flzuz2cca

July 27, 2024

[1]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

[2]:
```
df=pd.read_csv("diabetes.csv")
```

[3]:
```
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.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
```

[ ]:
```
# --preprocessing
```

[5]:
```
df.isnull().sum()
```

[5]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

[6]:
```
df1=df.copy()
```

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

[7]:

	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	

```
   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
```

[8]:
```
# df1["Glucose"].mean()
```

[9]:
```
# df1["Glucose"]=df1["Glucose"].replace(0,df1["Glucose"].mean())
```

[10]:
```
df1
```

[10]:
```
     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]
```

[11]:
```
df1.tail()
```

[11]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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
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
```

[12]:
```
contain_zero = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI",␣
↪"DiabetesPedigreeFunction", "Age"]
```

[13]:
```
(df1[contain_zero] == 0).any()
```

[13]:
```
Glucose                    True
BloodPressure              True
SkinThickness              True
Insulin                    True
BMI                        True
DiabetesPedigreeFunction  False
Age                       False
dtype: bool
```

[14]:
```
replace_zero = ["Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI"]
```

[15]:
```
# df1[replace_zero]=df1[replace_zero].replace(0,df1[replace_zero].mean())
```

[16]:
```
df1[replace_zero]=df1[replace_zero].replace(0,df1[replace_zero].median())
```

[17]:
```
df1
```

[17]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35     30.5  33.6
1              1       85             66             29     30.5  26.6
2              8      183             64             23     30.5  23.3
3              1       89             66             23     94.0  28.1
4              0      137             40             35    168.0  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48    180.0  32.9
764            2      122             70             27     30.5  36.8
765            5      121             72             23    112.0  26.2
766            1      126             60             23     30.5  30.1
767            1       93             70             31     30.5  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]
```

[18]:
```
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
```

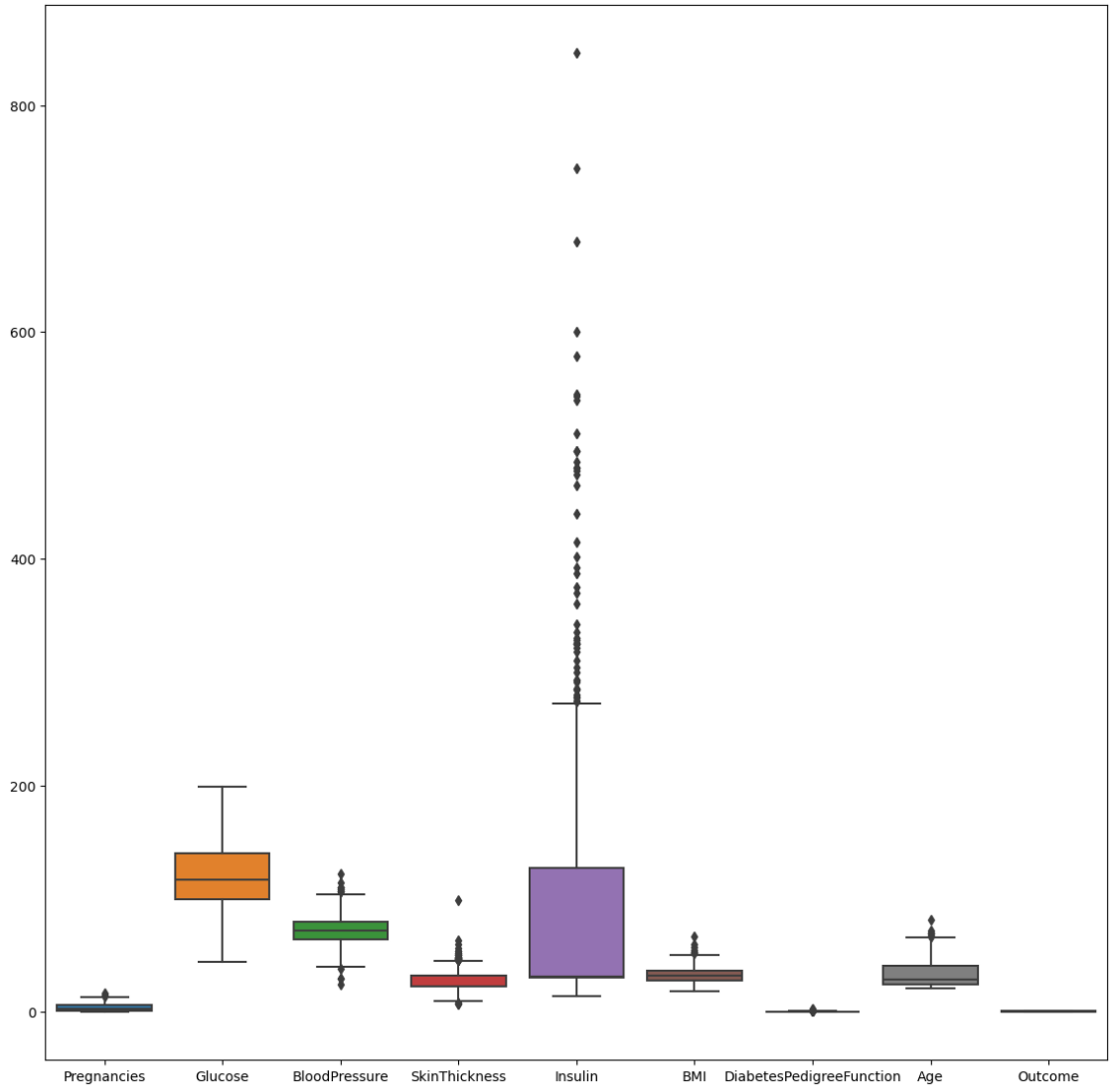
[19]:
```
(df1[contain_zero] == 0).any()
```

[19]:
```
Glucose                   False
BloodPressure             False
SkinThickness             False
Insulin                   False
BMI                       False
DiabetesPedigreeFunction  False
Age                       False
dtype: bool
```

[ ]:
```
# --outlier
```

[21]:
```
plt.figure(figsize=(14,14))
sns.boxplot(data=df1)
```

[21]: <Axes: >



[22]:
```
df100=df1[["Pregnancies","BloodPressure","SkinThickness","Glucose","Insulin",␣
  ↪"BMI","DiabetesPedigreeFunction","Age","Outcome"]]

Q1=df100.quantile(0.25)
Q3=df100.quantile(0.75)
IQR=Q3-Q1
Upper_fence=(Q3+1.5*IQR)
Lower_fence=(Q1-1.5*IQR)
```

```
print("Upper_fence \n",Upper_fence)
print("lower fence \n",Lower_fence)
```

```
Upper_fence
 Pregnancies                 13.500
BloodPressure               104.000
SkinThickness                45.500
Glucose                     201.000
Insulin                     272.375
BMI                          50.250
DiabetesPedigreeFunction      1.200
Age                          66.500
Outcome                       2.500
dtype: float64
lower fence
 Pregnancies                 -6.500
BloodPressure                40.000
SkinThickness                 9.500
Glucose                      39.000
Insulin                    -114.625
BMI                          13.850
DiabetesPedigreeFunction     -0.330
Age                          -1.500
Outcome                      -1.500
dtype: float64
```

[ ]:
```
# --caping
```

[24]:
```
df100["Pregnancies"]=np.where(df100["Pregnancies"]>13,13,np.
↪where(df100["Pregnancies"]<0,0,df100["Pregnancies"]))
```

[25]:
```
df100["BloodPressure"]=np.where(df100["BloodPressure"]>104,104,np.
↪where(df100["BloodPressure"]<40.0,40.0,df100["BloodPressure"]))
```

[26]:
```
df100["SkinThickness"]=np.where(df100["SkinThickness"]>45.5,45.5,np.
↪where(df100["SkinThickness"]<10,10,df100["SkinThickness"]))
```

[27]:
```
df100["Glucose"]=np.where(df100["Glucose"]>201,201,np.where(df100["Glucose"]<39.
↪0,39.0,df100["Glucose"]))
```

[28]:
```
df100["Insulin"]=np.where(df100["Insulin"]>272.3,272.3,np.
↪where(df100["Insulin"]<0,0,df100["Insulin"]))
```

[29]:
```
df100["BMI"]=np.where(df100["BMI"]>50.25,50.25,np.where(df100["BMI"]<13.8,13.
↪8,df100["BMI"]))
```

[30]:
```
df100["DiabetesPedigreeFunction"]=np.where(df100["DiabetesPedigreeFunction"]>1.
↪2,1.2,np.
↪where(df100["DiabetesPedigreeFunction"]<0,0,df100["DiabetesPedigreeFunction"]))
```

[31]:
```
df100["Age"]=np.where(df100["Age"]>66.5,66.5,np.
↪where(df100["Age"]<0,0,df100["Age"]))
```

[32]:
```
df1.describe()
```

[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	121.656250	72.386719	27.334635	94.652344	
std	3.369578	30.438286	12.096642	9.229014	105.547598	
min	0.000000	44.000000	24.000000	7.000000	14.000000	
25%	1.000000	99.750000	64.000000	23.000000	30.500000	
50%	3.000000	117.000000	72.000000	23.000000	31.250000	
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	32.450911	0.471876	33.240885	0.348958
std	6.875366	0.331329	11.760232	0.476951
min	18.200000	0.078000	21.000000	0.000000
25%	27.500000	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

[33]:
```
df100.describe()
```

[33]:

	Pregnancies	BloodPressure	SkinThickness	Glucose	Insulin	\
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.834635	72.358073	27.119141	121.656250	86.135026	
std	3.336808	11.697097	8.442060	30.438286	76.275681	
min	0.000000	40.000000	10.000000	44.000000	14.000000	
25%	1.000000	64.000000	23.000000	99.750000	30.500000	
50%	3.000000	72.000000	23.000000	117.000000	31.250000	
75%	6.000000	80.000000	32.000000	140.250000	127.250000	
max	13.000000	104.000000	45.500000	199.000000	272.300000	

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	32.389063	0.458914	33.199870	0.348958
std	6.667627	0.285596	11.628404	0.476951
min	18.200000	0.078000	21.000000	0.000000
25%	27.500000	0.243750	24.000000	0.000000

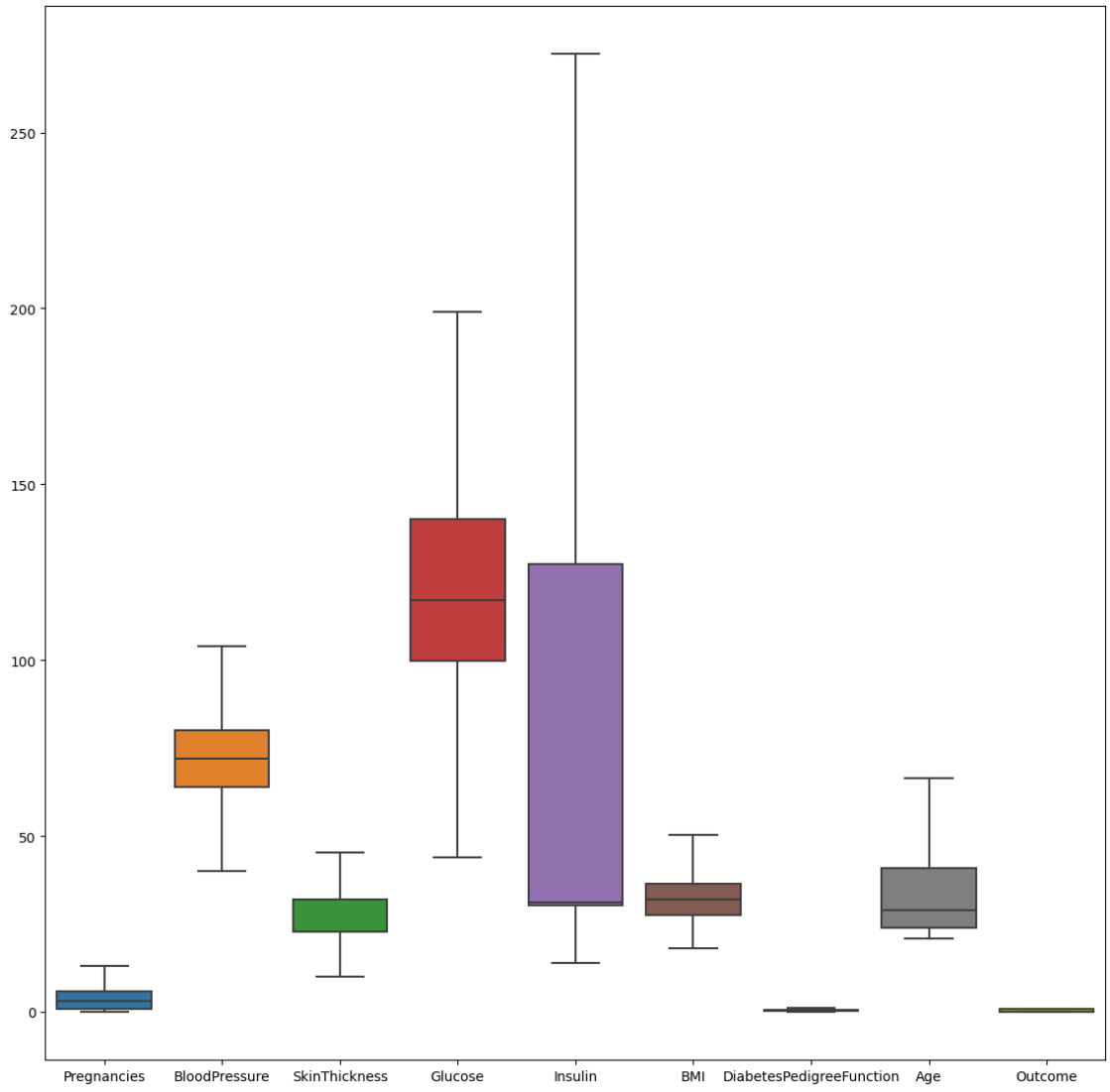
```
50%     32.000000                 0.372500   29.000000    0.000000
75%     36.600000                 0.626250   41.000000    1.000000
max     50.250000                 1.200000   66.500000    1.000000
```

[34]:
```
df100.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   BloodPressure             768 non-null    float64
 2   SkinThickness             768 non-null    float64
 3   Glucose                   768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    int64
dtypes: float64(7), int64(2)
memory usage: 54.1 KB
```

[35]:
```
plt.figure(figsize=(14,14))
sns.boxplot(data=df100)
```

[35]:
```
<Axes: >
```



[36]:
```
df100.describe()
```

[36]:

	Pregnancies	BloodPressure	SkinThickness	Glucose	Insulin
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.834635	72.358073	27.119141	121.656250	86.135026
std	3.336808	11.697097	8.442060	30.438286	76.275681
min	0.000000	40.000000	10.000000	44.000000	14.000000
25%	1.000000	64.000000	23.000000	99.750000	30.500000
50%	3.000000	72.000000	23.000000	117.000000	31.250000
75%	6.000000	80.000000	32.000000	140.250000	127.250000
max	13.000000	104.000000	45.500000	199.000000	272.300000

	BMI	DiabetesPedigreeFunction	Age	Outcome

```
count  768.000000                768.000000  768.000000  768.000000
mean    32.389063                  0.458914   33.199870    0.348958
std      6.667627                  0.285596   11.628404    0.476951
min     18.200000                  0.078000   21.000000    0.000000
25%     27.500000                  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     50.250000                  1.200000   66.500000    1.000000
```

[37]:
```
# boxplot
# plt.figure(figsize=(14,14))
# sns.boxplot(data=df100)
```

[38]:
```
# def outlier(column):
#     q1=column.quantile(0.25)
#     q3=column.quantile(0.75)
#     iqr=q3-q1
#     UF=(q3+1.5*iqr)
#     LF=(q1-1.5*iqr)

#     return column.clip(LF,UF,axis=0)
```

[39]:
```
# df100["Pregnancies"]=outlier(df100["Pregnancies"])
```

[40]:
```
# plt.figure(figsize=(14,14))
# sns.boxplot(data=df100)
```

[41]:
```
# --EDA
```

[42]:
```
df100
```

[42]:
```
     Pregnancies  BloodPressure  SkinThickness  Glucose  Insulin   BMI  \
0              6           72.0           35.0    148.0     30.5  33.6
1              1           66.0           29.0     85.0     30.5  26.6
2              8           64.0           23.0    183.0     30.5  23.3
3              1           66.0           23.0     89.0     94.0  28.1
4              0           40.0           35.0    137.0    168.0  43.1
..           ...            ...            ...      ...      ...   ...
763           10           76.0           45.5    101.0    180.0  32.9
764            2           70.0           27.0    122.0     30.5  36.8
765            5           72.0           23.0    121.0    112.0  26.2
766            1           60.0           23.0    126.0     30.5  30.1
767            1           70.0           31.0     93.0     30.5  30.4

     DiabetesPedigreeFunction   Age  Outcome
0                       0.627  50.0        1
1                       0.351  31.0        0
```

```
2                         0.672  32.0        1
3                         0.167  21.0        0
4                         1.200  33.0        1
..                          …    …         …
763                       0.171  63.0        0
764                       0.340  27.0        0
765                       0.245  30.0        0
766                       0.349  47.0        1
767                       0.315  23.0        0

[768 rows x 9 columns]
```

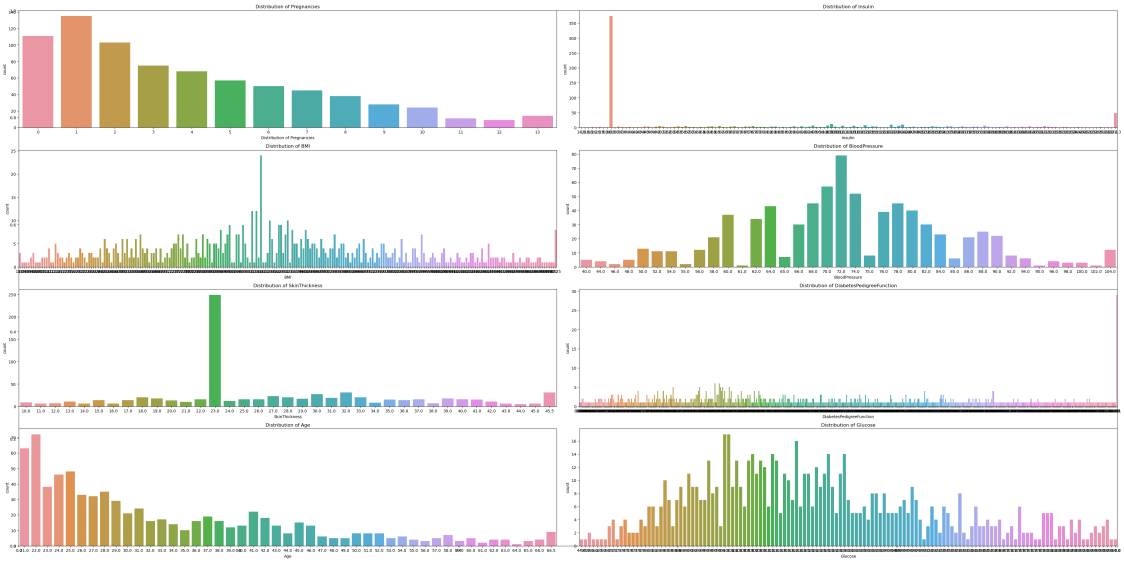
[43]:
```
df100.info()
```

```
<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   BloodPressure             768 non-null    float64
 2   SkinThickness             768 non-null    float64
 3   Glucose                   768 non-null    float64
 4   Insulin                   768 non-null    float64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    int64
dtypes: float64(7), int64(2)
memory usage: 54.1 KB
```

[ ]:
```
# --count plot
```

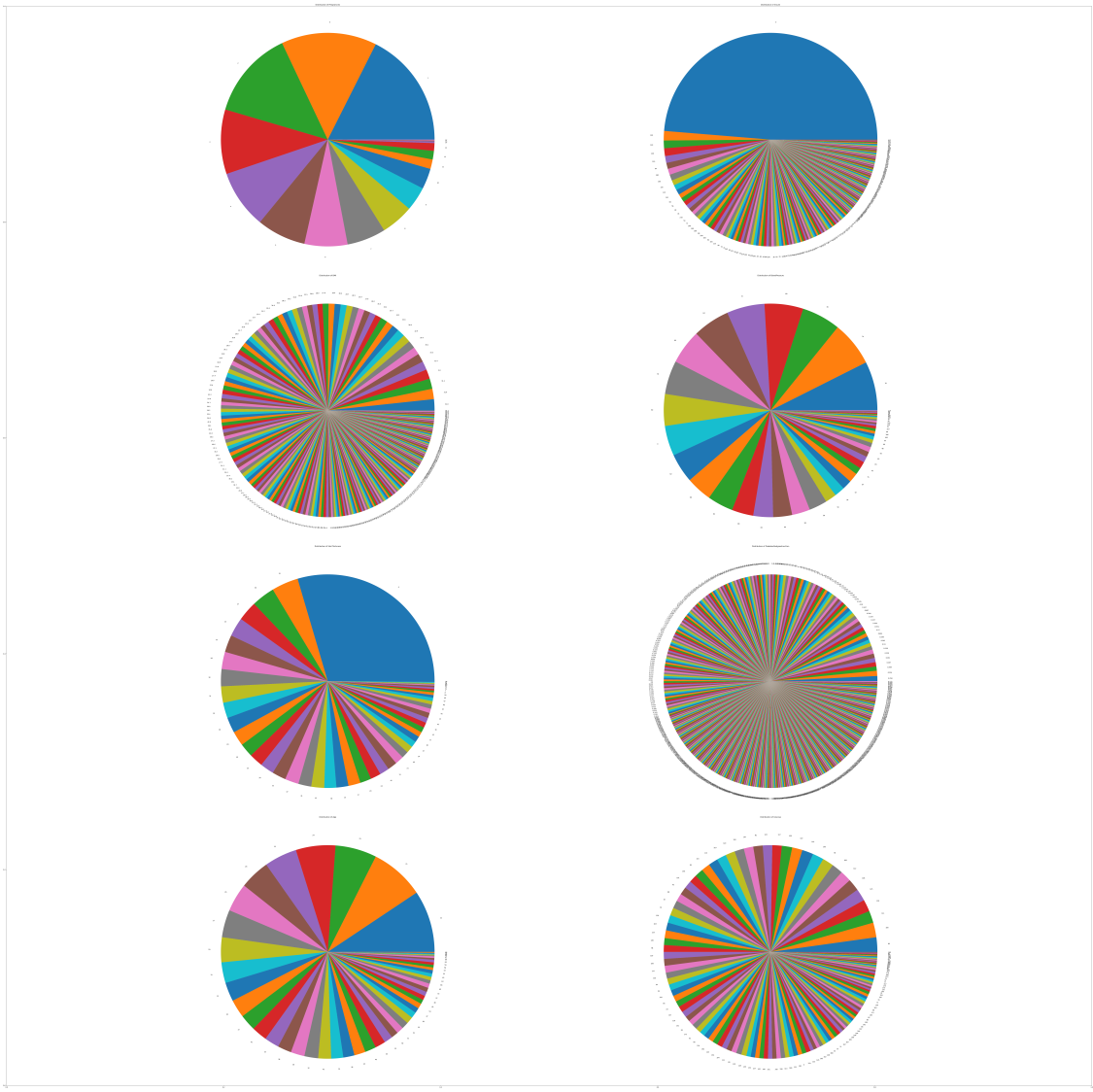
[45]:
```
# -SUBPLOT

plt.subplots(figsize=(40,20))

plt.subplot(4,2,1)
sns.countplot(x="Pregnancies", data=df100)
plt.xlabel("Distribution of Pregnancies")
plt.title("Distribution of Pregnancies")

plt.subplot(4,2,2)
sns.countplot(x="Insulin", data=df100)
plt.title("Distribution of Insulin")

plt.subplot(4,2,3)
```

```
sns.countplot(x="BMI", data=df100)
plt.title("Distribution of BMI")

plt.subplot(4,2,4)
sns.countplot(x="BloodPressure", data=df100)
plt.title("Distribution of BloodPressure")

plt.subplot(4,2,5)
sns.countplot(x="SkinThickness", data=df100)
plt.title("Distribution of SkinThickness")

plt.subplot(4,2,6)
sns.countplot(x="DiabetesPedigreeFunction", data=df100)
plt.title("Distribution of DiabetesPedigreeFunction")

plt.subplot(4,2,7)
sns.countplot(x="Age", data=df100)
plt.title("Distribution of Age")

plt.subplot(4,2,8)
sns.countplot(x="Glucose", data=df100)
plt.title("Distribution of Glucose")

plt.tight_layout()
```



[46]:
```
# -pieplot
```

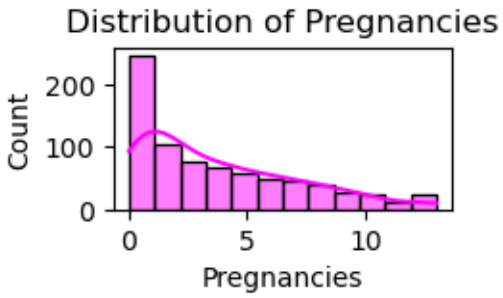
[47]:
```
plt.subplots(figsize=(100,100))

plt.subplot(4,2,1)
plt.pie(x=df["Pregnancies"].value_counts(),labels=df["Pregnancies"].
  ↪value_counts().index,data=df100)
plt.title('Distribution of Pregnancies')

plt.subplot(4,2,2)
plt.title("Distribution of Insulin")
plt.pie(x=df["Insulin"].value_counts(),labels=df["Insulin"].value_counts().
  ↪index,data=df100)

plt.subplot(4,2,3)
plt.title("Distribution of BMI")
plt.pie(x=df["BMI"].value_counts(),labels=df["BMI"].value_counts().
  ↪index,data=df100)

plt.subplot(4,2,4)
plt.title("Distribution of BloodPressure")
plt.pie(x=df["BloodPressure"].value_counts(),labels=df["BloodPressure"].
  ↪value_counts().index,data=df100)

plt.subplot(4,2,5)
plt.pie(x=df["SkinThickness"].value_counts(),labels=df["SkinThickness"].
  ↪value_counts().index,data=df100)
plt.title('Distribution of SkinThickness')

plt.subplot(4,2,6)
plt.title("Distribution of DiabetesPedigreeFunction")
plt.pie(x=df["DiabetesPedigreeFunction"].
  ↪value_counts(),labels=df["DiabetesPedigreeFunction"].value_counts().
  ↪index,data=df100)

plt.subplot(4,2,7)
plt.title("Distribution of Age")
plt.pie(x=df["Age"].value_counts(),labels=df["Age"].value_counts().
  ↪index,data=df100)

plt.subplot(4,2,8)
plt.title("Distribution of Glucose")
plt.pie(x=df["Glucose"].value_counts(),labels=df["Glucose"].value_counts().
  ↪index,data=df100)

plt.tight_layout()
```

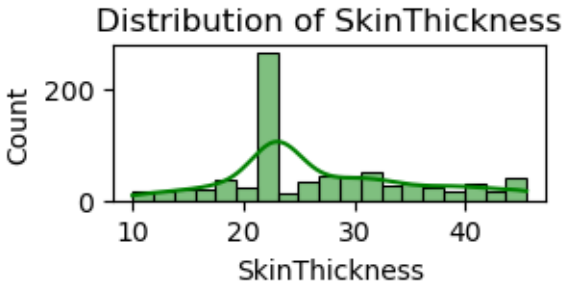


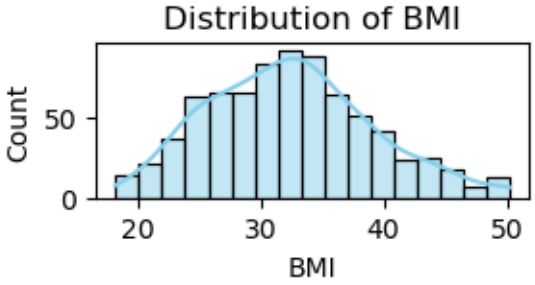
[48]:
```
# -histplot
```

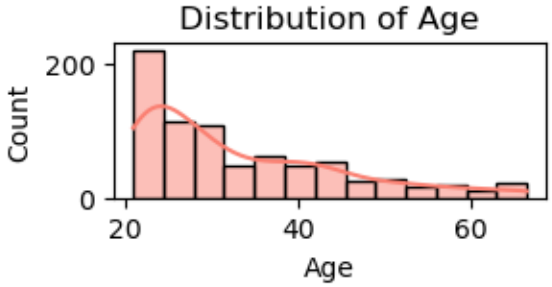
[49]:
```
plt.figure(figsize=(5,5))

plt.subplot(4,2,1)
sns.histplot(df100['Pregnancies'],kde=True,color='magenta')
plt.title('Distribution of Pregnancies')
plt.show()

plt.subplot(4,2,2)
sns.histplot(df100['SkinThickness'],kde=True,color='green')
plt.title('Distribution of SkinThickness')
plt.show()
```

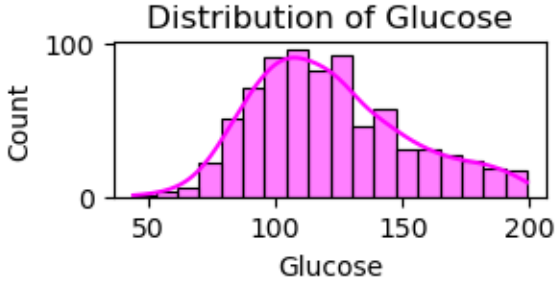
```
plt.subplot(4,2,3)
sns.histplot(df100['BMI'],kde=True,color='skyblue')
plt.title('Distribution of BMI')
plt.show()

plt.subplot(4,2,4)
sns.histplot(df100['Age'],kde=True,color='salmon')
plt.title('Distribution of Age')
plt.show()

plt.subplot(4,2,5)
sns.histplot(df100['Glucose'],kde=True,color='magenta')
plt.title('Distribution of Glucose')
plt.show()

plt.subplot(4,2,6)
sns.histplot(df100['DiabetesPedigreeFunction'],kde=True,color='skyblue')
plt.title('Distribution of DiabetesPedigreeFunction')
plt.show()

plt.subplot(4,2,7)
sns.histplot(df100['BloodPressure'],kde=True,color='green')
plt.title('Distribution of BloodPressure')
plt.show()

plt.subplot(4,2,8)
sns.histplot(df100['Insulin'],kde=True,color='salmon')
plt.title('Distribution of Insulin')
plt.show()
```

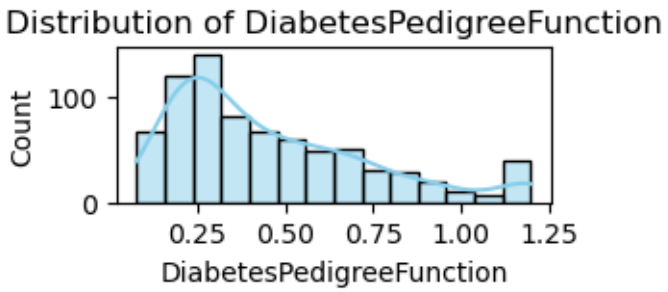


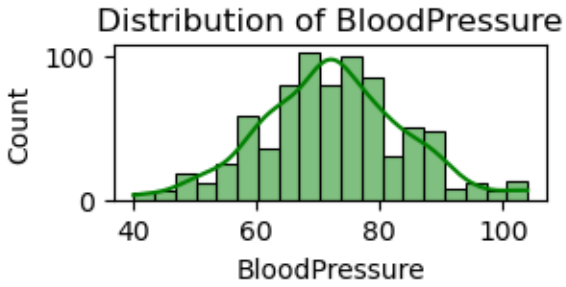

Distribution of SkinThickness
Count
200
0
10
20
30
40
SkinThickness

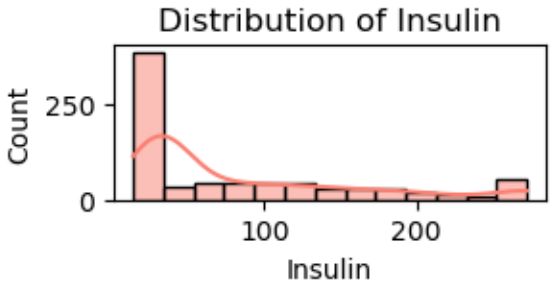

Distribution of BMI
Count
50
0
20
30
40
50
BMI


Distribution of Age
Count
200
0
20
40
60
Age


Distribution of Glucose
Count
100
0
50
100
150
200
Glucose



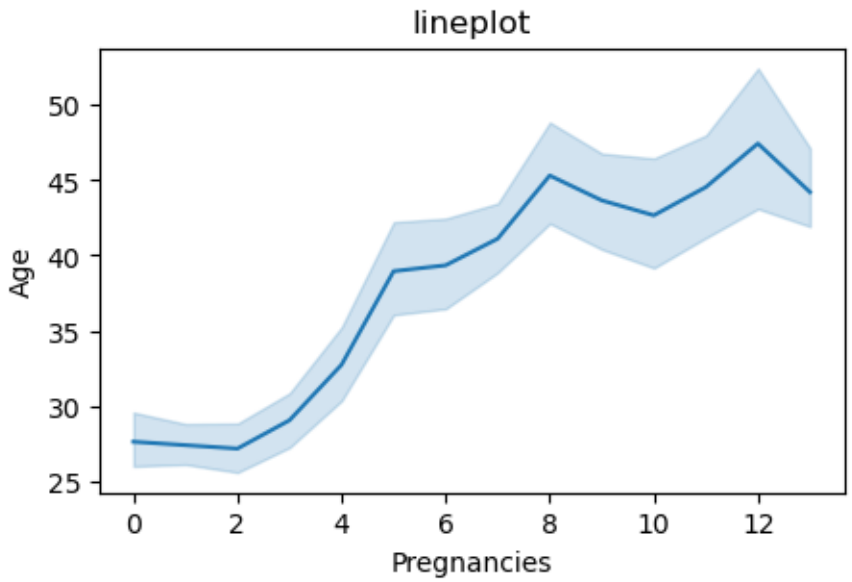




[50]:
```
# -lineplot
```

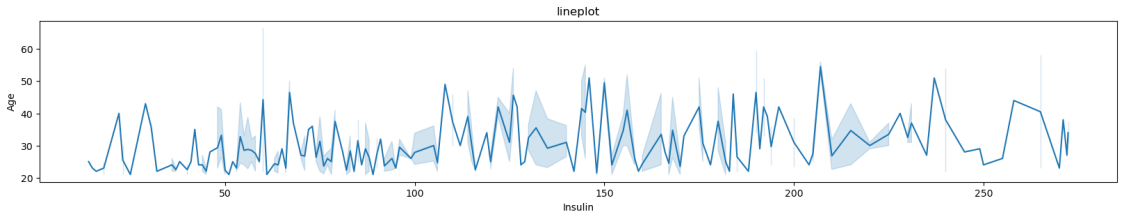
[51]:
```
plt.figure(figsize=(5,3))
sns.lineplot(x="Pregnancies", y="Age", data=df100)

plt.title('lineplot')
```

[51]:
```
Text(0.5, 1.0, 'lineplot')
```



[52]:
```
plt.figure(figsize=(20,3))
sns.lineplot(x="Insulin", y="Age", data=df100)

plt.title('lineplot')
```

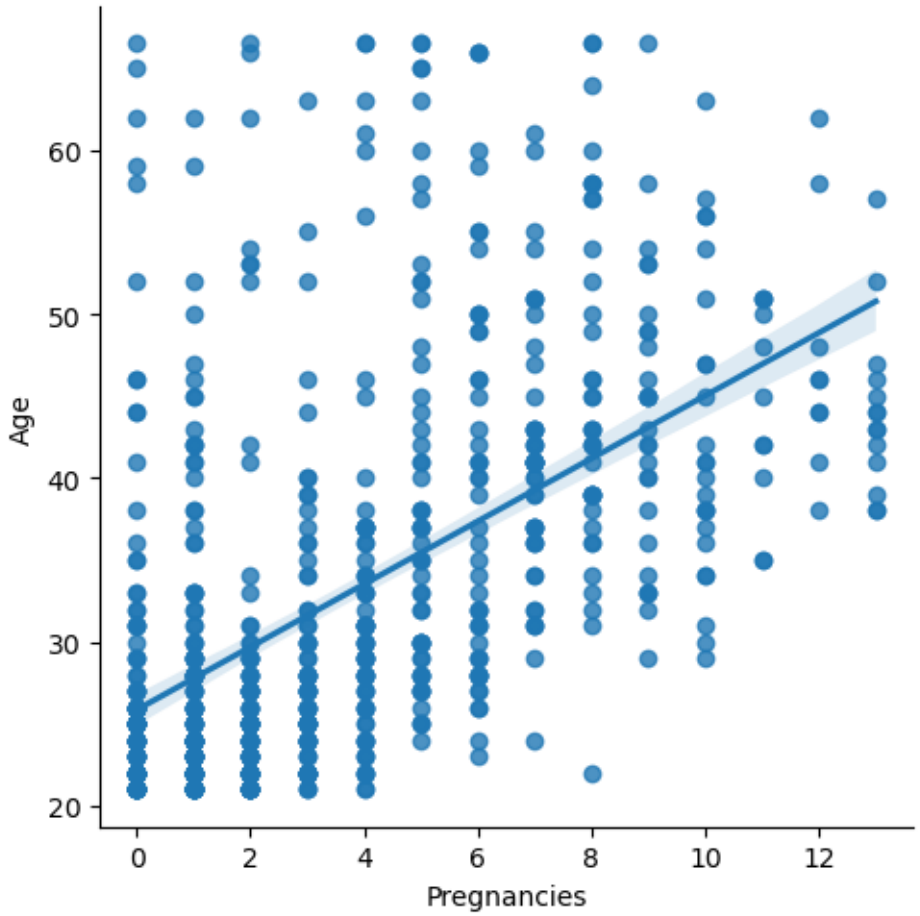
[52]:
```
Text(0.5, 1.0, 'lineplot')
```



[53]:
```
# -scatterplot
```

[54]:
```
sns.lmplot(x='Pregnancies', y='Age', data=df100)
```

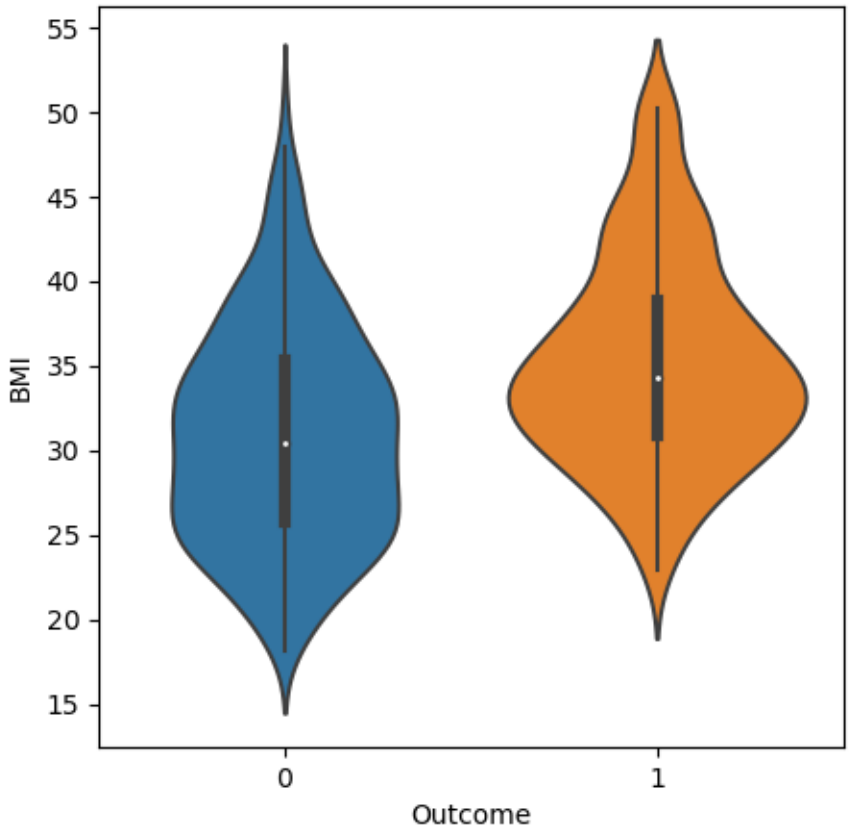
[54]:
```
<seaborn.axisgrid.FacetGrid at 0x270a1515350>
```



[55]:
```
# violinplot
```

[56]:
```
plt.figure(figsize=(5,5))
sns.violinplot(x='Outcome', y='BMI', data=df100)
```

[56]: <Axes: xlabel='Outcome', ylabel='BMI'>



[ ]:

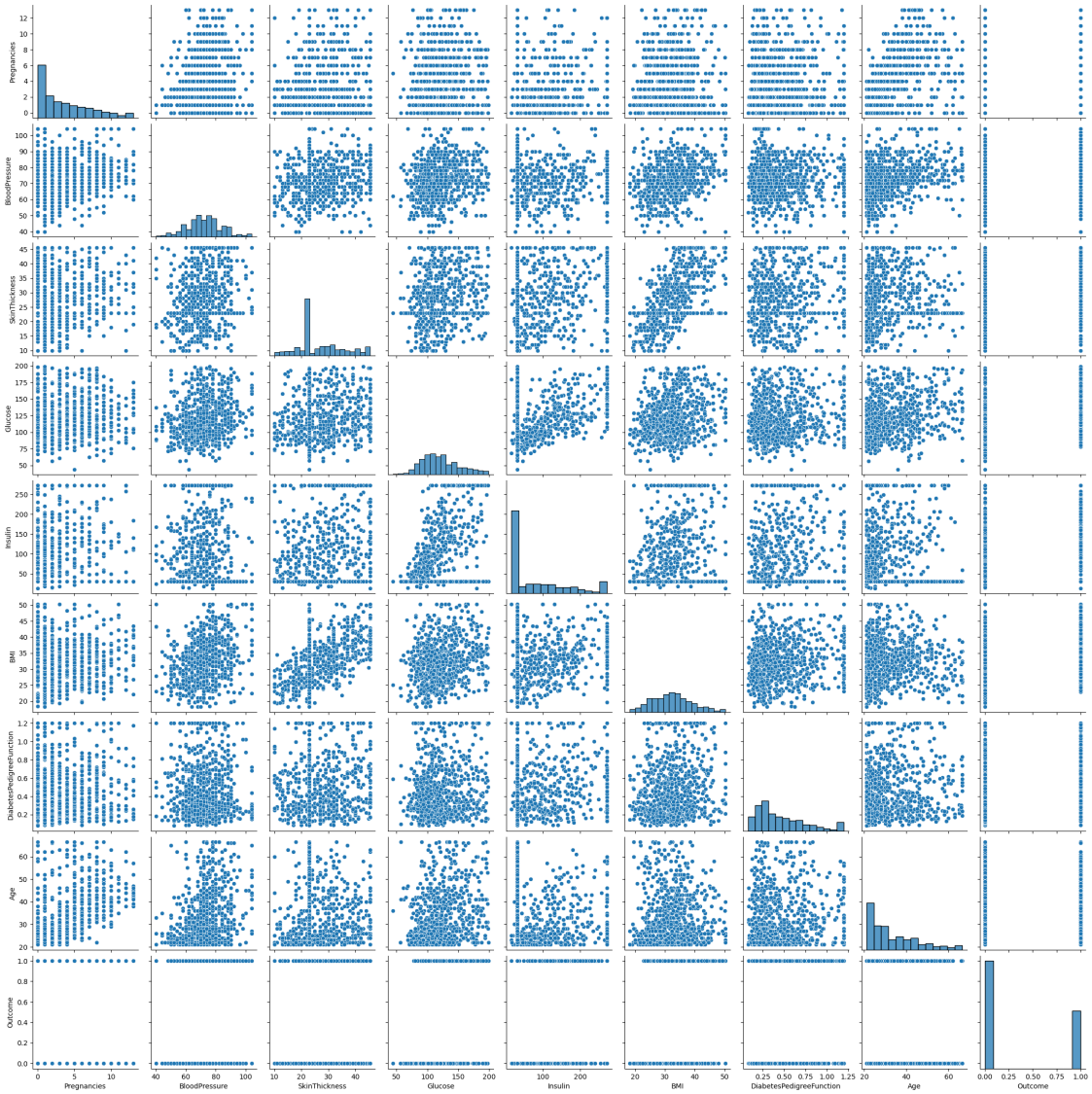
[57]:
```
df200=np.array(df100)
```

[58]:
```
df200
```

[58]:
```
array([[ 6.   , 72.   , 35.   , …,  0.627, 50.   ,  1.   ],
       [ 1.   , 66.   , 29.   , …,  0.351, 31.   ,  0.   ],
       [ 8.   , 64.   , 23.   , …,  0.672, 32.   ,  1.   ],
       …,
       [ 5.   , 72.   , 23.   , …,  0.245, 30.   ,  0.   ],
       [ 1.   , 60.   , 23.   , …,  0.349, 47.   ,  1.   ],
       [ 1.   , 70.   , 31.   , …,  0.315, 23.   ,  0.   ]])
```

[59]:
```
sns.pairplot(df100)
```

[59]:
```
<seaborn.axisgrid.PairGrid at 0x270a2647a10>
```



[60]:
```
df100.corr()
```

[60]:

	Pregnancies	BloodPressure	SkinThickness	Glucose	\
Pregnancies	1.000000	0.211495	0.047354	0.126574	
BloodPressure	0.211495	1.000000	0.159821	0.220199	
SkinThickness	0.047354	0.159821	1.000000	0.161964	
Glucose	0.126574	0.220199	0.161964	1.000000	
Insulin	-0.061574	-0.033848	0.286008	0.338407	
BMI	0.026658	0.286410	0.562480	0.233953	
DiabetesPedigreeFunction	-0.017271	0.012305	0.118976	0.118457	
Age	0.550214	0.332898	0.045650	0.268912	
Outcome	0.219811	0.168971	0.191391	0.492782	

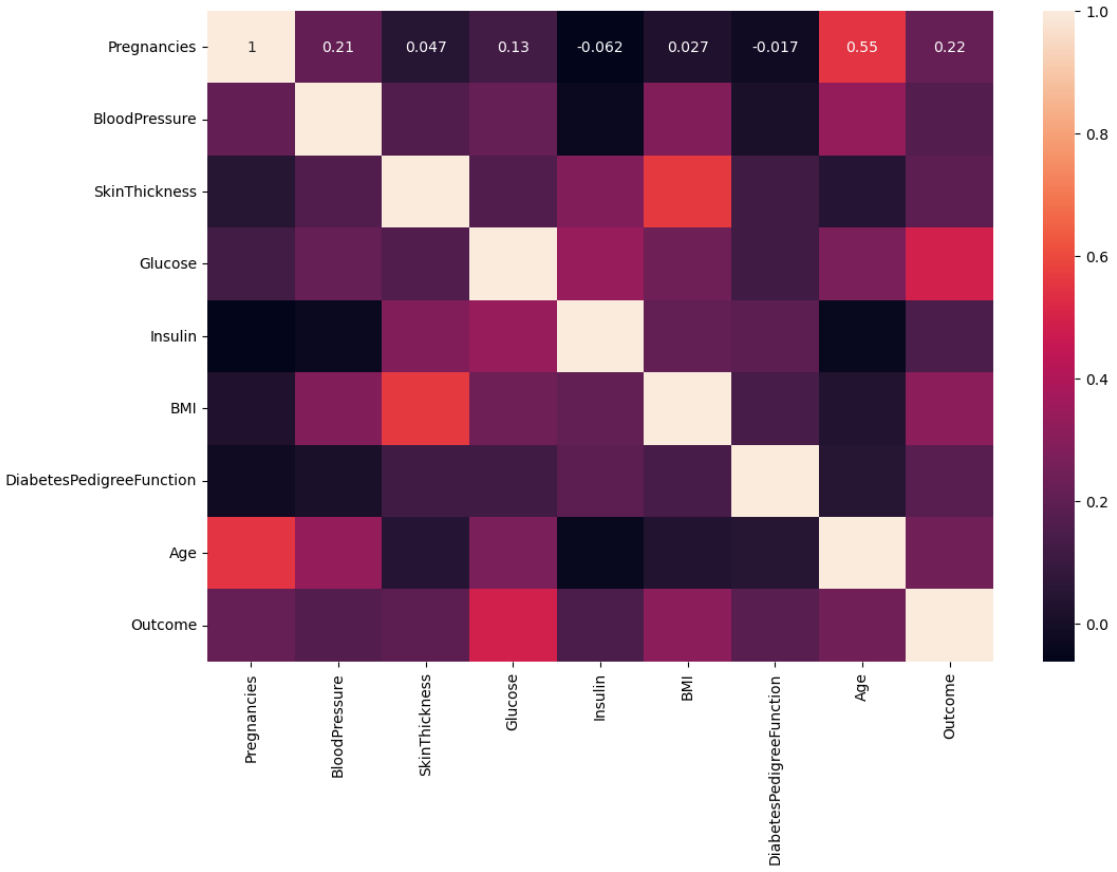
```
                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.061574  0.026658                 -0.017271
BloodPressure             -0.033848  0.286410                  0.012305
SkinThickness              0.286008  0.562480                  0.118976
Glucose                    0.338407  0.233953                  0.118457
Insulin                    1.000000  0.207482                  0.191885
BMI                        0.207482  1.000000                  0.138541
DiabetesPedigreeFunction   0.191885  0.138541                  1.000000
Age                       -0.040458  0.035861                  0.047289
Outcome                    0.147291  0.313030                  0.184969

                                Age   Outcome
Pregnancies                0.550214  0.219811
BloodPressure              0.332898  0.168971
SkinThickness              0.045650  0.191391
Glucose                    0.268912  0.492782
Insulin                   -0.040458  0.147291
BMI                        0.035861  0.313030
DiabetesPedigreeFunction   0.047289  0.184969
Age                        1.000000  0.242702
Outcome                    0.242702  1.000000
```

[61]:
```
df100.corr(method="spearman")
```

[61]:
```
                          Pregnancies  BloodPressure  SkinThickness   Glucose  \
Pregnancies                  1.000000       0.189700       0.038464  0.128795
BloodPressure                0.189700       1.000000       0.164263  0.242639
SkinThickness                0.038464       0.164263       1.000000  0.160745
Glucose                      0.128795       0.242639       0.160745  1.000000
Insulin                     -0.117391      -0.075376       0.269234  0.217522
BMI                          0.000536       0.289738       0.571605  0.225551
DiabetesPedigreeFunction    -0.042812       0.009924       0.108757  0.090131
Age                          0.607192       0.366269       0.115414  0.281873
Outcome                      0.198565       0.170851       0.196878  0.481397

                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.117391  0.000536                 -0.042812
BloodPressure             -0.075376  0.289738                  0.009924
SkinThickness              0.269234  0.571605                  0.108757
Glucose                    0.217522  0.225551                  0.090131
Insulin                    1.000000  0.182310                  0.207280
BMI                        0.182310  1.000000                  0.134440
DiabetesPedigreeFunction   0.207280  0.134440                  1.000000
Age                       -0.094859  0.120971                  0.043147
Outcome                    0.074389  0.307320                  0.175851

                                Age   Outcome
```

```
Pregnancies                0.607192  0.198565
BloodPressure              0.366269  0.170851
SkinThickness              0.115414  0.196878
Glucose                    0.281873  0.481397
Insulin                   -0.094859  0.074389
BMI                        0.120971  0.307320
DiabetesPedigreeFunction   0.043147  0.175851
Age                        1.000000  0.309053
Outcome                    0.309053  1.000000
```

[62]:
```
# corr = df100.corr() #number
# plt.figure(figsize=(10,7))
# sns.heatmap((corr),annot=True)
plt.figure(figsize=(12,8))
# plt.savefig('correlation_heatmap.png', dpi=300)
sns.heatmap(df100.corr(),annot=True)
plt.show()
```



[63]:
```
#model building
```

[64]:
```
# -train_test_split
```

[65]:
```
X=df100.drop(("Outcome"),axis=1)
```

[66]:
```
X.head()
```

[66]:
```
   Pregnancies  BloodPressure  SkinThickness  Glucose  Insulin   BMI  \
0            6           72.0           35.0    148.0     30.5  33.6
1            1           66.0           29.0     85.0     30.5  26.6
2            8           64.0           23.0    183.0     30.5  23.3
3            1           66.0           23.0     89.0     94.0  28.1
4            0           40.0           35.0    137.0    168.0  43.1

   DiabetesPedigreeFunction   Age
0                     0.627  50.0
1                     0.351  31.0
2                     0.672  32.0
3                     0.167  21.0
4                     1.200  33.0
```

[67]:
```
Y=df100["Outcome"]
```

[68]:
```
Y.head()
```

[68]:
```
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
```

[69]:
```
from sklearn.preprocessing import StandardScaler,MinMaxScaler,RobustScaler␣
 ↪#scaling
scaler=StandardScaler()
X_scaled=scaler.fit_transform(X)
```

[70]:
```
X_scaled
```

[70]:
```
array([[ 0.64935582, -0.03063207,  0.93413159, …,  0.18173282,
         0.58892732,  1.44569096],
       [-0.8500587 , -0.54391414,  0.22294144, …, -0.86880015,
        -0.37810147, -0.189304  ],
       [ 1.24912163, -0.71500816, -0.4882487 , …, -1.3640514 ,
         0.74659506, -0.10325164],
       …,
       [ 0.34947292, -0.03063207, -0.4882487 , …, -0.9288306 ,
        -0.74949659, -0.27535637],
```

```
       [-0.8500587 , -1.05719621, -0.4882487 , …, -0.34353366,
        -0.38510892,  1.18753386],
       [-0.8500587 , -0.20172609,  0.46000482, …, -0.29851082,
        -0.50423566, -0.87772293]])
```

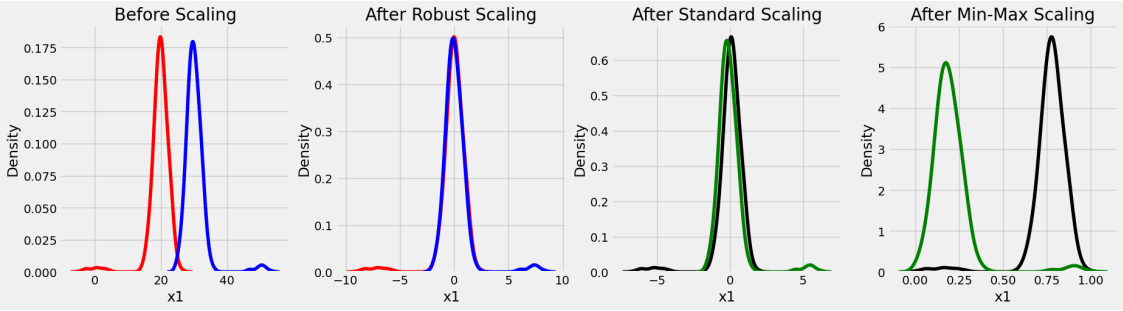
[71]:
```
X_scaled=pd.DataFrame(X_scaled,columns=X.columns)
```

[72]:
```
X_scaled.head()
```

[72]:
```
   Pregnancies  BloodPressure  SkinThickness   Glucose   Insulin       BMI  \
0     0.649356      -0.030632       0.934132  0.866045 -0.729869  0.181733
1    -0.850059      -0.543914       0.222941 -1.205066 -0.729869 -0.868800
2     1.249122      -0.715008      -0.488249  2.016662 -0.729869 -1.364051
3    -0.850059      -0.543914      -0.488249 -1.073567  0.103180 -0.643686
4    -1.149942      -2.768136       0.934132  0.504422  1.073977  1.607456

   DiabetesPedigreeFunction       Age
0                  0.588927  1.445691
1                 -0.378101 -0.189304
2                  0.746595 -0.103252
3                 -1.022787 -1.049828
4                  2.596563 -0.017199
```

[73]:
```
from sklearn import preprocessing
import matplotlib
import matplotlib.pyplot as plt

%matplotlib inline
matplotlib.style.use('fivethirtyeight')
# data
x = pd.DataFrame({
    # Distribution with lower outliers
    'x1': np.concatenate([np.random.normal(20, 2, 1000), np.random.normal(1, 2,␣
 ↪25)]),
    # Distribution with higher outliers
    'x2': np.concatenate([np.random.normal(30, 2, 1000), np.random.normal(50,␣
 ↪2, 25)]),
})
np.random.normal

scaler = preprocessing.RobustScaler()
robust_df = scaler.fit_transform(x)
robust_df = pd.DataFrame(robust_df, columns =['x1', 'x2'])

scaler = preprocessing.StandardScaler()
standard_df = scaler.fit_transform(x)
standard_df = pd.DataFrame(standard_df, columns =['x1', 'x2'])
```

```
scaler = preprocessing.MinMaxScaler()
minmax_df = scaler.fit_transform(x)
minmax_df = pd.DataFrame(minmax_df, columns =['x1', 'x2'])

fig, (ax1, ax2, ax3, ax4) = plt.subplots(ncols = 4, figsize =(20, 5))
ax1.set_title('Before Scaling')

sns.kdeplot(x['x1'], ax = ax1, color ='r')
sns.kdeplot(x['x2'], ax = ax1, color ='b')
ax2.set_title('After Robust Scaling')

sns.kdeplot(robust_df['x1'], ax = ax2, color ='red')
sns.kdeplot(robust_df['x2'], ax = ax2, color ='blue')
ax3.set_title('After Standard Scaling')

sns.kdeplot(standard_df['x1'], ax = ax3, color ='black')
sns.kdeplot(standard_df['x2'], ax = ax3, color ='g')
ax4.set_title('After Min-Max Scaling')

sns.kdeplot(minmax_df['x1'], ax = ax4, color ='black')
sns.kdeplot(minmax_df['x2'], ax = ax4, color ='g')
plt.show()
```



[74]:
```
from sklearn.model_selection import train_test_split
```

[75]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X_scaled,Y,test_size=0.
↪2,random_state=10,stratify=Y)
```

[76]:
```
X_train.shape
```

[76]: (614, 8)

[77]:
```
X_test.shape
```

[77]: (154, 8)

[78]:
```
#LogisticRegression
from sklearn.linear_model import LogisticRegression
```

[79]:
```
log_mod=LogisticRegression()
```

[80]:
```
log_mod.fit(X_train,Y_train)
```

[80]: LogisticRegression()

[81]:
```
Y_train_predict=log_mod.predict(X_train)
```

[82]:
```
log_mod.fit(X_test,Y_test)
```

[82]: LogisticRegression()

[83]:
```
probability_score=log_mod.predict_proba(X_test) #ROC AUC
probability_score
```

[83]:
```
array([[0.95704623, 0.04295377],
       [0.82050038, 0.17949962],
       [0.84605321, 0.15394679],
       [0.96989006, 0.03010994],
       [0.94866885, 0.05133115],
       [0.77078194, 0.22921806],
       [0.72137412, 0.27862588],
       [0.73262216, 0.26737784],
       [0.85716421, 0.14283579],
       [0.4753479 , 0.5246521 ],
       [0.76160899, 0.23839101],
       [0.94947334, 0.05052666],
       [0.23877412, 0.76122588],
       [0.04778177, 0.95221823],
       [0.98322123, 0.01677877],
       [0.30183919, 0.69816081],
       [0.85465878, 0.14534122],
       [0.69869231, 0.30130769],
       [0.71130221, 0.28869779],
       [0.02641639, 0.97358361],
       [0.96572824, 0.03427176],
       [0.40041764, 0.59958236],
       [0.78047113, 0.21952887],
       [0.74701588, 0.25298412],
       [0.85871279, 0.14128721],
       [0.88560584, 0.11439416],
       [0.47472655, 0.52527345],
```

```
[0.77833336, 0.22166664],
[0.04999129, 0.95000871],
[0.22224632, 0.77775368],
[0.62962006, 0.37037994],
[0.03973807, 0.96026193],
[0.80698334, 0.19301666],
[0.33825581, 0.66174419],
[0.75913522, 0.24086478],
[0.82017985, 0.17982015],
[0.53025006, 0.46974994],
[0.97055718, 0.02944282],
[0.22717744, 0.77282256],
[0.83606949, 0.16393051],
[0.91663797, 0.08336203],
[0.78246791, 0.21753209],
[0.74754186, 0.25245814],
[0.96153822, 0.03846178],
[0.91547705, 0.08452295],
[0.45679961, 0.54320039],
[0.08212349, 0.91787651],
[0.73439846, 0.26560154],
[0.96494163, 0.03505837],
[0.9077469 , 0.0922531 ],
[0.97579828, 0.02420172],
[0.77283102, 0.22716898],
[0.86984714, 0.13015286],
[0.21069402, 0.78930598],
[0.15054227, 0.84945773],
[0.97038006, 0.02961994],
[0.65574651, 0.34425349],
[0.74126578, 0.25873422],
[0.83205673, 0.16794327],
[0.61968957, 0.38031043],
[0.64876729, 0.35123271],
[0.94341249, 0.05658751],
[0.37008718, 0.62991282],
[0.46470697, 0.53529303],
[0.93277631, 0.06722369],
[0.45288963, 0.54711037],
[0.21863989, 0.78136011],
[0.94332021, 0.05667979],
[0.92263922, 0.07736078],
[0.78450142, 0.21549858],
[0.31995226, 0.68004774],
[0.10034365, 0.89965635],
[0.30157363, 0.69842637],
[0.82718999, 0.17281001],
```

```
[0.19233888, 0.80766112],
[0.88308422, 0.11691578],
[0.4046128 , 0.5953872 ],
[0.79631132, 0.20368868],
[0.38637085, 0.61362915],
[0.52576075, 0.47423925],
[0.57170626, 0.42829374],
[0.22823914, 0.77176086],
[0.34975724, 0.65024276],
[0.9246407 , 0.0753593 ],
[0.90172015, 0.09827985],
[0.77824226, 0.22175774],
[0.58438208, 0.41561792],
[0.95193645, 0.04806355],
[0.21336166, 0.78663834],
[0.95344928, 0.04655072],
[0.17393561, 0.82606439],
[0.8873581 , 0.1126419 ],
[0.68788198, 0.31211802],
[0.90159954, 0.09840046],
[0.66130489, 0.33869511],
[0.82257706, 0.17742294],
[0.130898  , 0.869102  ],
[0.70026216, 0.29973784],
[0.72718326, 0.27281674],
[0.46612682, 0.53387318],
[0.48768832, 0.51231168],
[0.58389065, 0.41610935],
[0.20652592, 0.79347408],
[0.69371094, 0.30628906],
[0.97007107, 0.02992893],
[0.72657458, 0.27342542],
[0.11334964, 0.88665036],
[0.30502667, 0.69497333],
[0.16323697, 0.83676303],
[0.9129742 , 0.0870258 ],
[0.98521472, 0.01478528],
[0.22631905, 0.77368095],
[0.84574088, 0.15425912],
[0.93466372, 0.06533628],
[0.75998486, 0.24001514],
[0.92698124, 0.07301876],
[0.75497891, 0.24502109],
[0.87384318, 0.12615682],
[0.02206062, 0.97793938],
[0.80756643, 0.19243357],
[0.79728601, 0.20271399],
```

```
       [0.90776953, 0.09223047],
       [0.77584108, 0.22415892],
       [0.90268094, 0.09731906],
       [0.8597524 , 0.1402476 ],
       [0.9682034 , 0.0317966 ],
       [0.6551159 , 0.3448841 ],
       [0.32902527, 0.67097473],
       [0.86325668, 0.13674332],
       [0.74079623, 0.25920377],
       [0.50218413, 0.49781587],
       [0.95094442, 0.04905558],
       [0.54642126, 0.45357874],
       [0.87604345, 0.12395655],
       [0.93321624, 0.06678376],
       [0.91937275, 0.08062725],
       [0.32131194, 0.67868806],
       [0.09912862, 0.90087138],
       [0.73441711, 0.26558289],
       [0.88235066, 0.11764934],
       [0.72195381, 0.27804619],
       [0.05895611, 0.94104389],
       [0.27084296, 0.72915704],
       [0.96860926, 0.03139074],
       [0.95227615, 0.04772385],
       [0.14514298, 0.85485702],
       [0.97903915, 0.02096085],
       [0.89953965, 0.10046035],
       [0.88471773, 0.11528227],
       [0.44218313, 0.55781687],
       [0.15412373, 0.84587627],
       [0.94318025, 0.05681975],
       [0.81963488, 0.18036512],
       [0.7953916 , 0.2046084 ]])
```

[84]:
```
from sklearn.metrics import roc_auc_score

# predict probabilities
probability_prediction_positive = log_mod.predict_proba(X_test)[:,1]
```

[85]:
```
probability_prediction_positive
```

[85]:
```
array([0.04295377, 0.17949962, 0.15394679, 0.03010994, 0.05133115,
       0.22921806, 0.27862588, 0.26737784, 0.14283579, 0.5246521 ,
       0.23839101, 0.05052666, 0.76122588, 0.95221823, 0.01677877,
       0.69816081, 0.14534122, 0.30130769, 0.28869779, 0.97358361,
       0.03427176, 0.59958236, 0.21952887, 0.25298412, 0.14128721,
       0.11439416, 0.52527345, 0.22166664, 0.95000871, 0.77775368,
```

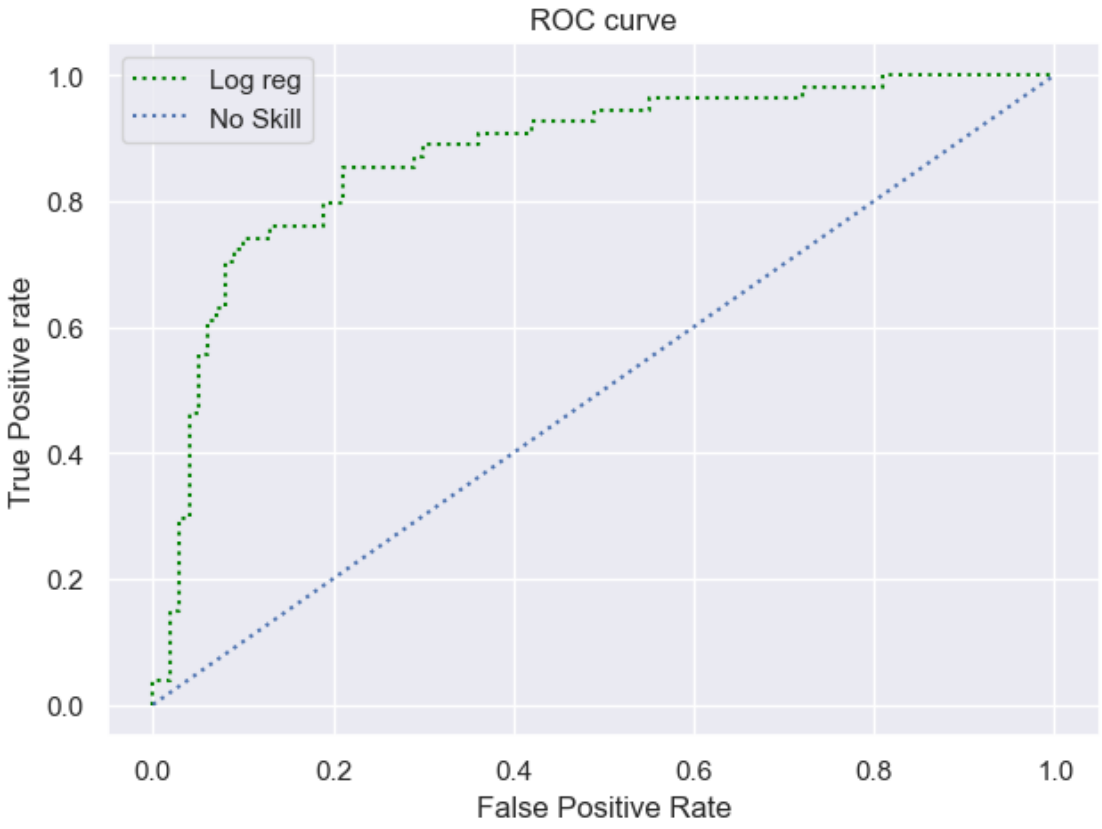
```
       0.37037994, 0.96026193, 0.19301666, 0.66174419, 0.24086478,
       0.17982015, 0.46974994, 0.02944282, 0.77282256, 0.16393051,
       0.08336203, 0.21753209, 0.25245814, 0.03846178, 0.08452295,
       0.54320039, 0.91787651, 0.26560154, 0.03505837, 0.0922531 ,
       0.02420172, 0.22716898, 0.13015286, 0.78930598, 0.84945773,
       0.02961994, 0.34425349, 0.25873422, 0.16794327, 0.38031043,
       0.35123271, 0.05658751, 0.62991282, 0.53529303, 0.06722369,
       0.54711037, 0.78136011, 0.05667979, 0.07736078, 0.21549858,
       0.68004774, 0.89965635, 0.69842637, 0.17281001, 0.80766112,
       0.11691578, 0.5953872 , 0.20368868, 0.61362915, 0.47423925,
       0.42829374, 0.77176086, 0.65024276, 0.0753593 , 0.09827985,
       0.22175774, 0.41561792, 0.04806355, 0.78663834, 0.04655072,
       0.82606439, 0.1126419 , 0.31211802, 0.09840046, 0.33869511,
       0.17742294, 0.869102  , 0.29973784, 0.27281674, 0.53387318,
       0.51231168, 0.41610935, 0.79347408, 0.30628906, 0.02992893,
       0.27342542, 0.88665036, 0.69497333, 0.83676303, 0.0870258 ,
       0.01478528, 0.77368095, 0.15425912, 0.06533628, 0.24001514,
       0.07301876, 0.24502109, 0.12615682, 0.97793938, 0.19243357,
       0.20271399, 0.09223047, 0.22415892, 0.09731906, 0.1402476 ,
       0.0317966 , 0.3448841 , 0.67097473, 0.13674332, 0.25920377,
       0.49781587, 0.04905558, 0.45357874, 0.12395655, 0.06678376,
       0.08062725, 0.67868806, 0.90087138, 0.26558289, 0.11764934,
       0.27804619, 0.94104389, 0.72915704, 0.03139074, 0.04772385,
       0.85485702, 0.02096085, 0.10046035, 0.11528227, 0.55781687,
       0.84587627, 0.05681975, 0.18036512, 0.2046084 ])
```

[86]:
```
# auc scores
auc_score1 = roc_auc_score(Y_test, probability_prediction_positive)


from sklearn.metrics import roc_curve
fpr1, tpr1, thresh1 = roc_curve(Y_test, probability_prediction_positive,␣
 ↪pos_label=1)

print("AUC SCORE:",auc_score1)
```

```
AUC SCORE: 0.8738888888888889
```

[87]:
```
import matplotlib.pyplot as plt
import seaborn as sns
# plt.style.use('seaborn')
sns.set_theme()
# plot roc curves
plt.plot(fpr1, tpr1, linestyle='dotted',color='green', label='Log reg')

# plot no skill roc curve
plt.plot([0, 1], [0, 1], linestyle='dotted', label='No Skill')
```

```
# title
plt.title('ROC curve')

# x label
plt.xlabel('False Positive Rate')
# y label
plt.ylabel('True Positive rate')

plt.legend(loc='best')

plt.savefig('ROC', dpi=400)
plt.show()
```



[88]:
```
from sklearn.metrics import␣
↪accuracy_score,confusion_matrix,classification_report
from sklearn.metrics import mean_squared_error , r2_score
```

[89]:
```
accuracy_score(Y_train,Y_train_predict)
```

[89]: 0.7687296416938111

[90]:
```
confusion_matrix(Y_train,Y_train_predict)
```

[90]:
```
array([[356,  44],
       [ 98, 116]], dtype=int64)
```

[91]:
```
print(classification_report(Y_train,Y_train_predict))
```

```
              precision    recall  f1-score   support

           0       0.78      0.89      0.83       400
           1       0.72      0.54      0.62       214

    accuracy                           0.77       614
   macro avg       0.75      0.72      0.73       614
weighted avg       0.76      0.77      0.76       614
```

[92]:
```
Y_predict_test=log_mod.predict(X_test)
```

[93]:
```
accuracy_score(Y_test,Y_predict_test)
```

[93]: 0.8376623376623377

[94]:
```
confusion_matrix(Y_test,Y_predict_test)
```

[94]:
```
array([[91,  9],
       [16, 38]], dtype=int64)
```

[95]:
```
print(classification_report(Y_test,Y_predict_test))
```

```
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       100
           1       0.81      0.70      0.75        54

    accuracy                           0.84       154
   macro avg       0.83      0.81      0.82       154
weighted avg       0.84      0.84      0.83       154
```

[96]:
```
from sklearn.svm import SVC,SVR
from sklearn.metrics import mean_squared_error , r2_score
```

[97]:
```
svc_model=SVC(kernel="poly",degree=8,C=0.1,gamma=0.01,verbose=True) #kernel=poly
svc_model.fit(X_train,Y_train)
```

```
Y_pred_train=svc_model.predict(X_train)
print("accuracy score train",accuracy_score(Y_train,Y_pred_train))

Y_pred=svc_model.predict(X_test)
print("accuracy score test",accuracy_score(Y_test,Y_pred))

print("confusion matrix of train\n", confusion_matrix(Y_train,Y_pred_train))
print("confusion matrix of train\n", confusion_matrix(Y_test,Y_pred))
```

```
[LibSVM]accuracy score train 0.6514657980456026
accuracy score test 0.6493506493506493
confusion matrix of train
 [[400   0]
 [214   0]]
confusion matrix of train
 [[100   0]
 [ 54   0]]
```

[98]:
```
svc_model=SVC(kernel="rbf",degree=3,C=10,gamma=10) #kernel=rbf
svc_model.fit(X_train,Y_train)
Y_pred_train=svc_model.predict(X_train)
print("accuracy score train",accuracy_score(Y_train,Y_pred_train))

Y_pred=svc_model.predict(X_test)
print("accuracy score test",accuracy_score(Y_test,Y_pred))

print("confusion matrix of train\n", confusion_matrix(Y_train,Y_pred_train))
print("confusion matrix of train\n", confusion_matrix(Y_test,Y_pred))
```

```
accuracy score train 1.0
accuracy score test 0.6493506493506493
confusion matrix of train
 [[400   0]
 [  0 214]]
confusion matrix of train
 [[100   0]
 [ 54   0]]
```

[99]:
```
svc_model=SVC(kernel="sigmoid",degree=3,C=0.1,gamma=1) #kernel=sigmoid
svc_model.fit(X_train,Y_train)
Y_pred_train=svc_model.predict(X_train)
print("accuracy score train",accuracy_score(Y_train,Y_pred_train))

Y_pred=svc_model.predict(X_test)
print("accuracy score test",accuracy_score(Y_test,Y_pred))

print("confusion matrix of train\n", confusion_matrix(Y_train,Y_pred_train))
```

```
print("confusion matrix of train\n", confusion_matrix(Y_test,Y_pred))
```

```
accuracy score train 0.6351791530944625
accuracy score test 0.7532467532467533
confusion matrix of train
 [[294 106]
 [118  96]]
confusion matrix of train
 [[83 17]
 [21 33]]
```

[100]:
```
svc_model=SVC(kernel="linear",degree=3,C=15,gamma=10,probability=True)
 ↪#kernel=linear
svc_model.fit(X_train,Y_train)
Y_pred_train=svc_model.predict(X_train)
print("accuracy score train",accuracy_score(Y_train,Y_pred_train))

Y_pred=svc_model.predict(X_test)
print("accuracy score test",accuracy_score(Y_test,Y_pred))

print("confusion matrix of train\n", confusion_matrix(Y_train,Y_pred_train))
print("confusion matrix of train\n", confusion_matrix(Y_test,Y_pred))
```

```
accuracy score train 0.762214983713355
accuracy score test 0.8181818181818182
confusion matrix of train
 [[352  48]
 [ 98 116]]
confusion matrix of train
 [[89 11]
 [17 37]]
```

[ ]:
```

```

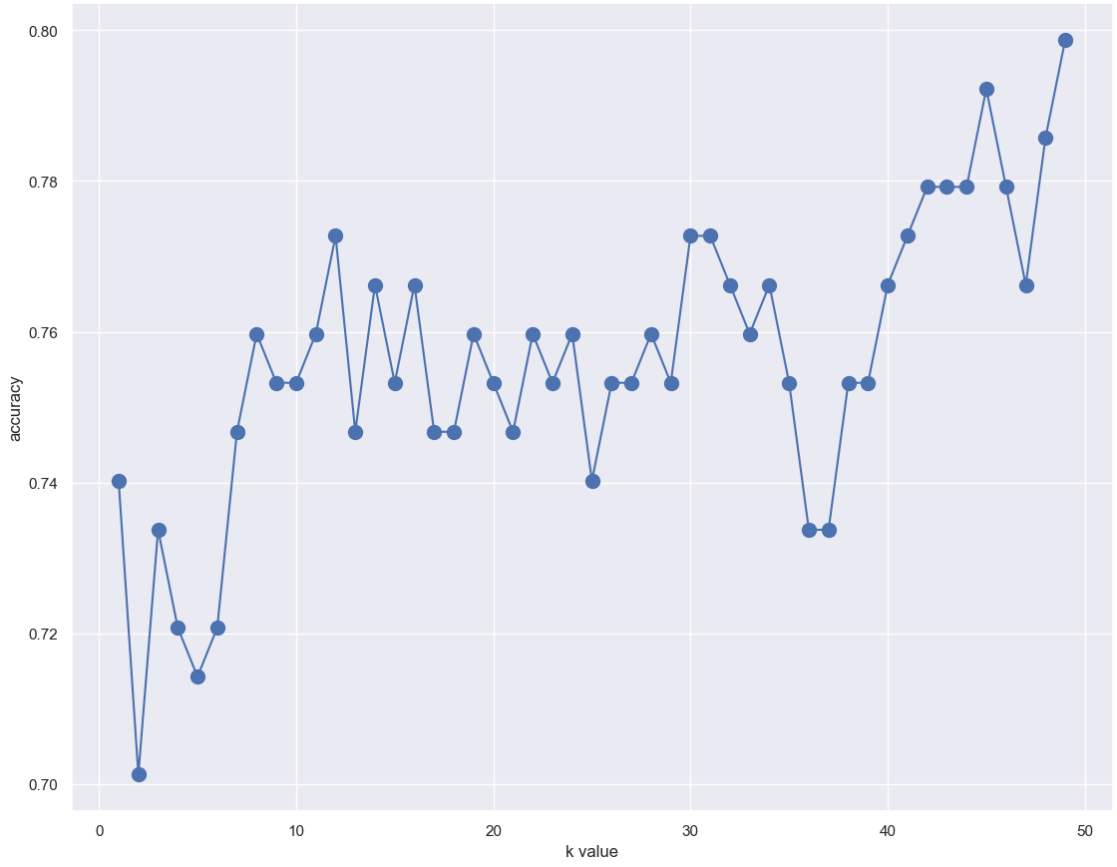
[101]:
```
from sklearn.neighbors import KNeighborsClassifier,KNeighborsRegressor #knn
```

[102]:
```
from sklearn.metrics import
 ↪accuracy_score,confusion_matrix,classification_report
```

[103]:
```
best_accuracy=[]
for k in range(1,50):
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,Y_train)
    Y_pred=knn.predict(X_test)
    accuracy=accuracy_score(Y_test,Y_pred)
    best_accuracy.append(accuracy)
```

```
plt.figure(figsize=(12,10))
plt.plot(range(1,50),best_accuracy,marker="o",markersize=10)
plt.ylabel("accuracy")
plt.xlabel("k value")
```

[103]: Text(0.5, 0, 'k value')



[104]:
```
#--value of k can be 14 or 16
```

[105]:
```
knn=KNeighborsClassifier(n_neighbors=16)
```

[106]:
```
knn.fit(X_train,Y_train)
```

[106]: KNeighborsClassifier(n_neighbors=16)

[107]:
```
Y_pred_train=knn.predict(X_train)
Y_pred_train
```

[107]:
```
array([1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

[108]:
```
Y_train
```

[108]:
```
332    1
111    1
64     1
742    0
101    0
      ..
735    0
475    0
46     0
767    0
709    1
Name: Outcome, Length: 614, dtype: int64
```

[109]:
```
accuracy_train=accuracy_score(Y_train,Y_pred_train)
print("accuracy of train is ",accuracy_train)
```

```
confusion_matrix(Y_train,Y_pred_train)
```

```
accuracy of train is  0.7899022801302932
```

[109]:
```
array([[368,  32],
       [ 97, 117]], dtype=int64)
```

[110]:
```
Y_pred_test=knn.predict(X_test)
accuracy_test=accuracy_score(Y_test,Y_pred_test)
print('accuracy of test is ', accuracy_test)
confusion_matrix(Y_test,Y_pred_test)
```

```
accuracy of test is  0.7662337662337663
```

[110]:
```
array([[87, 13],
       [23, 31]], dtype=int64)
```

[111]:
```
Y_pred_test
```

[111]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0],
      dtype=int64)
```

[112]:
```
probability=knn.predict_proba(X_test)
probability
```

[112]:
```
array([[0.75  , 0.25  ],
       [0.6875, 0.3125],
       [1.    , 0.    ],
       [1.    , 0.    ],
       [1.    , 0.    ],
       [0.6875, 0.3125],
       [0.5625, 0.4375],
       [0.875 , 0.125 ],
       [0.5625, 0.4375],
       [0.625 , 0.375 ],
       [0.9375, 0.0625],
       [0.9375, 0.0625],
       [0.5625, 0.4375],
       [0.3125, 0.6875],
       [1.    , 0.    ],
       [0.1875, 0.8125],
```

```
[0.8125, 0.1875],
[0.6875, 0.3125],
[0.5   , 0.5   ],
[0.25  , 0.75  ],
[1.    , 0.    ],
[0.4375, 0.5625],
[0.5   , 0.5   ],
[0.625 , 0.375 ],
[0.875 , 0.125 ],
[0.4375, 0.5625],
[0.75  , 0.25  ],
[1.    , 0.    ],
[0.375 , 0.625 ],
[0.125 , 0.875 ],
[0.625 , 0.375 ],
[0.1875, 0.8125],
[0.5625, 0.4375],
[0.3125, 0.6875],
[0.6875, 0.3125],
[0.875 , 0.125 ],
[0.4375, 0.5625],
[1.    , 0.    ],
[0.75  , 0.25  ],
[0.75  , 0.25  ],
[0.875 , 0.125 ],
[0.875 , 0.125 ],
[0.6875, 0.3125],
[0.9375, 0.0625],
[0.875 , 0.125 ],
[0.5   , 0.5   ],
[0.4375, 0.5625],
[0.375 , 0.625 ],
[0.8125, 0.1875],
[1.    , 0.    ],
[1.    , 0.    ],
[0.4375, 0.5625],
[0.875 , 0.125 ],
[0.5625, 0.4375],
[0.375 , 0.625 ],
[0.8125, 0.1875],
[0.8125, 0.1875],
[0.3125, 0.6875],
[0.5   , 0.5   ],
[0.4375, 0.5625],
[0.6875, 0.3125],
[1.    , 0.    ],
[0.75  , 0.25  ],
```

```
[0.4375, 0.5625],
[0.5   , 0.5   ],
[0.3125, 0.6875],
[0.625 , 0.375 ],
[1.    , 0.    ],
[1.    , 0.    ],
[0.75  , 0.25  ],
[0.3125, 0.6875],
[0.1875, 0.8125],
[0.1875, 0.8125],
[0.75  , 0.25  ],
[0.3125, 0.6875],
[0.5625, 0.4375],
[0.4375, 0.5625],
[0.6875, 0.3125],
[0.375 , 0.625 ],
[0.4375, 0.5625],
[0.8125, 0.1875],
[0.5625, 0.4375],
[0.25  , 0.75  ],
[0.9375, 0.0625],
[0.875 , 0.125 ],
[0.6875, 0.3125],
[0.5625, 0.4375],
[0.75  , 0.25  ],
[0.375 , 0.625 ],
[0.75  , 0.25  ],
[0.375 , 0.625 ],
[0.5   , 0.5   ],
[0.6875, 0.3125],
[0.875 , 0.125 ],
[0.4375, 0.5625],
[0.5   , 0.5   ],
[0.375 , 0.625 ],
[0.6875, 0.3125],
[0.5   , 0.5   ],
[0.3125, 0.6875],
[0.5   , 0.5   ],
[0.625 , 0.375 ],
[0.625 , 0.375 ],
[0.5625, 0.4375],
[1.    , 0.    ],
[0.625 , 0.375 ],
[0.125 , 0.875 ],
[0.3125, 0.6875],
[0.375 , 0.625 ],
[1.    , 0.    ],
```

```
       [1.    , 0.    ],
       [0.375 , 0.625 ],
       [0.75  , 0.25  ],
       [1.    , 0.    ],
       [0.6875, 0.3125],
       [1.    , 0.    ],
       [0.5   , 0.5   ],
       [0.6875, 0.3125],
       [0.4375, 0.5625],
       [0.6875, 0.3125],
       [0.75  , 0.25  ],
       [0.9375, 0.0625],
       [0.6875, 0.3125],
       [0.875 , 0.125 ],
       [0.875 , 0.125 ],
       [1.    , 0.    ],
       [0.6875, 0.3125],
       [0.5625, 0.4375],
       [0.875 , 0.125 ],
       [0.3125, 0.6875],
       [0.4375, 0.5625],
       [0.9375, 0.0625],
       [0.625 , 0.375 ],
       [0.8125, 0.1875],
       [0.5625, 0.4375],
       [1.    , 0.    ],
       [0.375 , 0.625 ],
       [0.3125, 0.6875],
       [0.5625, 0.4375],
       [1.    , 0.    ],
       [0.6875, 0.3125],
       [0.25  , 0.75  ],
       [0.4375, 0.5625],
       [0.9375, 0.0625],
       [0.625 , 0.375 ],
       [0.25  , 0.75  ],
       [1.    , 0.    ],
       [0.9375, 0.0625],
       [0.875 , 0.125 ],
       [0.6875, 0.3125],
       [0.3125, 0.6875],
       [0.875 , 0.125 ],
       [0.875 , 0.125 ],
       [0.875 , 0.125 ]])
```

[113]:
```
from sklearn.tree import DecisionTreeRegressor #DecisionTreeRegressor
# Instantiate model with 1000 decision trees
```

```
dt = DecisionTreeRegressor()
# Train the model on training data
dt.fit(X_train, Y_train);
Y_predict = dt.predict(X_test)
accuracy_train=accuracy_score(Y_train,Y_pred_train)
print("accuracy of train is ",accuracy_train)
confusion_matrix(Y_train,Y_pred_train)

Y_pred_test=dt.predict(X_test)
accuracy_test=accuracy_score(Y_test,Y_pred_test)
print('accuracy of test is ', accuracy_test)
confusion_matrix(Y_test,Y_pred_test)
```

```
accuracy of train is  0.7899022801302932
accuracy of test is  0.6948051948051948
```

[113]:
```
array([[75, 25],
       [22, 32]], dtype=int64)
```

[114]:
```
from sklearn.ensemble import RandomForestRegressor
# Instantiate model with 1000 decision trees

rf = RandomForestRegressor(n_estimators = 1000, random_state = 70)
# Train the model on training data
rf.fit(X_train, Y_train);
y_predict = rf.predict(X_test)
print('The R2score for Random Forest Regressor is ' , r2_score( Y_test ,␣
 ↪Y_predict ))
```

```
The R2score for Random Forest Regressor is  -0.34037037037037066
```

[115]:
```
featureImportance = pd.Series(rf.feature_importances_ , index = X_train.
 ↪columns).sort_values(ascending = True)
featureImportance
```

[115]:
```
Insulin                     0.044904
SkinThickness               0.056205
BloodPressure               0.079350
Pregnancies                 0.079525
DiabetesPedigreeFunction    0.116787
Age                         0.142847
BMI                         0.160052
Glucose                     0.320332
dtype: float64
```

[ ]:
```

```

[ ]: