recall)

f\_measure = (2\*precision\*recall)/(precision+recall)

print("\nF-Measure")

print(f\_measure)

accuracy = ((tp+tn)/(tp+tn+fp+fn))

print("\nAccuracy:\n",accuracy\*100)

print("\nError\n",(1-accuracy)\*100)

####To calculate using package

print("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculated values using package\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

pf = precision\_recall\_fscore\_support(y\_test,y\_pred)

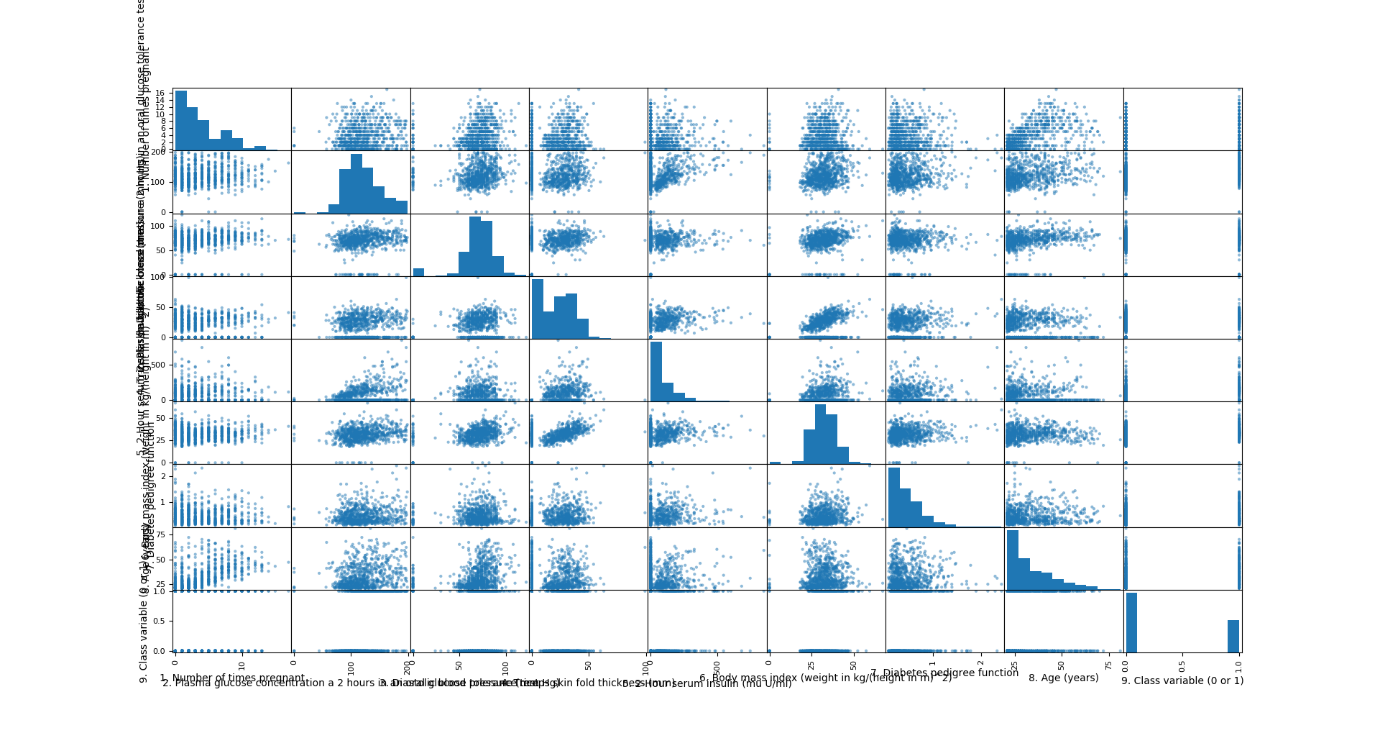
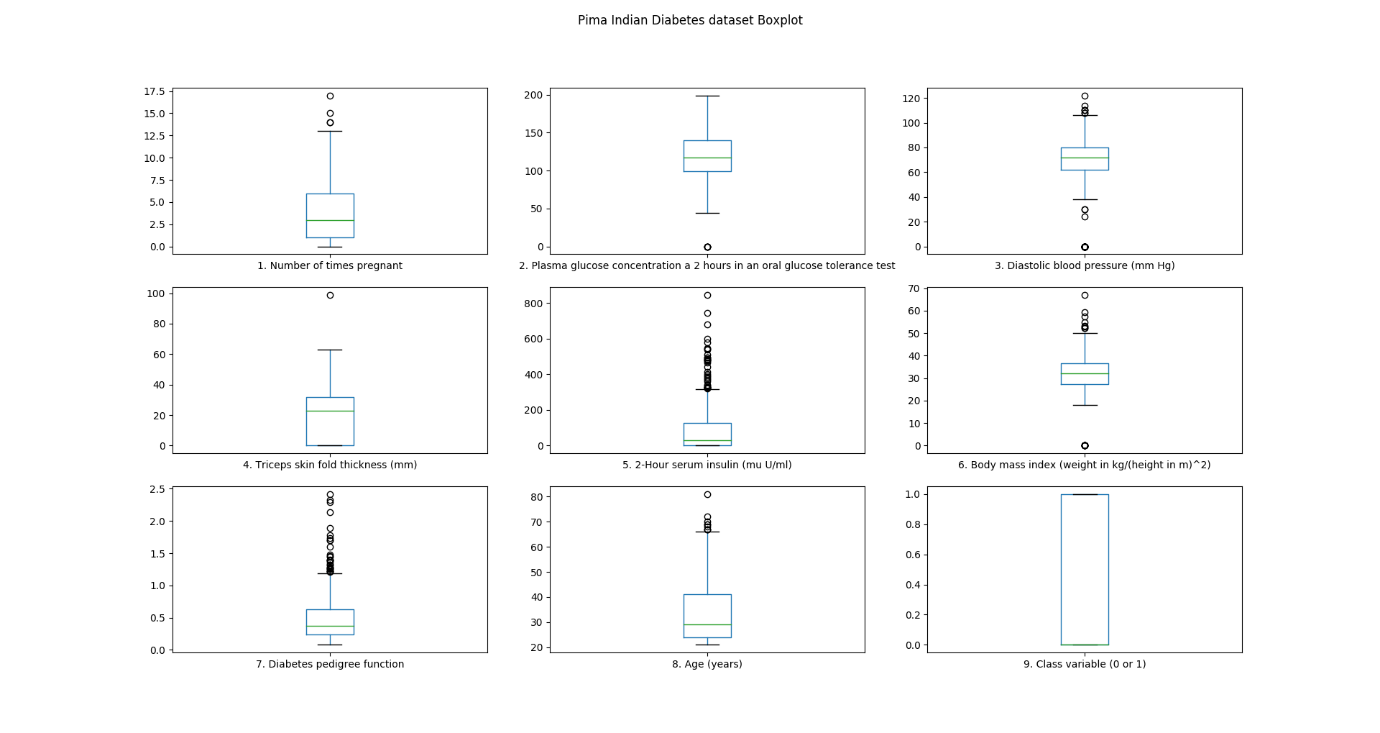
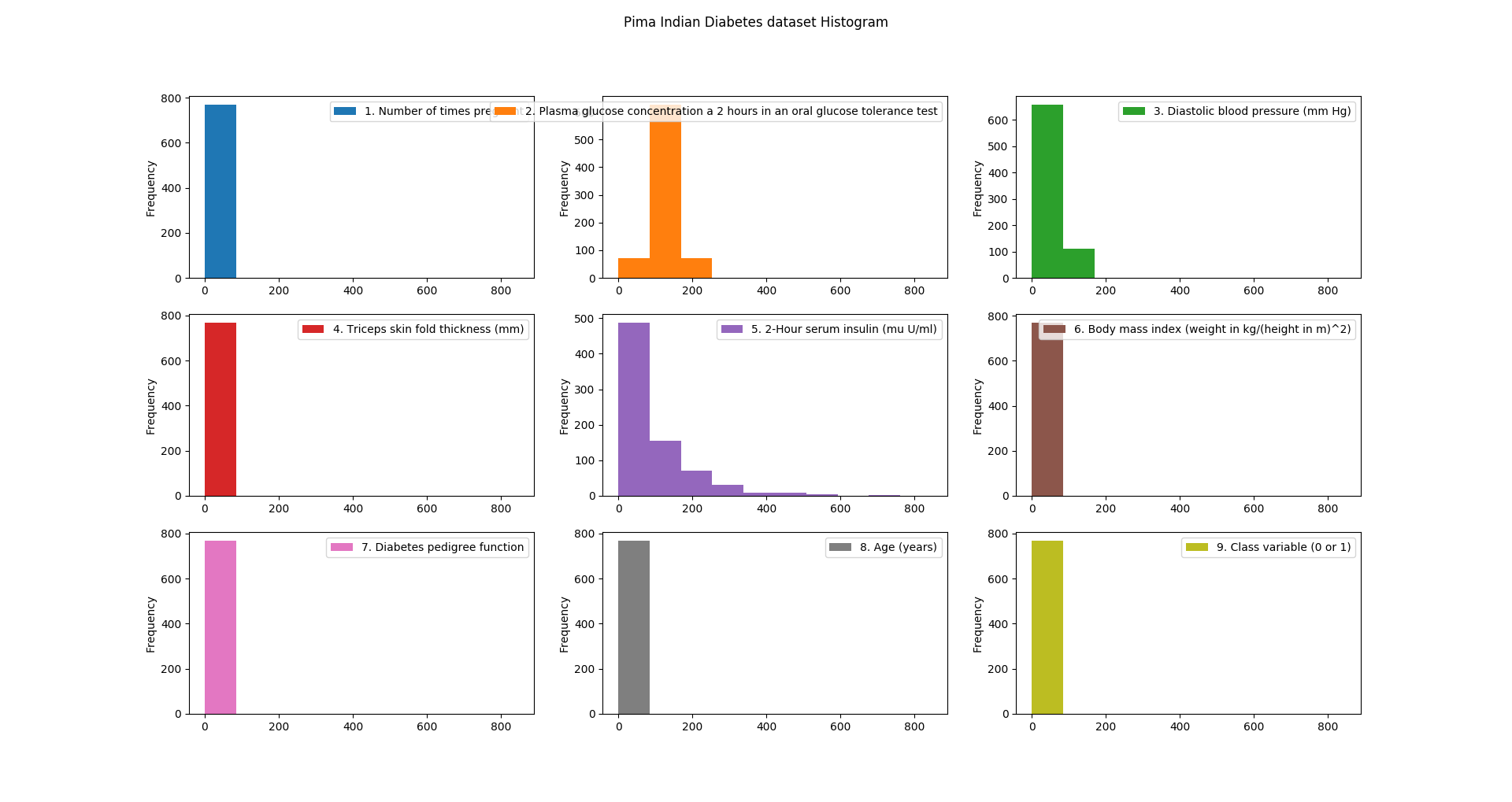
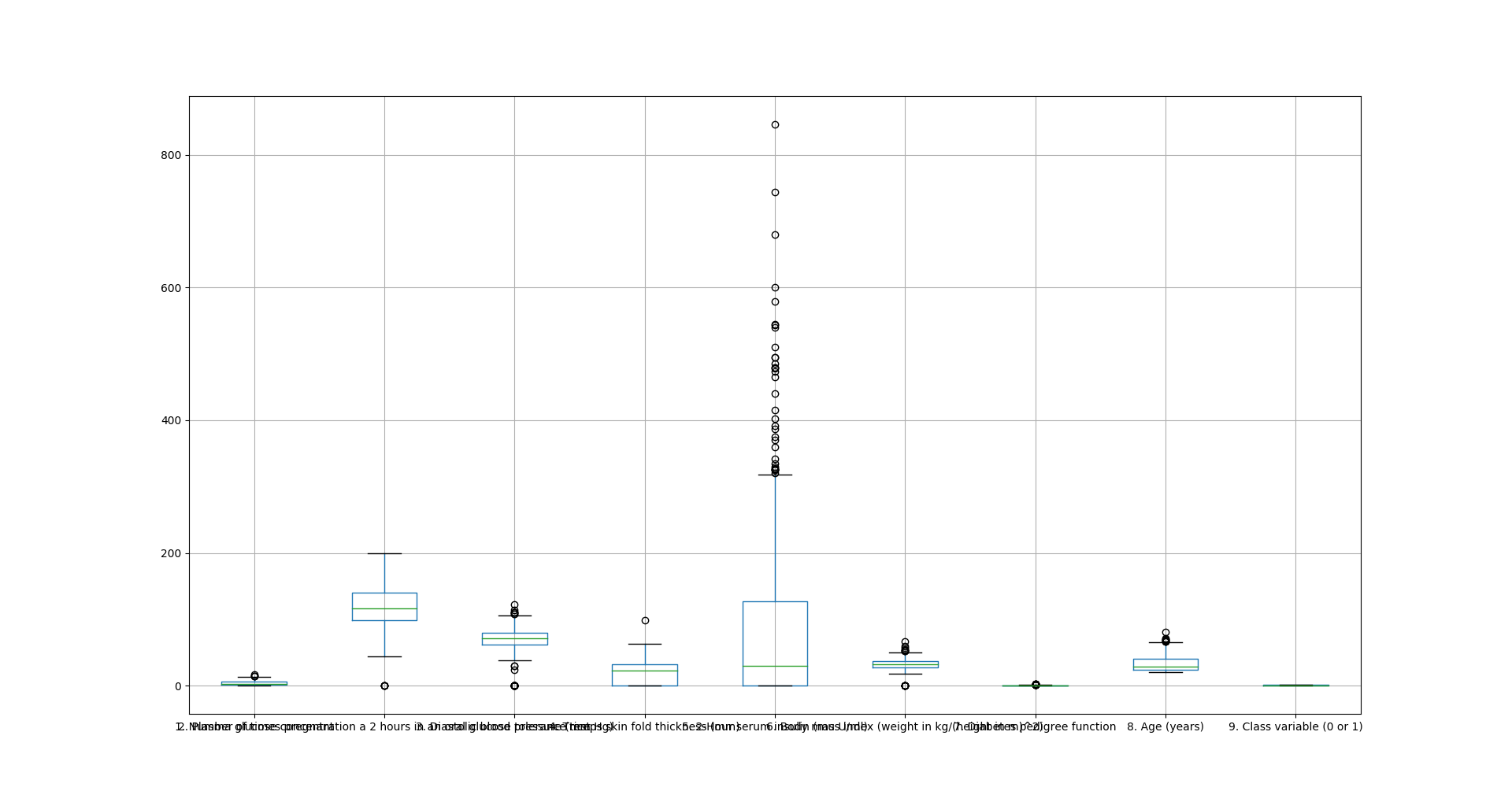
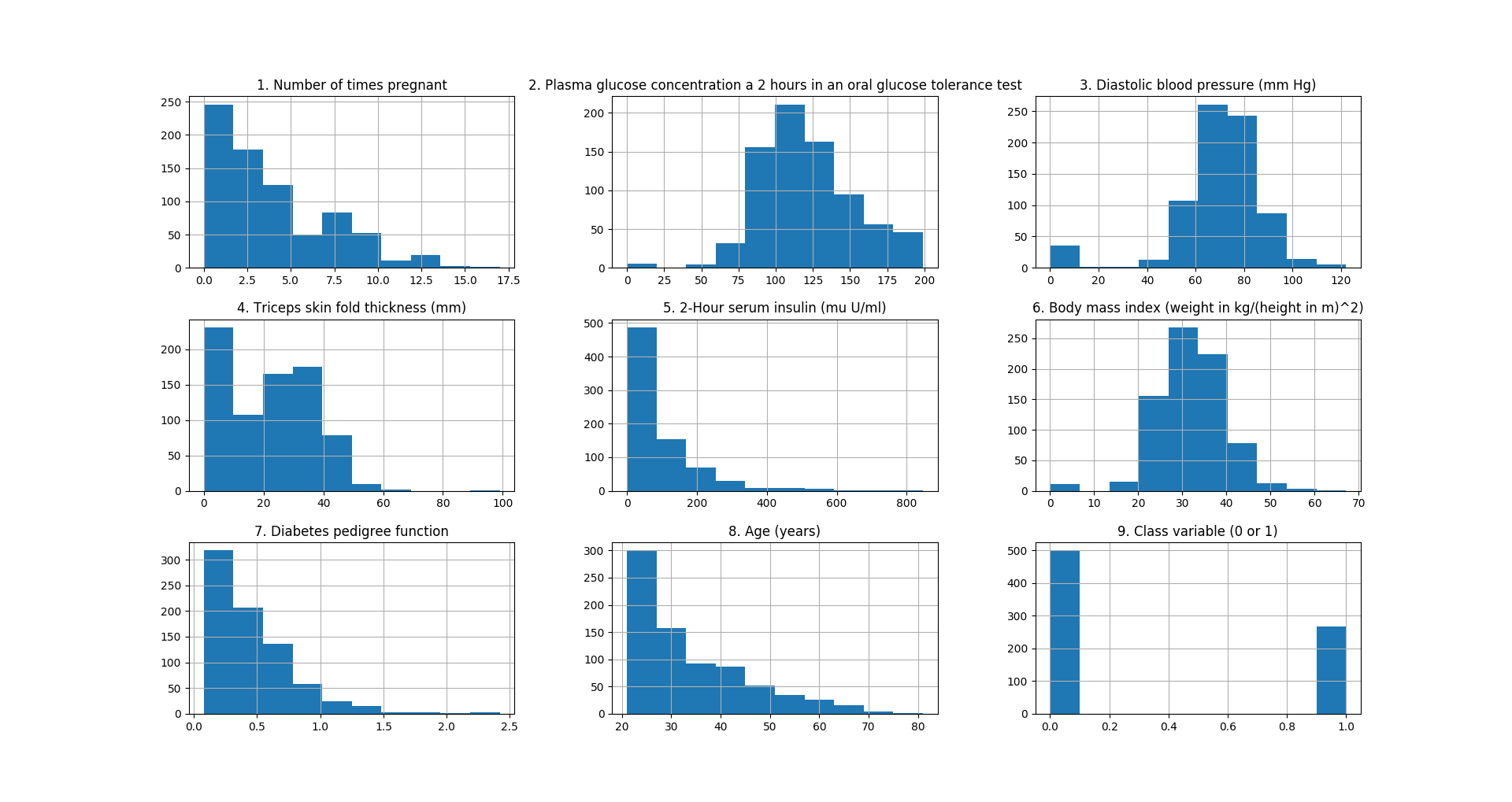
print("\nPrecision\n",pf[0])

print("\nRecall\n",pf[1])

print("\nFScore\n",pf[2])

print("\nSupport\n",pf[3])

**Output:**



----------------------------------------------Dataset Values-------------------------------------------------

1. Number of times pregnant 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test ... 8. Age (years) 9. Class variable (0 or 1)

0 6 148 ... 50 1

1 1 85 ... 31 0

2 8 183 ... 32 1

3 1 89 ... 21 0

4 0 137 ... 33 1

[5 rows x 9 columns]

----------------------------------------------Dataset Values-------------------------------------------------

1. Number of times pregnant 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test ... 8. Age (years) 9. Class variable (0 or 1)

763 10 101 ... 63 0

764 2 122 ... 27 0

765 5 121 ... 30 0

766 1 126 ... 47 1

767 1 93 ... 23 0

[5 rows x 9 columns]

--------------------------------------------------X values-----------------------------------------------------

1. Number of times pregnant 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test ... 7. Diabetes pedigree function 8. Age (years)

0 6 148 ... 0.627 50

1 1 85 ... 0.351 31

2 8 183 ... 0.672 32

3 1 89 ... 0.167 21

4 0 137 ... 2.288 33

[5 rows x 8 columns]

--------------------------------------------------Y values-----------------------------------------------------

0 1

1 0

2 1

3 0

4 1

Name: 9. Class variable (0 or 1), dtype: int64

--------------------------------------------------------------------------------------------------------

1. Number of times pregnant False

2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test False

3. Diastolic blood pressure (mm Hg) False

4. Triceps skin fold thickness (mm) False

5. 2-Hour serum insulin (mu U/ml) False

6. Body mass index (weight in kg/(height in m)^2) False

7. Diabetes pedigree function False

8. Age (years) False

dtype: bool

--------------------------------------------------------------------------------------------------------

False

--------------------------------------------------------------------------------------------------------

Information about X dataframe:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 768 entries, 0 to 767

Data columns (total 8 columns):

1. Number of times pregnant 768 non-null int64

2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test 768 non-null int64

3. Diastolic blood pressure (mm Hg) 768 non-null int64

4. Triceps skin fold thickness (mm) 768 non-null int64

5. 2-Hour serum insulin (mu U/ml) 768 non-null int64

6. Body mass index (weight in kg/(height in m)^2) 768 non-null float64

7. Diabetes pedigree function 768 non-null float64

8. Age (years) 768 non-null int64

dtypes: float64(2), int64(6)

memory usage: 48.1 KB

None

---------------------------------------------x\_train--------------------------------------------------------

1. Number of times pregnant 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test ... 7. Diabetes pedigree function 8. Age (years)

613 6 105 ... 0.878 26

159 17 163 ... 0.817 47

711 5 126 ... 0.439 40

745 12 100 ... 0.488 46

447 0 95 ... 0.330 26

[5 rows x 8 columns]

---------------------------------------------x\_test---------------------------------------------------------

1. Number of times pregnant 2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test ... 7. Diabetes pedigree function 8. Age (years)

661 1 199 ... 1.394 22

122 2 107 ... 0.404 23

113 4 76 ... 0.391 25

14 5 166 ... 0.587 51

529 0 111 ... 0.660 31

[5 rows x 8 columns]

---------------------------------------------y\_train--------------------------------------------------------

613 0

159 1

711 0

745 0

447 0

Name: 9. Class variable (0 or 1), dtype: int64

---------------------------------------------y\_test---------------------------------------------------------

661 1

122 0

113 0

14 1

529 0

Name: 9. Class variable (0 or 1), dtype: int64

('confusion matrix:\n', array([[147, 23],

[ 41, 43]]))

Correct Predictions

190

Total Predictions

254

Precision

0

Recall

0

ass10.py:108: RuntimeWarning: divide by zero encountered in long\_scalars

f\_measure = (2\*precision\*recall)/(precision+recall)

F-Measure

0

('\nAccuracy:\n', 0)

('\nError\n', 100)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*Calculated values using package\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

('\nPrecision\n', array([ 0.78191489, 0.65151515]))

('\nRecall\n', array([ 0.86470588, 0.51190476]))

('\nFScore\n', array([ 0.82122905, 0.57333333]))

('\nSupport\n', array([170, 84]))