dipak-mani-diabetes-prediction

March 6, 2023

[1]:
```
import pandas as pd
import numpy as np
!pip install scikit-learn
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import missingno as msno
```

```
Requirement already satisfied: scikit-learn in
c:\users\dipmani\anaconda3\lib\site-packages (1.0.2)
Requirement already satisfied: joblib>=0.11 in
c:\users\dipmani\anaconda3\lib\site-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: numpy>=1.14.6 in
c:\users\dipmani\anaconda3\lib\site-packages (from scikit-learn) (1.21.5)
Requirement already satisfied: scipy>=1.1.0 in
c:\users\dipmani\anaconda3\lib\site-packages (from scikit-learn) (1.9.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\users\dipmani\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
```

[2]:
```
# Loading the data
df = pd.read_csv("diabetes.csv")
```

[3]:
```
# reading top 5 data values
df.head()
```

[3]:

	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

0.1 Exploratory data analysis

[4]:
```
# Data Inormation - Coumns & its types
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
```

[5]:
```
## Data Description
df.describe()
```

[5]:

	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

[6]:
```
df.describe().transpose()
```

[6]:

	count	mean	std	min	25%	\
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	
Glucose	768.0	120.894531	31.972618	0.000	99.00000	
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	
Insulin	768.0	79.799479	115.244002	0.000	0.00000	
BMI	768.0	31.992578	7.884160	0.000	27.30000	
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	
Age	768.0	33.240885	11.760232	21.000	24.00000	
Outcome	768.0	0.348958	0.476951	0.000	0.00000	

	50%	75%	max
Pregnancies	3.0000	6.00000	17.00
Glucose	117.0000	140.25000	199.00
BloodPressure	72.0000	80.00000	122.00
SkinThickness	23.0000	32.00000	99.00
Insulin	30.5000	127.25000	846.00
BMI	32.0000	36.60000	67.10
DiabetesPedigreeFunction	0.3725	0.62625	2.42
Age	29.0000	41.00000	81.00
Outcome	0.0000	1.00000	1.00

[7]:
```
# Checking Standard deviation of each column variable
df.apply(np.std)
```

[7]:
```
Pregnancies                 3.367384
Glucose                    31.951796
BloodPressure              19.343202
SkinThickness              15.941829
Insulin                   115.168949
BMI                         7.879026
DiabetesPedigreeFunction    0.331113
Age                        11.752573
Outcome                     0.476641
dtype: float64
```

[8]:
```
# Checking for Missing values
df.isnull().any()
```

[8]:
```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
```

```
Outcome                    False
dtype: bool
```

[9]:
```
df.isnull().head(10)
```

[9]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin    BMI  \
0        False    False          False          False    False  False
1        False    False          False          False    False  False
2        False    False          False          False    False  False
3        False    False          False          False    False  False
4        False    False          False          False    False  False
5        False    False          False          False    False  False
6        False    False          False          False    False  False
7        False    False          False          False    False  False
8        False    False          False          False    False  False
9        False    False          False          False    False  False

   DiabetesPedigreeFunction    Age  Outcome
0                     False  False    False
1                     False  False    False
2                     False  False    False
3                     False  False    False
4                     False  False    False
5                     False  False    False
6                     False  False    False
7                     False  False    False
8                     False  False    False
9                     False  False    False
```

[10]:
```
# check null values of each column
df.isnull().sum()
```

[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[11]:
```
# check total null values present in dataset
df.isnull().sum().sum()
```

[11]:
```
0
```

[12]:
```
# Check the data types
df.dtypes
```

[12]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

[13]:
```
df_positive = df[df['Outcome'] == 1]
df_negative = df[df['Outcome'] == 0]

print("The count of positive data", df_positive.shape)
print("The count of negative data", df_negative.shape)
```

```
The count of positive data (268, 9)
The count of negative data (500, 9)
```

[14]:
```
df['Outcome'].value_counts()
```

[14]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

[15]:
```
df_positive.head()
```

[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
2	8	183	64	0	0	23.3	
4	0	137	40	35	168	43.1	
6	3	78	50	32	88	31.0	
8	2	197	70	45	543	30.5	

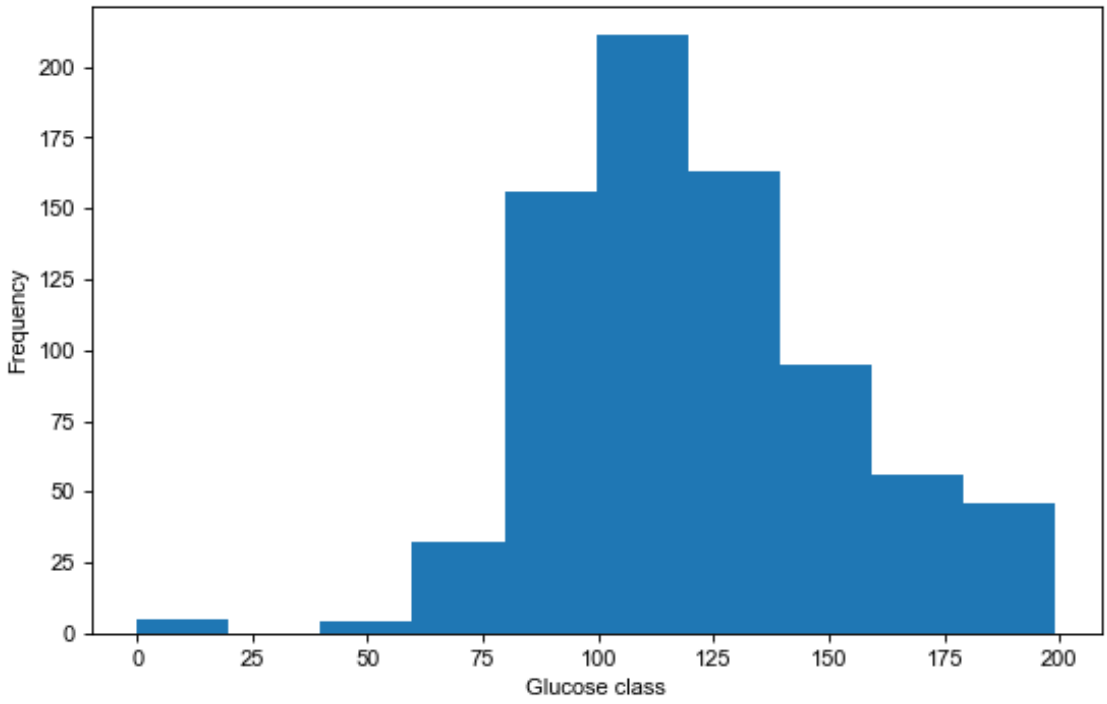
	DiabetesPedigreeFunction	Age	Outcome
0	0.627	50	1
2	0.672	32	1
4	2.288	33	1
6	0.248	26	1
8	0.158	53	1

[16]:
```
df['Glucose'].value_counts().head(10)
```

[16]:
```
99     17
100    17
111    14
129    14
125    14
106    14
112    13
108    13
95     13
105    13
Name: Glucose, dtype: int64
```

[17]:
```
plt.figure(figsize=(8,5),dpi=80)
plt.xlabel('Glucose class')
df['Glucose'].plot.hist()
sns.set_style(style="whitegrid")
print("Mean of Glucose level is :",df['Glucose'].mean())
print("Datatypes of Glucose Variable is",df['Glucose'].dtypes)
```

```
Mean of Glucose level is : 120.89453125
Datatypes of Glucose Variable is int64
```



[18]:
```
# we have to find out here how much 0 & 1 values present in glucose column
Glucose_zero = df[df['Glucose'] == 0]
```

```
Glucose_not_zero = df[df['Glucose'] != 0]
```

[19]:
```
# shape of Glucose column
Glucose_not_zero.shape
```

[19]: (763, 9)

[20]:
```
Glucose_zero.shape
```

[20]: (5, 9)

[21]:
```
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =␣
↪df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.
↪nan)
df
```

[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148.0	72.0	35.0	NaN	33.6	
1	1	85.0	66.0	29.0	NaN	26.6	
2	8	183.0	64.0	NaN	NaN	23.3	
3	1	89.0	66.0	23.0	94.0	28.1	
4	0	137.0	40.0	35.0	168.0	43.1	
..	…	…	…	…	…	…	
763	10	101.0	76.0	48.0	180.0	32.9	
764	2	122.0	70.0	27.0	NaN	36.8	
765	5	121.0	72.0	23.0	112.0	26.2	
766	1	126.0	60.0	NaN	NaN	30.1	
767	1	93.0	70.0	31.0	NaN	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]

[22]:
```
df['Glucose'].mean()
```

[22]: 121.6867627785059

[23]:
```
# Now the Missing values has been treated and filled with Mean values()
df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] =␣
 ↪df[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].mean()
df
```

[23]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	121.686763	72.405184	29.15342	155.548223	32.457464	0.627	50	1
1	1	121.686763	72.405184	29.15342	155.548223	32.457464	0.351	31	0
2	8	121.686763	72.405184	29.15342	155.548223	32.457464	0.672	32	1
3	1	121.686763	72.405184	29.15342	155.548223	32.457464	0.167	21	0
4	0	121.686763	72.405184	29.15342	155.548223	32.457464	2.288	33	1
..	…	…	…	…	…	…	…	…	…
763	10	121.686763	72.405184	29.15342	155.548223	32.457464	0.171	63	0
764	2	121.686763	72.405184	29.15342	155.548223	32.457464	0.340	27	0
765	5	121.686763	72.405184	29.15342	155.548223	32.457464	0.245	30	0
766	1	121.686763	72.405184	29.15342	155.548223	32.457464	0.349	47	1
767	1	121.686763	72.405184	29.15342	155.548223	32.457464	0.315	23	0

[768 rows x 9 columns]

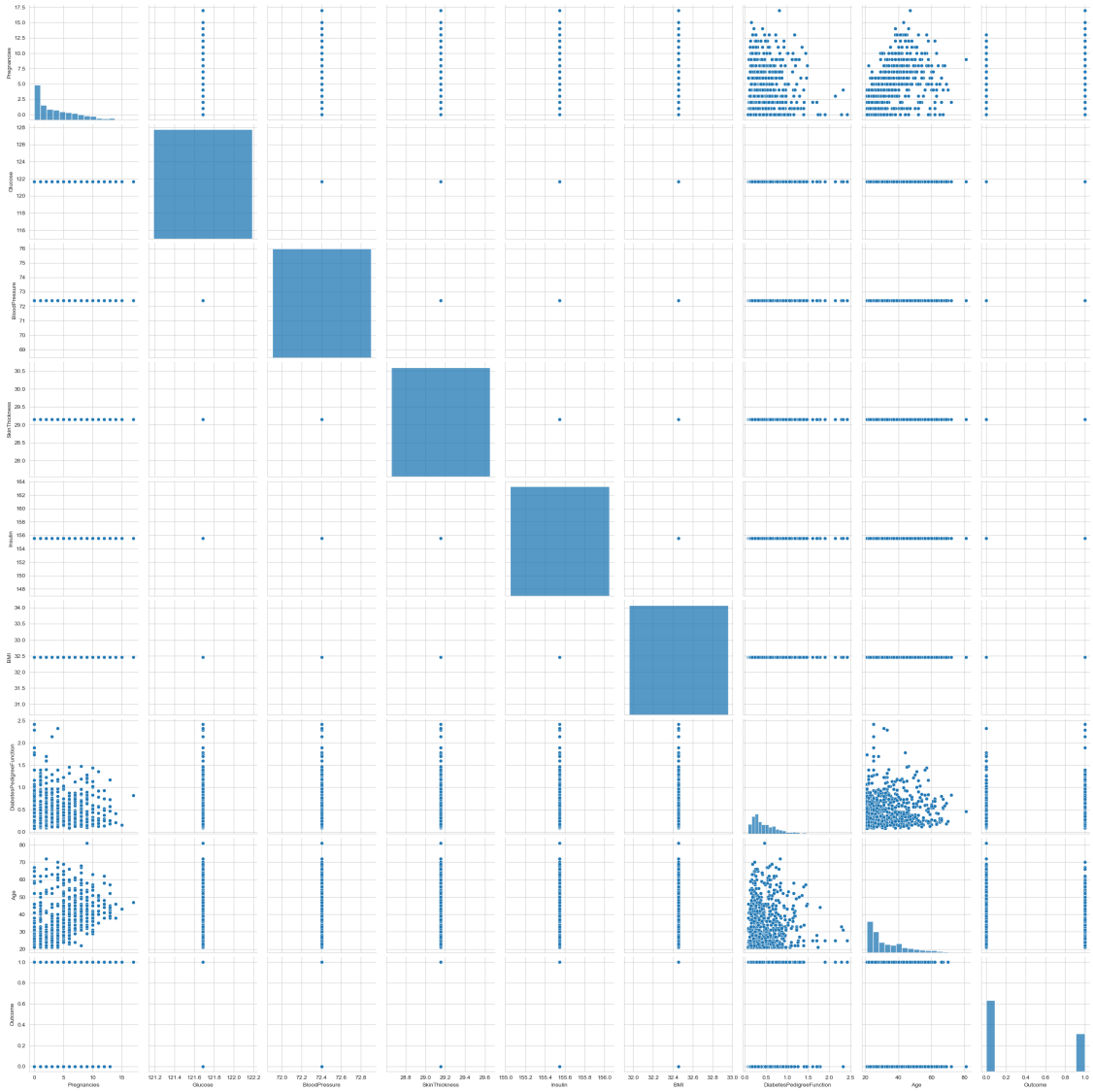
[24]:
```
df.isna().sum()
```

[24]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

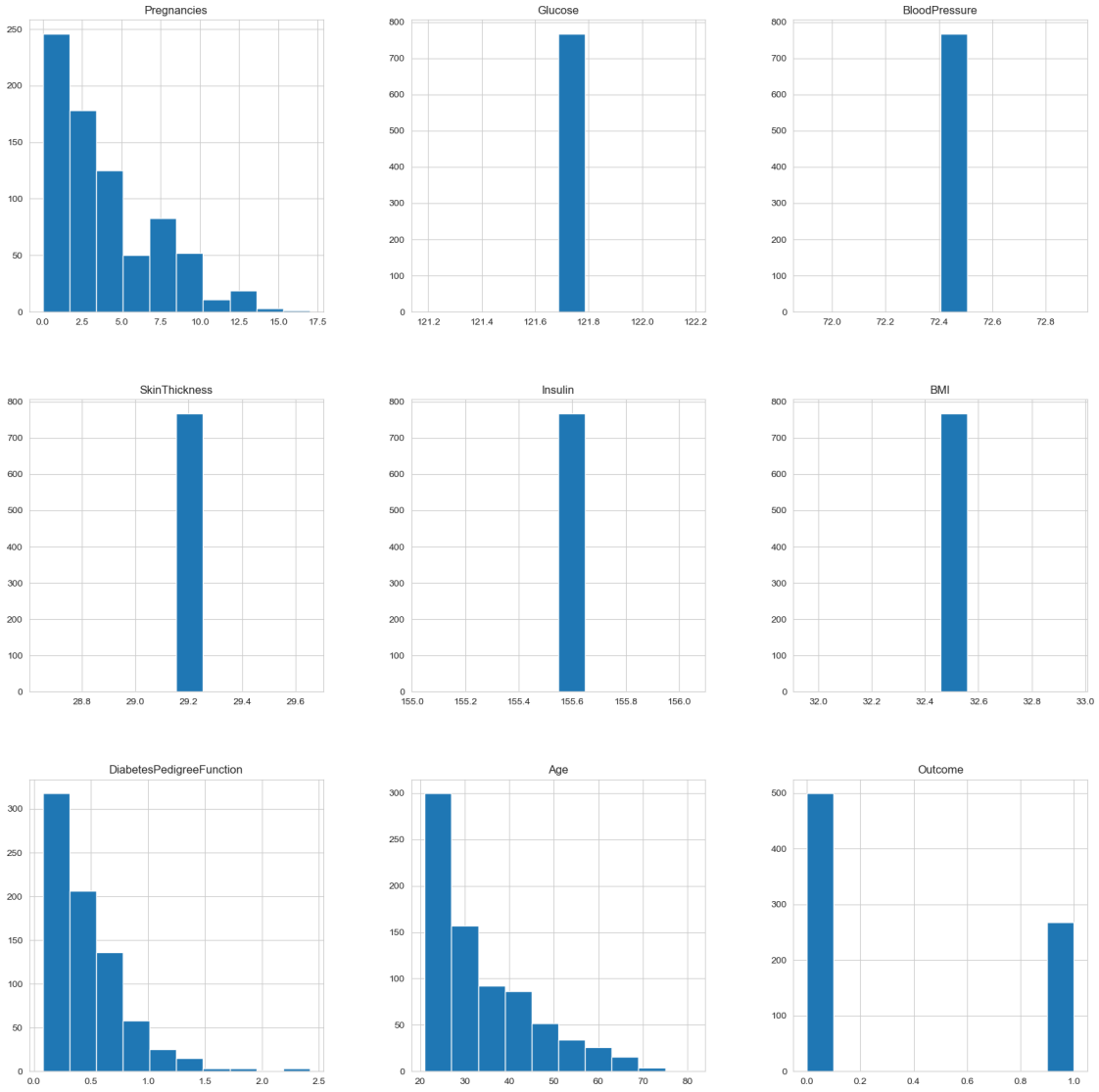
[25]:
```
# Plot data and let see how it looks

sns.pairplot(df,size=3)
```

```
C:\Users\DIPMANI\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2095:
UserWarning: The `size` parameter has been renamed to `height`; please update
your code.
  warnings.warn(msg, UserWarning)
```

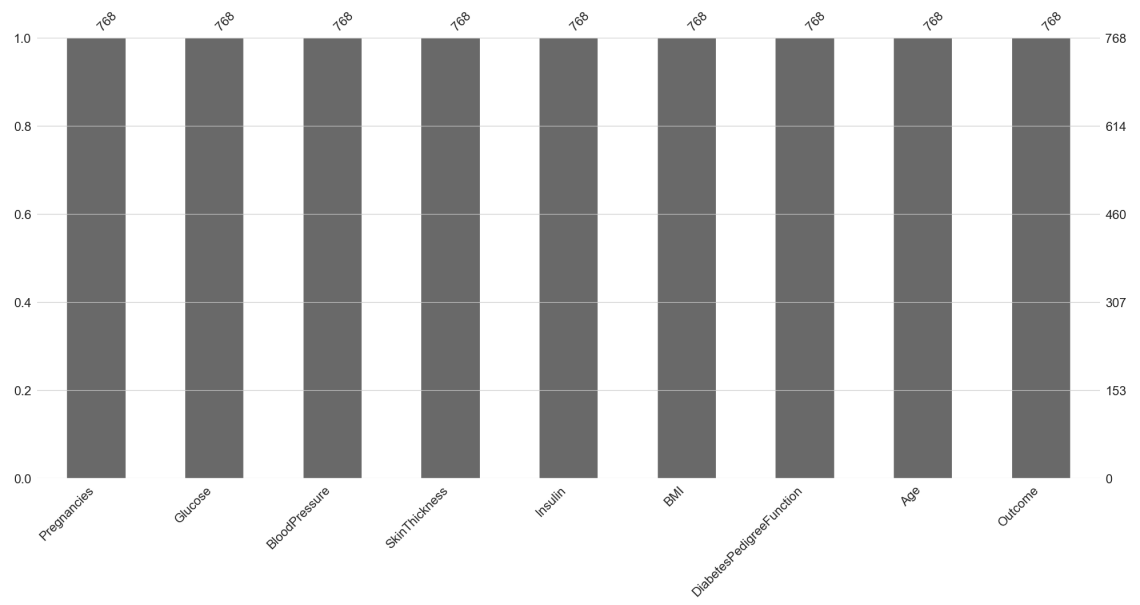
[25]:
```
<seaborn.axisgrid.PairGrid at 0x1d1cb6e3970>
```



[26]:
```
p = df.hist(figsize = (20,20))
```

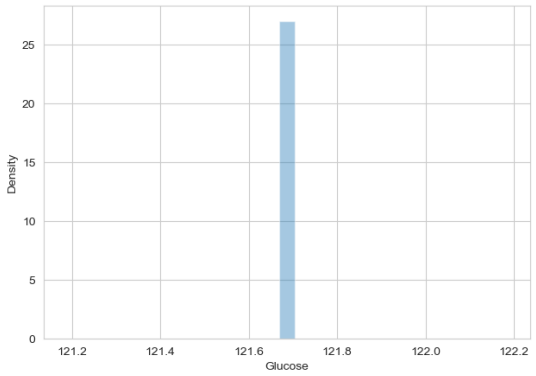


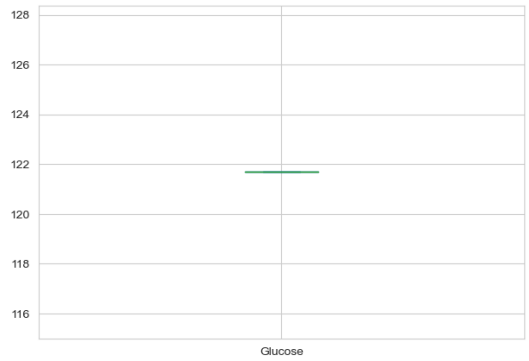
[27]:
```
# MISSING VALUES CHECKS WITH BAR GRAPHS
p = msno.bar(df)
```



[28]:
```
plt.subplot(121), sns.distplot(df['Glucose'])
plt.subplot(122), df['Glucose'].plot.box(figsize=(16,5))
plt.show()
```

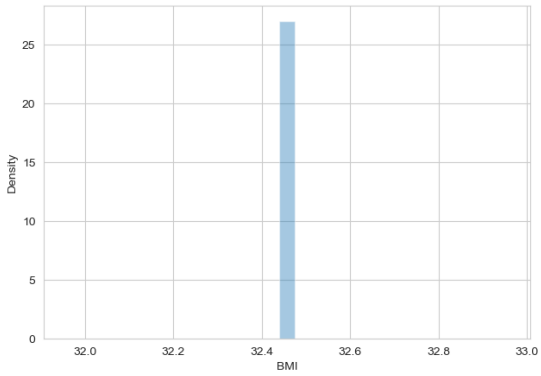
```
C:\Users\DIPMANI\AppData\Local\Temp\ipykernel_22676\1037445198.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  plt.subplot(121), sns.distplot(df['Glucose'])
C:\Users\DIPMANI\Anaconda3\lib\site-packages\seaborn\distributions.py:2517:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```

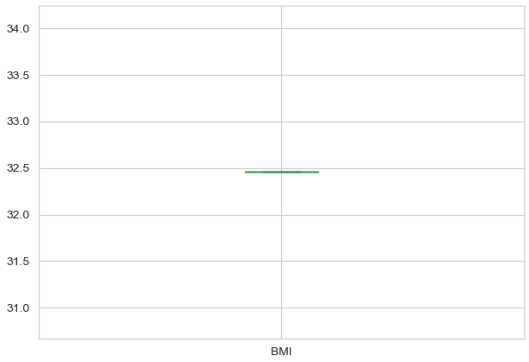




[29]:
```
plt.subplot(121), sns.distplot(df['BMI'])
plt.subplot(122), df['BMI'].plot.box(figsize=(16,5))
plt.show()
```

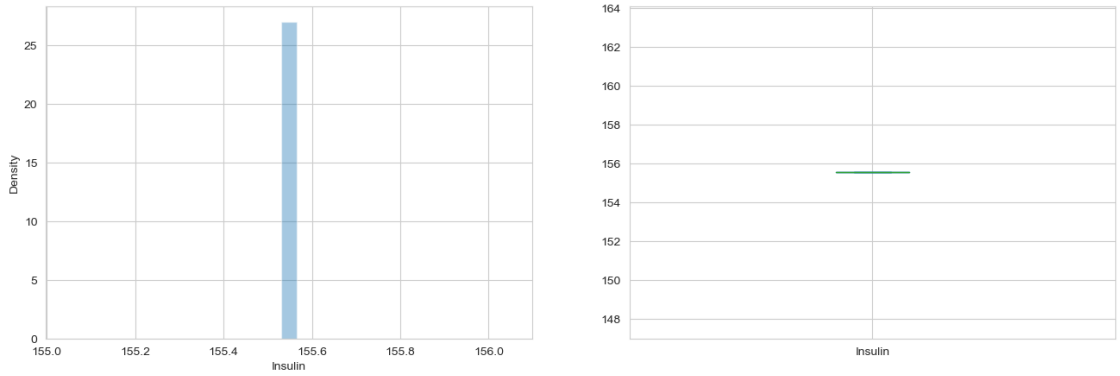
```
C:\Users\DIPMANI\AppData\Local\Temp\ipykernel_22676\195754228.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  plt.subplot(121), sns.distplot(df['BMI'])
C:\Users\DIPMANI\Anaconda3\lib\site-packages\seaborn\distributions.py:2517:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```





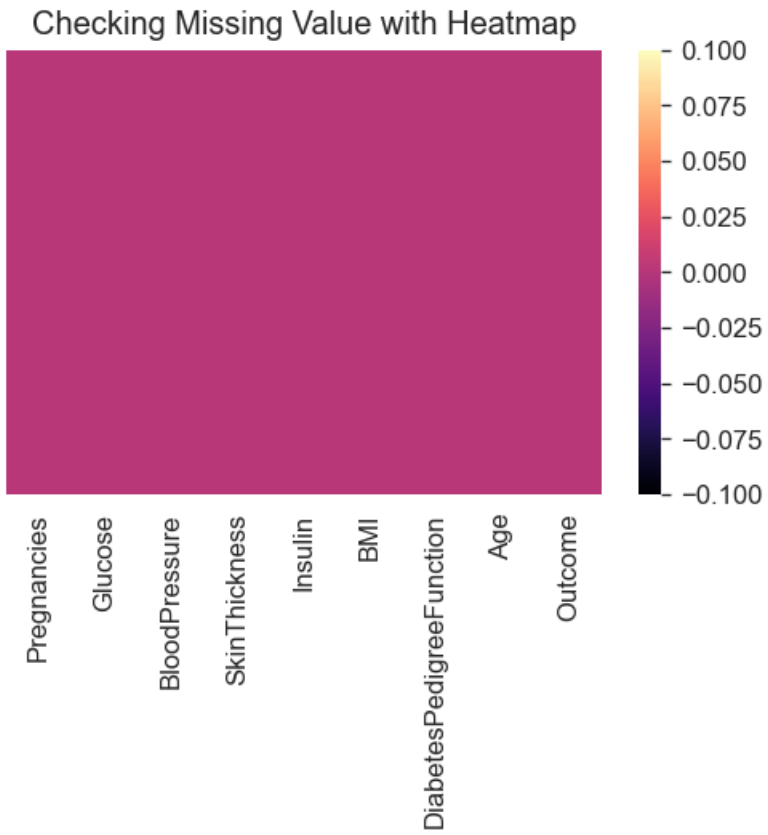
[30]:
```
plt.subplot(121), sns.distplot(df['Insulin'])
plt.subplot(122), df['Insulin'].plot.box(figsize=(16,5))
plt.show()
```

```
C:\Users\DIPMANI\AppData\Local\Temp\ipykernel_22676\3209351696.py:1:
UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  plt.subplot(121), sns.distplot(df['Insulin'])
C:\Users\DIPMANI\Anaconda3\lib\site-packages\seaborn\distributions.py:2517:
UserWarning: Dataset has 0 variance; skipping density estimate. Pass
`warn_singular=False` to disable this warning.
  kdeplot(**{axis: a}, ax=ax, color=kde_color, **kde_kws)
```

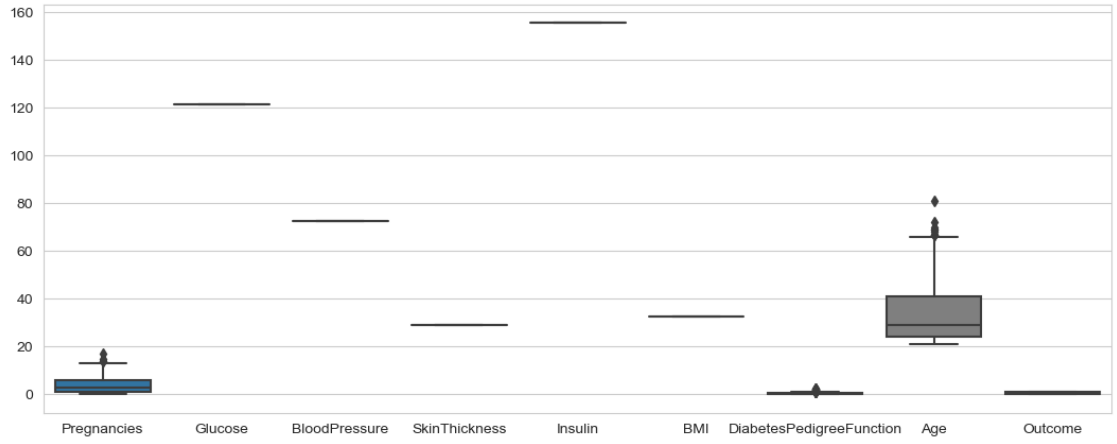


[31]:
```
## Using Heat map To check missing values
plt.figure(figsize=(5,3),dpi=120)
plt.title('Checking Missing Value with Heatmap')
sns.heatmap(df.isnull(),cmap='magma',yticklabels=False)
```

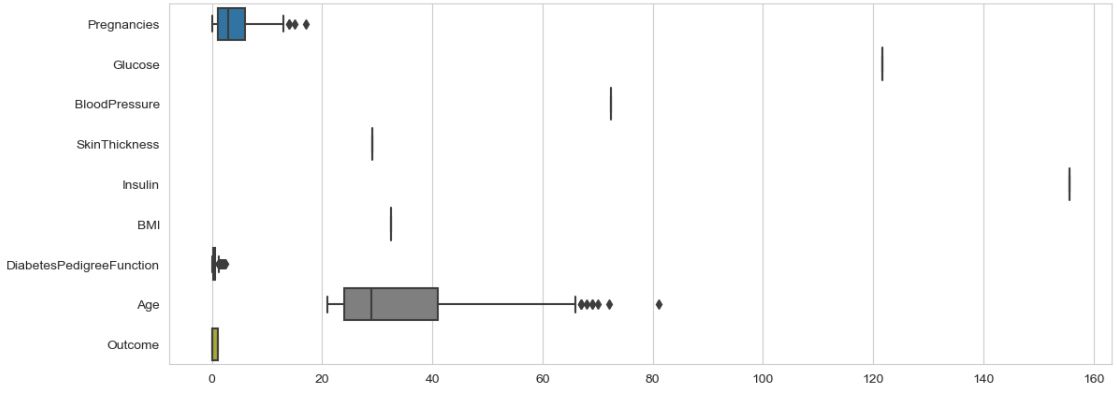
[31]: <AxesSubplot:title={'center':'Checking Missing Value with Heatmap'}>



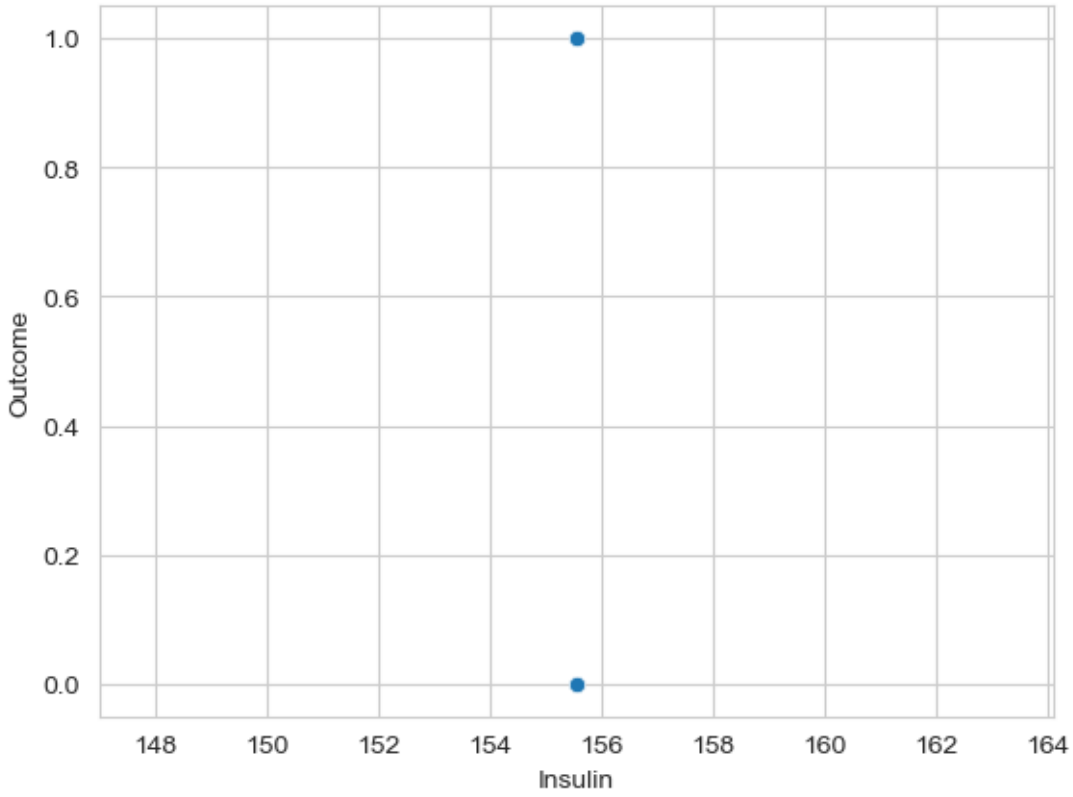
[32]:
```
# For using boxplot showing outliers is present or not

plt.figure(figsize=(13,5))
sns.boxplot(data=df)
plt.show()
```



[33]:
```
plt.figure(figsize=(13,5))
sns.boxplot(data=df,orient='h')
plt.show()
```



[34]:
```
sns.scatterplot(x=df['Insulin'],y=df['Outcome'])
plt.show()
```



[35]:
```
# Corelation metrics
df.corr()
```

[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	\
Pregnancies	1.000000	NaN	NaN	NaN	
Glucose	NaN	NaN	NaN	NaN	
BloodPressure	NaN	NaN	NaN	NaN	
SkinThickness	NaN	NaN	NaN	NaN	
Insulin	NaN	NaN	NaN	NaN	
BMI	NaN	NaN	NaN	NaN	
DiabetesPedigreeFunction	-0.033523	NaN	NaN	NaN	
Age	0.544341	NaN	NaN	NaN	
Outcome	0.221898	NaN	NaN	NaN	

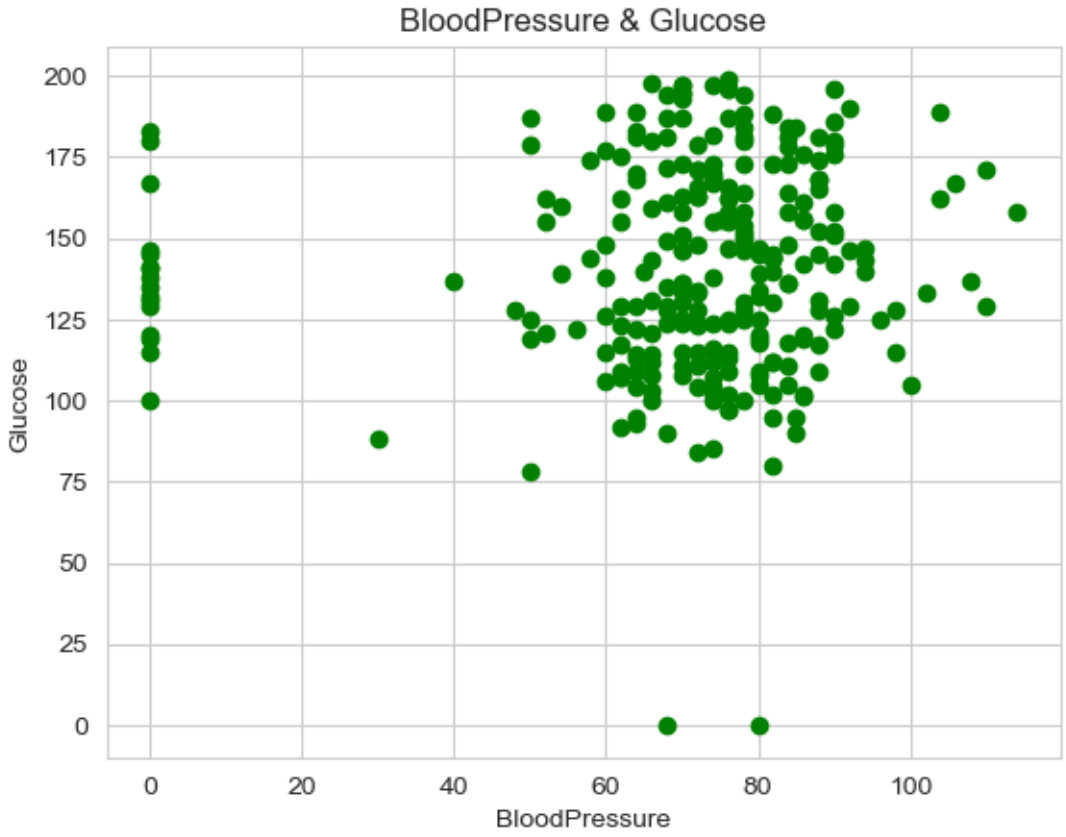
	Insulin	BMI	DiabetesPedigreeFunction	Age	\
Pregnancies	NaN	NaN	-0.033523	0.544341	
Glucose	NaN	NaN	NaN	NaN	
BloodPressure	NaN	NaN	NaN	NaN	
SkinThickness	NaN	NaN	NaN	NaN	
Insulin	NaN	NaN	NaN	NaN	
BMI	NaN	NaN	NaN	NaN	
DiabetesPedigreeFunction	NaN	NaN	1.000000	0.033561	
Age	NaN	NaN	0.033561	1.000000	
Outcome	NaN	NaN	0.173844	0.238356	

	Outcome
Pregnancies	0.221898
Glucose	NaN
BloodPressure	NaN
SkinThickness	NaN
Insulin	NaN
BMI	NaN
DiabetesPedigreeFunction	0.173844
Age	0.238356
Outcome	1.000000

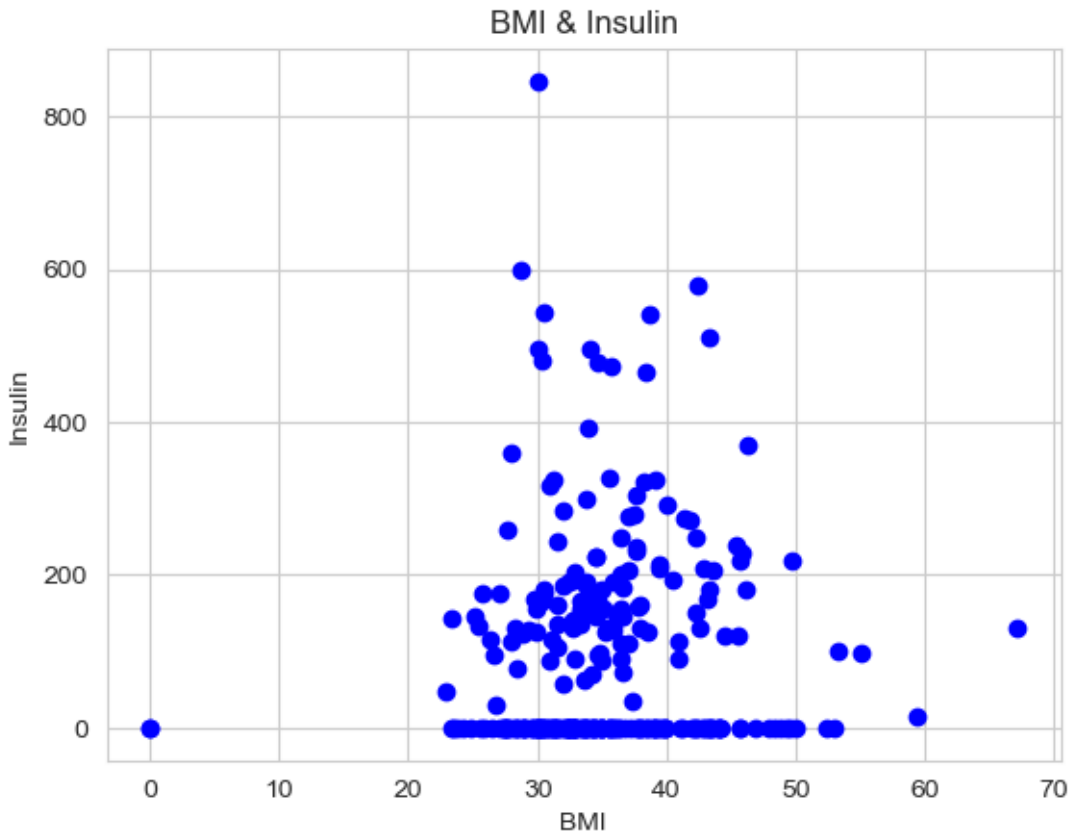
[36]:
```
# Scatter plot

BloodPressure = df_positive['BloodPressure']
Glucose = df_positive['Glucose']
SkinThickness = df_positive['SkinThickness']
Insulin = df_positive['Insulin']
BMI = df_positive['BMI']
```

[37]:
```
plt.scatter(BloodPressure, Glucose, color=['g'])
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
```

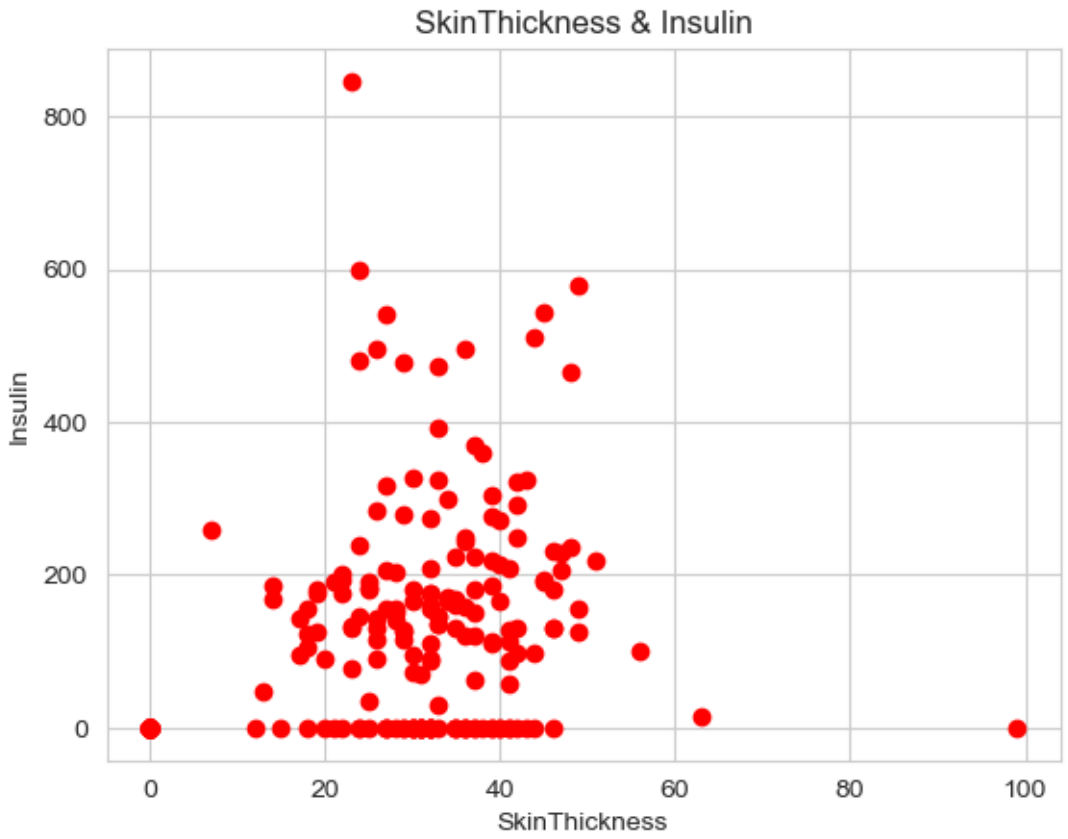
```
plt.title('BloodPressure & Glucose')
plt.show()
```



[38]:
```
plt.scatter(BMI, Insulin, color=['b'])
plt.xlabel('BMI')
plt.ylabel('Insulin')
plt.title('BMI & Insulin')
plt.show()
```



[39]:
```
plt.scatter(SkinThickness, Insulin, color=['r'])
plt.xlabel('SkinThickness')
plt.ylabel('Insulin')
plt.title('SkinThickness & Insulin')
plt.show()
```



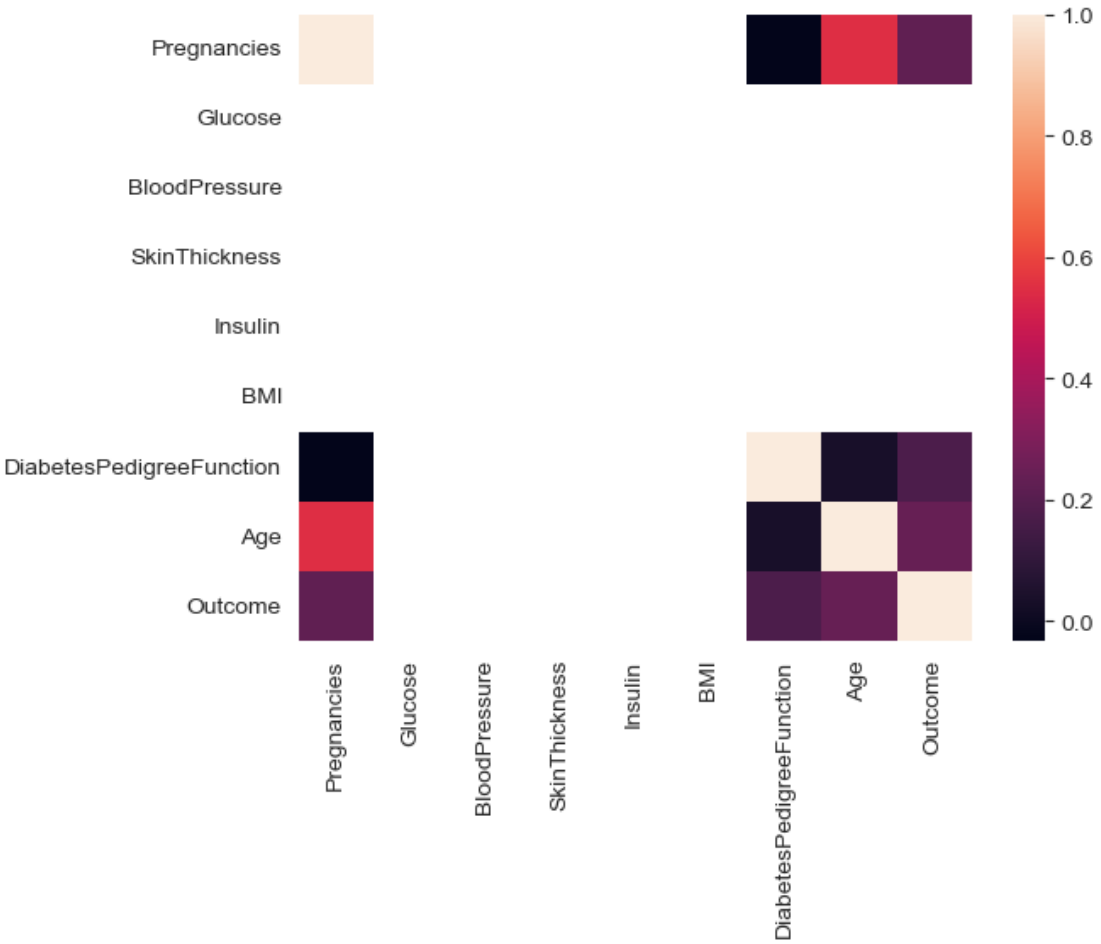
[40]:
```
correlation = df.corr()
print(correlation)
```

```
                          Pregnancies  Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000      NaN            NaN            NaN
Glucose                           NaN      NaN            NaN            NaN
BloodPressure                     NaN      NaN            NaN            NaN
SkinThickness                     NaN      NaN            NaN            NaN
Insulin                           NaN      NaN            NaN            NaN
BMI                               NaN      NaN            NaN            NaN
DiabetesPedigreeFunction    -0.033523      NaN            NaN            NaN
Age                          0.544341      NaN            NaN            NaN
Outcome                      0.221898      NaN            NaN            NaN

                          Insulin  BMI  DiabetesPedigreeFunction       Age  \
Pregnancies                   NaN  NaN                 -0.033523  0.544341
Glucose                       NaN  NaN                       NaN       NaN
BloodPressure                 NaN  NaN                       NaN       NaN
SkinThickness                 NaN  NaN                       NaN       NaN
Insulin                       NaN  NaN                       NaN       NaN
```

```
BMI                         NaN  NaN                  NaN       NaN
DiabetesPedigreeFunction    NaN  NaN             1.000000  0.033561
Age                         NaN  NaN             0.033561  1.000000
Outcome                     NaN  NaN             0.173844  0.238356

                          Outcome
Pregnancies              0.221898
Glucose                       NaN
BloodPressure                 NaN
SkinThickness                 NaN
Insulin                       NaN
BMI                           NaN
DiabetesPedigreeFunction 0.173844
Age                      0.238356
Outcome                  1.000000
```

[41]:
```
sns.heatmap(correlation)
```

[41]: <AxesSubplot:>



[42]:
```
from sklearn.model_selection import train_test_split
```

0.2 Train test split

[43]:
```
X=df.drop('Outcome',axis=1)
Y=df['Outcome']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)
X_train
```

[43]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
168	4	121.686763	72.405184	29.15342	155.548223	32.457464	0.471	29
591	2	121.686763	72.405184	29.15342	155.548223	32.457464	0.175	24
366	6	121.686763	72.405184	29.15342	155.548223	32.457464	0.368	29
18	1	121.686763	72.405184	29.15342	155.548223	32.457464	0.183	33
321	3	121.686763	72.405184	29.15342	155.548223	32.457464	0.197	25
..	…	…	…	…	…	…	…	…
8	2	121.686763	72.405184	29.15342	155.548223	32.457464	0.158	53
604	4	121.686763	72.405184	29.15342	155.548223	32.457464	0.212	36
724	1	121.686763	72.405184	29.15342	155.548223	32.457464	0.265	45
403	9	121.686763	72.405184	29.15342	155.548223	32.457464	0.280	38
423	2	121.686763	72.405184	29.15342	155.548223	32.457464	0.421	21

[614 rows x 8 columns]

0.3 Training the model

[52]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression()
model.fit(X_train, Y_train)
```

[52]:
```
LogisticRegression()
```

0.4 Making Predictions

[53]:
```
predictions = model.predict(X_test)
```

[54]:
```
print(predictions)
```

```
[1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0]
```

0.5 Evaluation

[57]:
```
accuracy = accuracy_score(predictions, Y_test)
```

[58]:
```
print(accuracy)
```

```
0.6688311688311688
```

[60]:
```
#X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size=0.
↪25,random_state=None)
```

[ ]:
```
#X_train
```

[ ]:
```
#Y_train
```

[61]:
```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, Y_train)
```

[61]:
```
RandomForestClassifier()
```

[62]:
```
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(Y_train, rfc_train)))
```

```
Accuracy_Score = 1.0
```

[63]:
```
from sklearn import metrics

predictions = rfc.predict(X_test)
print("Accuracy_Score =", format(metrics.accuracy_score(Y_test, predictions)))
```

```
Accuracy_Score = 0.6558441558441559
```

[64]:
```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test,predictions))
```

```
[[81 18]
 [35 20]]
              precision    recall  f1-score   support

           0       0.70      0.82      0.75        99
           1       0.53      0.36      0.43        55

    accuracy                           0.66       154
   macro avg       0.61      0.59      0.59       154
weighted avg       0.64      0.66      0.64       154
```

0.6 Standard Scaler

[65]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
```

[66]:
```
X_train_std=std.fit_transform(X_train)
X_test_std=std.transform(X_test)
```

[67]:
```
X_train_std
```

[67]:
```
array([[ 1.57441082e-02,  4.26325641e-14, -1.42108547e-14, …,
         0.00000000e+00,  2.64414485e-02, -3.56677496e-01],
       [-5.70127553e-01,  4.26325641e-14, -1.42108547e-14, …,
         0.00000000e+00, -8.91704256e-01, -7.88800586e-01],
       [ 6.01615770e-01,  4.26325641e-14, -1.42108547e-14, …,
         0.00000000e+00, -2.93048442e-01, -3.56677496e-01],
       …,
       [-8.63063384e-01,  4.26325641e-14, -1.42108547e-14, …,
         0.00000000e+00, -6.12538332e-01,  1.02611639e+00],
       [ 1.48042326e+00,  4.26325641e-14, -1.42108547e-14, …,
         0.00000000e+00, -5.66010679e-01,  4.21144068e-01],
       [-5.70127553e-01,  4.26325641e-14, -1.42108547e-14, …,
         0.00000000e+00, -1.28650731e-01, -1.04807444e+00]])
```

0.7 Decision Tree

[68]:
```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
```

[69]:
```
dtree.fit(X_train,Y_train)
```

[69]:
```
DecisionTreeClassifier()
```

[70]:
```
from sklearn import metrics

predictions = dtree.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(Y_test,predictions)))
```

```
Accuracy Score = 0.6103896103896104
```

[71]:
```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(Y_test, predictions))
print(classification_report(Y_test,predictions))
```

```
[[68 31]
 [29 26]]
              precision    recall  f1-score   support

           0       0.70      0.69      0.69        99
           1       0.46      0.47      0.46        55

    accuracy                           0.61       154
   macro avg       0.58      0.58      0.58       154
weighted avg       0.61      0.61      0.61       154
```

0.8 Support Vector Machine (SVM)

[72]:
```
from sklearn.svm import SVC

svc_model = SVC()
```

[73]:
```
svc_model.fit(X_train, Y_train)
```

[73]:
```
SVC()
```

[74]:
```
from sklearn import metrics
svc_pred = svc_model.predict(X_test)
print("Accuracy Score =", format(metrics.accuracy_score(Y_test, svc_pred)))
```

```
Accuracy Score = 0.6428571428571429
```

[75]:
```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(Y_test, svc_pred))
print(classification_report(Y_test,svc_pred))
```

```
[[99  0]
 [55  0]]
              precision    recall  f1-score   support

           0       0.64      1.00      0.78        99
           1       0.00      0.00      0.00        55

    accuracy                           0.64       154
   macro avg       0.32      0.50      0.39       154
weighted avg       0.41      0.64      0.50       154

C:\Users\DIPMANI\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\DIPMANI\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\DIPMANI\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1318: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[76]:
```
print("Train Set: ", X_train.shape, Y_train.shape)
print("Test Set: ", X_test.shape, Y_test.shape)
```

```
Train Set:  (614, 8) (614,)
Test Set:  (154, 8) (154,)
```

[ ]:

[ ]: