In [46]:

```
import numpy as np
import pandas as pd

import seaborn as sns
sns.set(rc={'figure.figsize':(6,4)})
import matplotlib.pyplot as plt
%matplotlib inline

from tqdm import tqdm
import random
import pickle
import time

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import PowerTransformer
from sklearn.preprocessing import Normalizer

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import log_loss
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix
from sklearn import metrics

# for ignore warnings
import warnings
warnings.filterwarnings("ignore")

plot_data_list = []
```

In [31]:
```
df = pd.read_csv('Dataset\Pima diabetes_csv.csv')
df.head()
```

Out[31]:

	preg	plas	pres	skin	insu	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	tested_positive
1	1	85	66	29	0	26.6	0.351	31	tested_negative
2	8	183	64	0	0	23.3	0.672	32	tested_positive
3	1	89	66	23	94	28.1	0.167	21	tested_negative
4	0	137	40	35	168	43.1	2.288	33	tested_positive

EDA

In [32]:
```
df.shape
```

Out[32]:
```
(768, 9)
```

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   preg    768 non-null    int64  
 1   plas    768 non-null    int64  
 2   pres    768 non-null    int64  
 3   skin    768 non-null    int64  
 4   insu    768 non-null    int64  
 5   mass    768 non-null    float64
 6   pedi    768 non-null    float64
 7   age     768 non-null    int64  
 8   class   768 non-null    object 
dtypes: float64(2), int64(6), object(1)
memory usage: 54.1+ KB
```

In [34]:
```
round(df.describe(),2)
```

Out[34]:

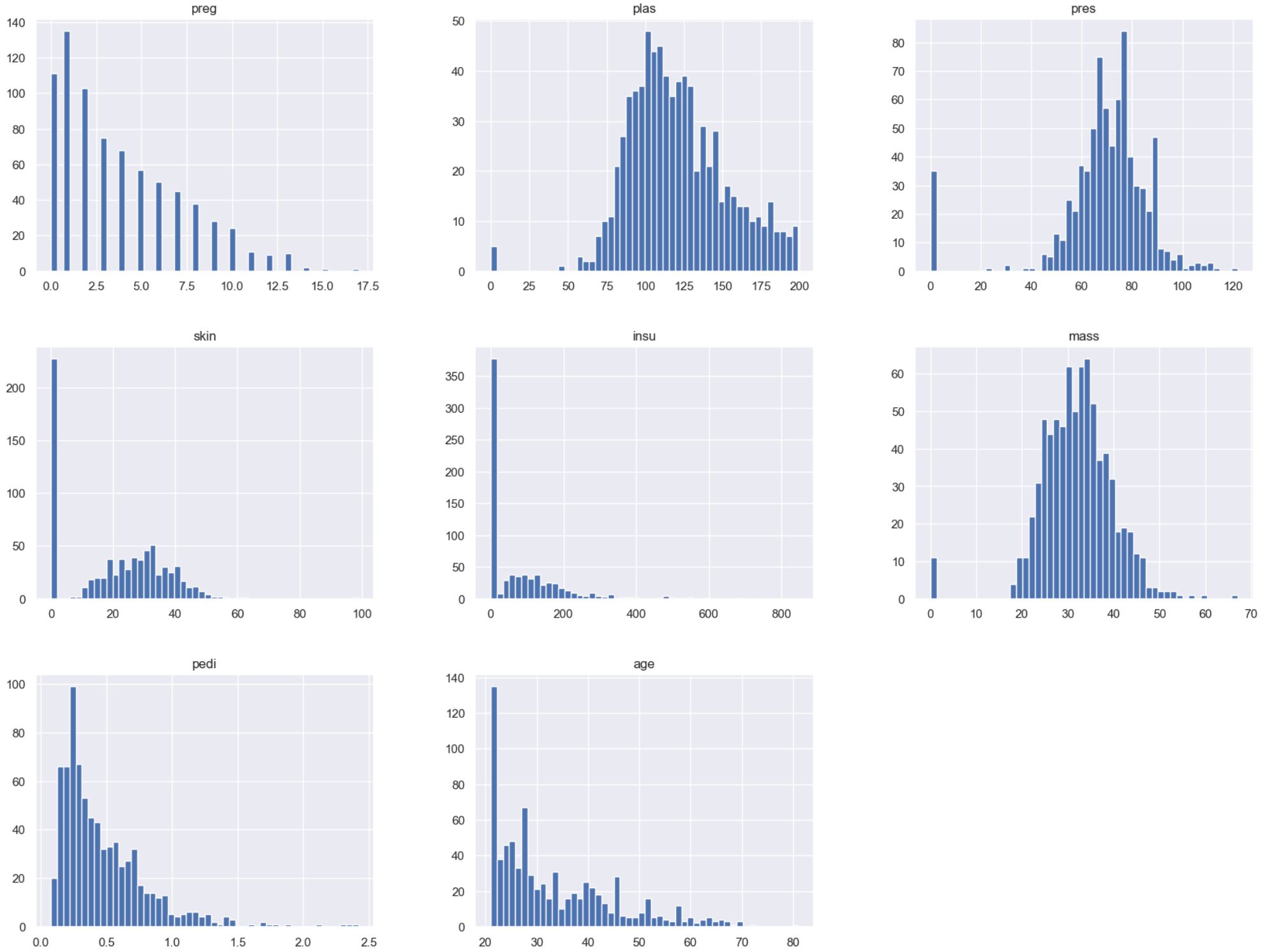
	preg	plas	pres	skin	insu	mass	pedi	age
count	768.00	768.00	768.00	768.00	768.00	768.00	768.00	768.00
mean	3.85	120.89	69.11	20.54	79.80	31.99	0.47	33.24
std	3.37	31.97	19.36	15.95	115.24	7.88	0.33	11.76
min	0.00	0.00	0.00	0.00	0.00	0.00	0.08	21.00
25%	1.00	99.00	62.00	0.00	0.00	27.30	0.24	24.00
50%	3.00	117.00	72.00	23.00	30.50	32.00	0.37	29.00
75%	6.00	140.25	80.00	32.00	127.25	36.60	0.63	41.00
max	17.00	199.00	122.00	99.00	846.00	67.10	2.42	81.00

In [35]:
```
# check null
df.isnull().sum()
```

Out[35]:
```
preg     0
plas     0
pres     0
skin     0
insu     0
mass     0
pedi     0
age      0
class    0
dtype: int64
```

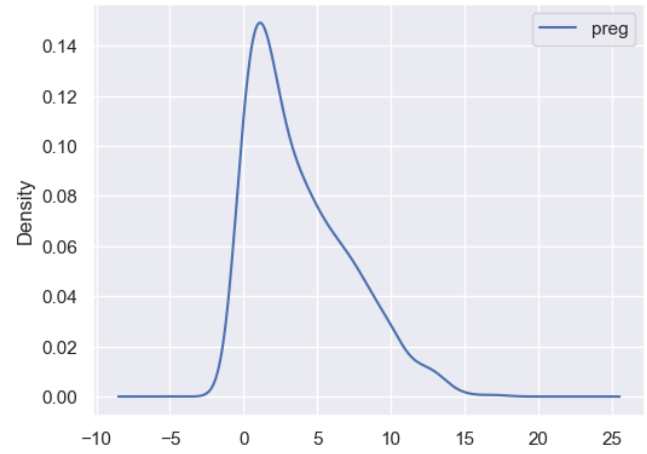
In [36]:

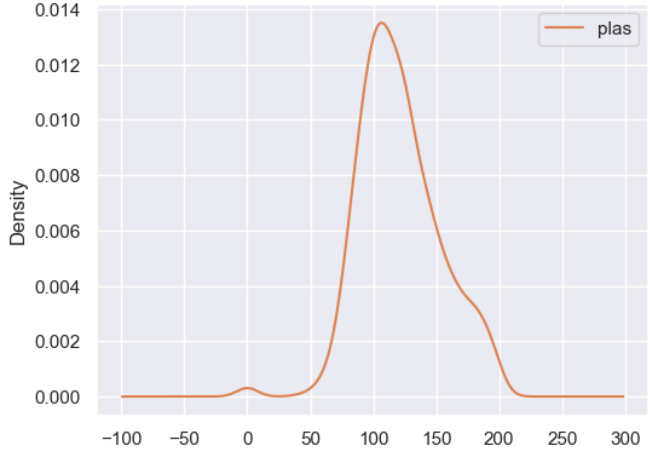
```
df.hist(bins=50, figsize=(20, 15))
plt.show()
```

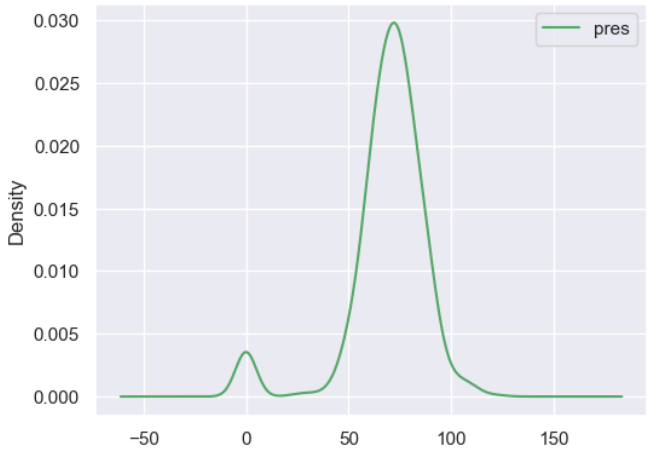

preg
plas
pres
skin
insu
mass
pedi
age

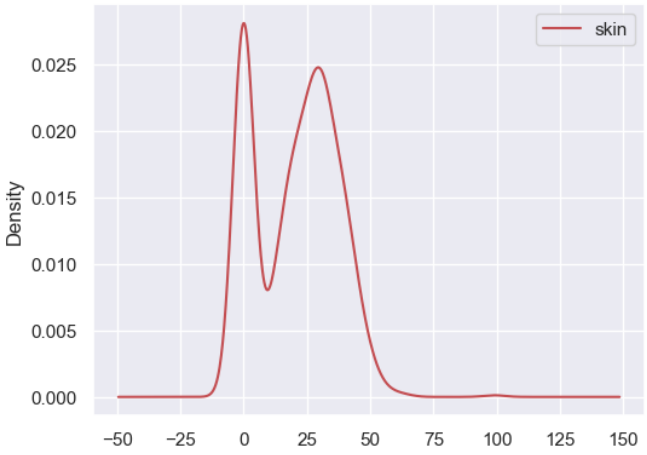
In [37]:

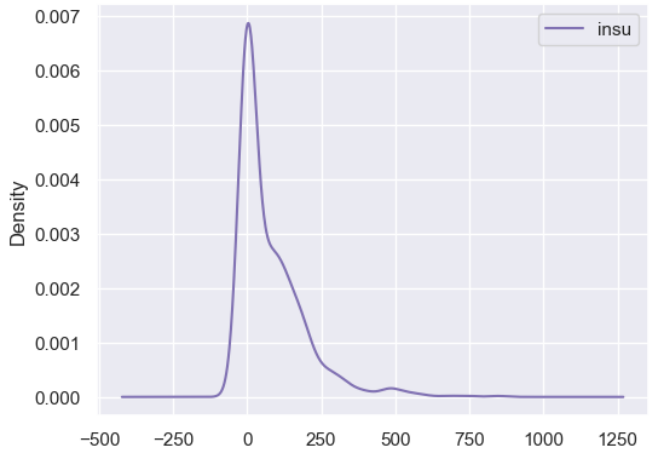
```
# Density plots for all attributes to visualize the distribution of each attribute
df.plot(kind='density', subplots=True, layout=(3,3), figsize=(20, 15), sharex=False)
plt.show()
```

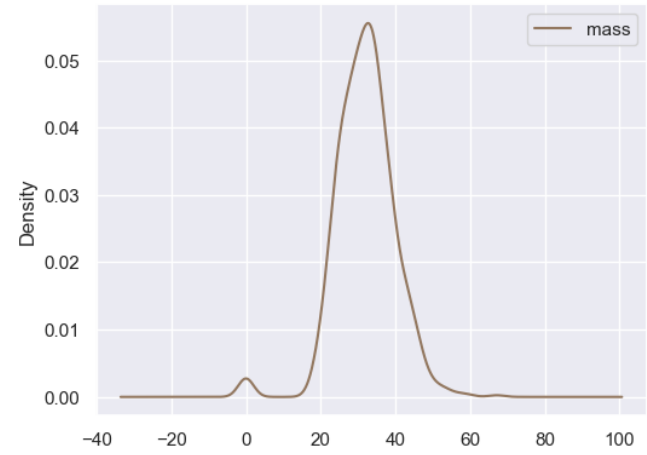

preg
Density
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−10
−5
0
5
10
15
20
25

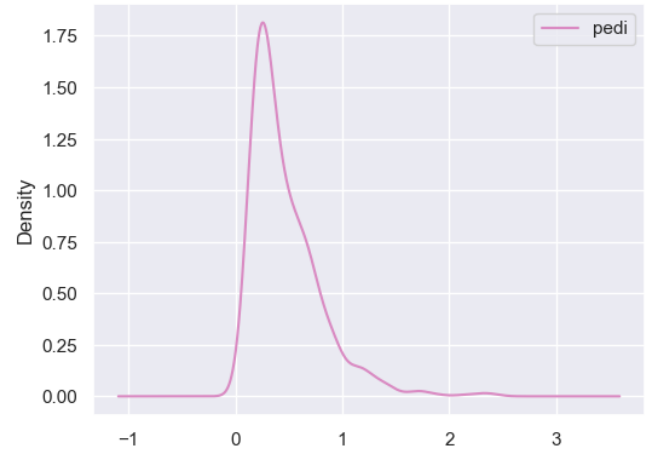

plas
Density
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
−100
−50
0
50
100
150
200
250
300

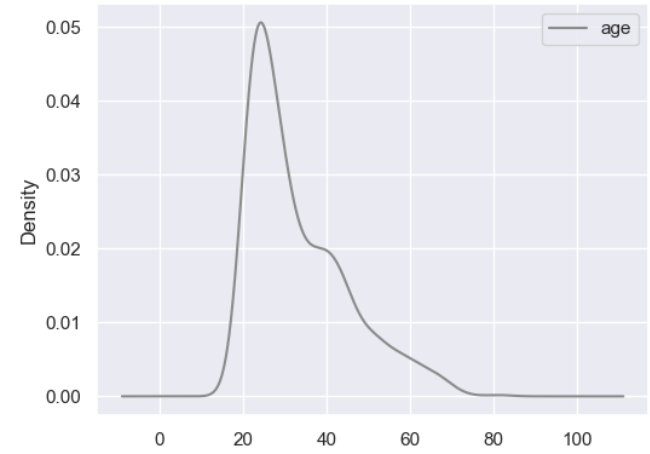

pres
Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−50
0
50
100
150


skin
Density
0.025
0.020
0.015
0.010
0.005
0.000
−50
−25
0
25
50
75
100
125
150


insu
Density
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
−500
−250
0
250
500
750
1000
1250

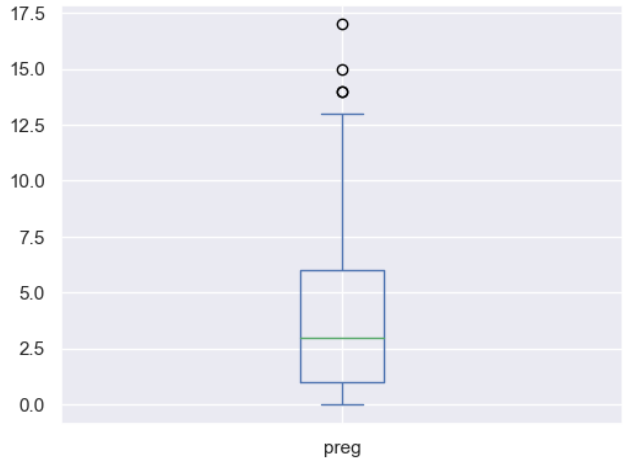

mass
Density
0.05
0.04
0.03
0.02
0.01
0.00
−40
−20
0
20
40
60
80
100

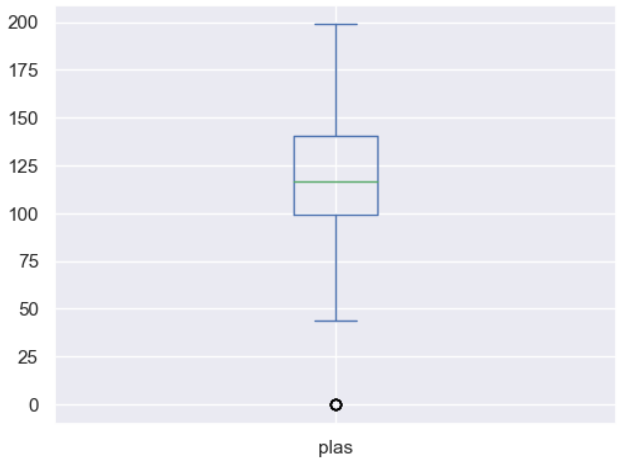

pedi
Density
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1
0
1
2
3

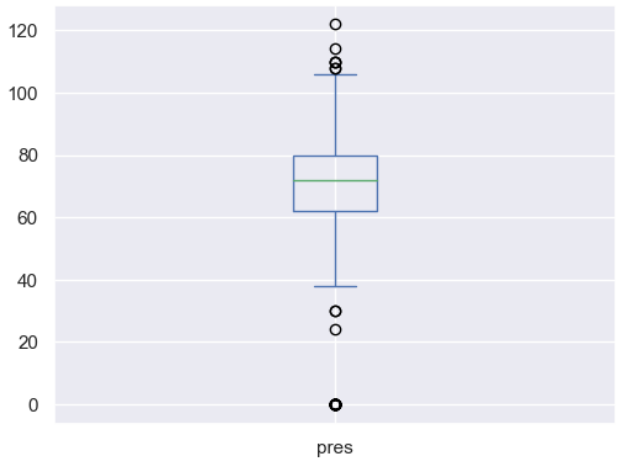

age
Density
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
100

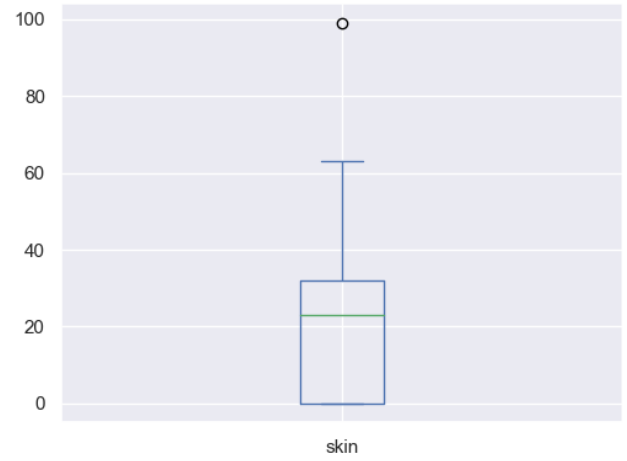
In [38]:

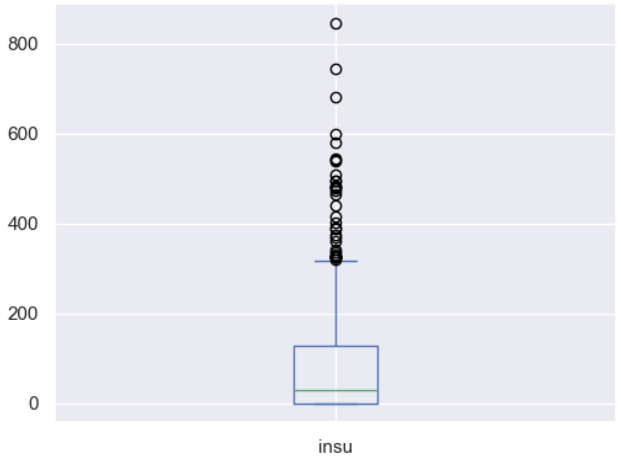
```
# Box and Whisker plot to visualize the distribution of all atributes
df.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False, figsize=(20,15))
plt.show()
```

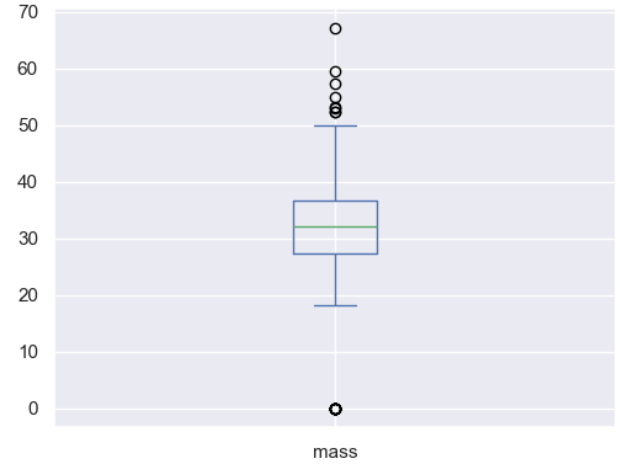

17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
preg

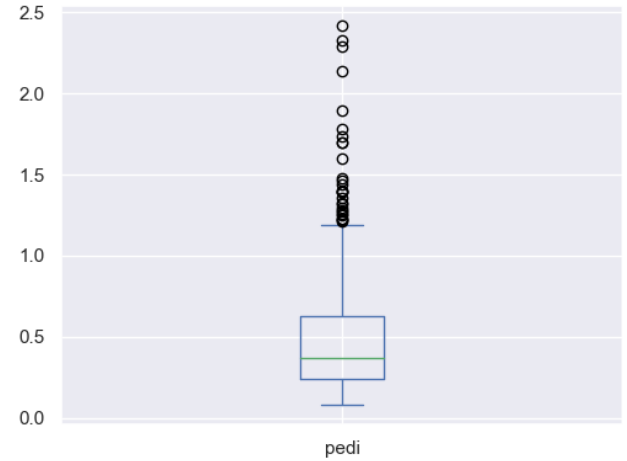

200
175
150
125
100
75
50
25
0
plas

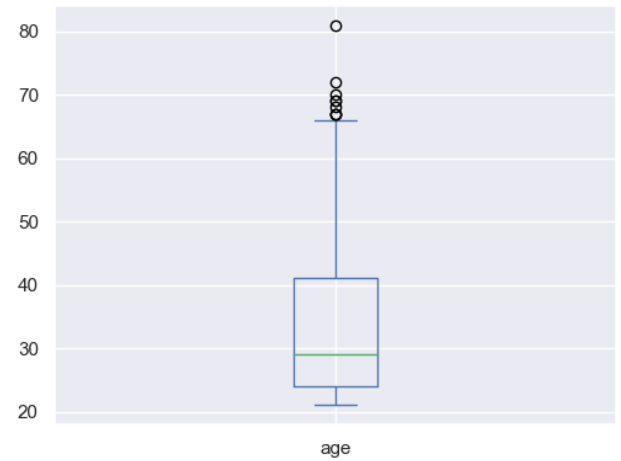

120
100
80
60
40
20
0
pres


100
80
60
40
20
0
skin


800
600
400
200
0
insu

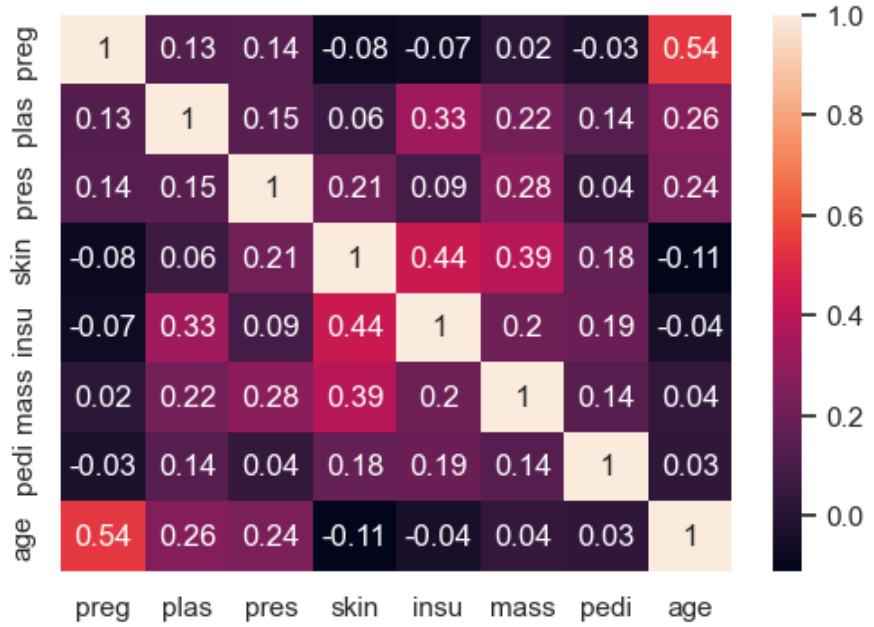

70
60
50
40
30
20
10
0
mass


2.5
2.0
1.5
1.0
0.5
0.0
pedi


80
70
60
50
40
30
20
age

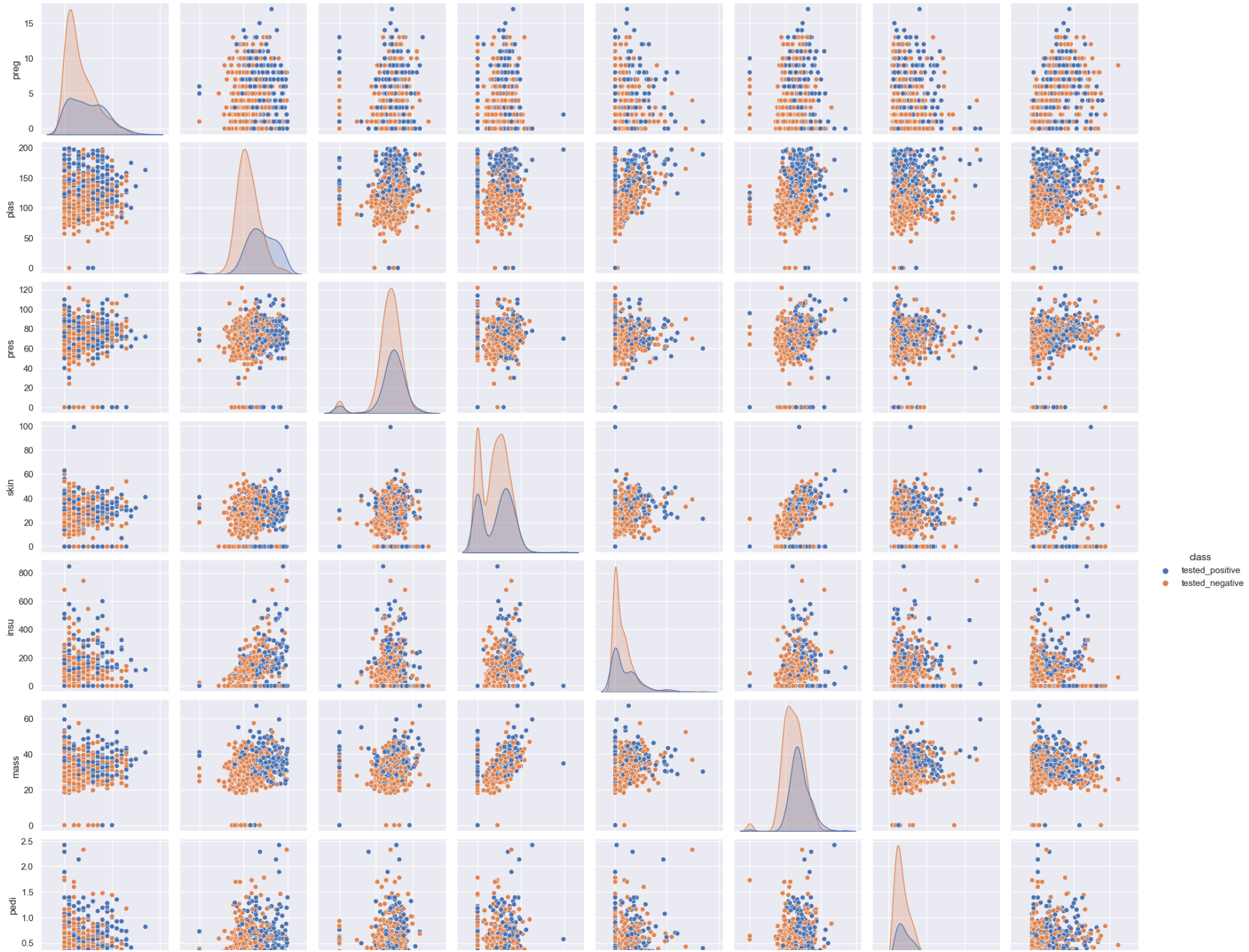
observation : outlier is detected in all features

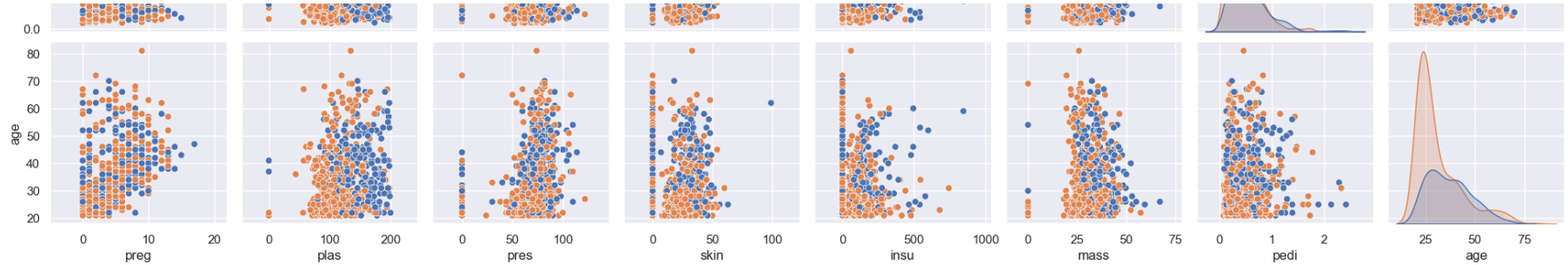
In [39]:
```
# Correlation between the different characteristics. Closer to 1 better is the correlation.
sns.heatmap(round(df.corr(method='pearson'),2), annot = True)
plt.show()
```



In [40]:

```
# Pairplot
sns.pairplot(df, hue='class')
plt.show()
```


preg
plas
pres
skin
insu
mass
pedi
class
tested_positive
tested_negative



In [41]:
```
df['classification'] = df['class'].apply(lambda x: 1 if x=='tested_negative' else 0)
# Remove two columns name is 'class'
df.drop(['class'], axis=1, inplace=True)
df.head(10)
```

Out[41]:

	preg	plas	pres	skin	insu	mass	pedi	age	classification
0	6	148	72	35	0	33.6	0.627	50	0
1	1	85	66	29	0	26.6	0.351	31	1
2	8	183	64	0	0	23.3	0.672	32	0
3	1	89	66	23	94	28.1	0.167	21	1
4	0	137	40	35	168	43.1	2.288	33	0
5	5	116	74	0	0	25.6	0.201	30	1
6	3	78	50	32	88	31.0	0.248	26	0
7	10	115	0	0	0	35.3	0.134	29	1
8	2	197	70	45	543	30.5	0.158	53	0
9	8	125	96	0	0	0.0	0.232	54	0

Remove Outlier using IQR

In [42]:

```python
def remove_outliers_iqr_all_columns(df):
    # Create an empty DataFrame to store filtered data
    filtered_df = pd.DataFrame()

    # Iterate through each column in the input DataFrame
    for col in df.columns:
        # Calculate the first quartile (Q1) and third quartile (Q3) for the column
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)

        # Calculate the IQR (Interquartile Range) for the column
        iqr = q3 - q1

        # Define the lower and upper bounds to identify outliers for the column
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr

        # Remove outliers from the column and add it to the filtered DataFrame
        filtered_df[col] = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)][col]

        # Drop nan value from the column and add it to the filtered DataFrame
        filtered_df = filtered_df.dropna(axis=0).reset_index(drop=True)

    return filtered_df
```

In [43]:

```python
df = remove_outliers_iqr_all_columns(df)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 629 entries, 0 to 628
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   preg            629 non-null    int64
 1   plas            629 non-null    float64
 2   pres            629 non-null    float64
 3   skin            629 non-null    float64
 4   insu            629 non-null    float64
 5   mass            629 non-null    float64
 6   pedi            629 non-null    float64
 7   age             629 non-null    float64
 8   classification  629 non-null    int64
dtypes: float64(7), int64(2)
memory usage: 44.4 KB
```

In [59]:
```
# # Box and Whisker plot to visualize the distribution of all atributes
# df.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False, figsize=(20,15))
# plt.show()
```

Classification

In [47]:
```
# Split data
# X, Y = df.iloc[:,:-1], df.iloc[:,-1]
X, Y = df.drop(['classification'], axis = 1),  df['classification']
# train_test_split 80/20
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, random_state = 42, stratify = Y)
```

In [53]:
```
# Model initialization
lr_Classifier = LogisticRegression()
knn_Classifier = KNeighborsClassifier()
gnb_Classifier = GaussianNB()
dt_Classifier = DecisionTreeClassifier()
rf_Classifier = RandomForestClassifier()
model_list = [lr_Classifier, knn_Classifier, gnb_Classifier, dt_Classifier, rf_Classifier]

# Scaler initialization
MinMax_scaler = MinMaxScaler()
Standard_scaler = StandardScaler()
MaxAbs_scaler = MaxAbsScaler()
Robust_scaler = RobustScaler()
Quantile_scaler = QuantileTransformer()
Power_scaler = PowerTransformer()
Normalizer_scaler = Normalizer()
scaler_list = [MinMax_scaler, Standard_scaler, MaxAbs_scaler, Robust_scaler,
               Quantile_scaler, Power_scaler, Normalizer_scaler]
```

In [56]:

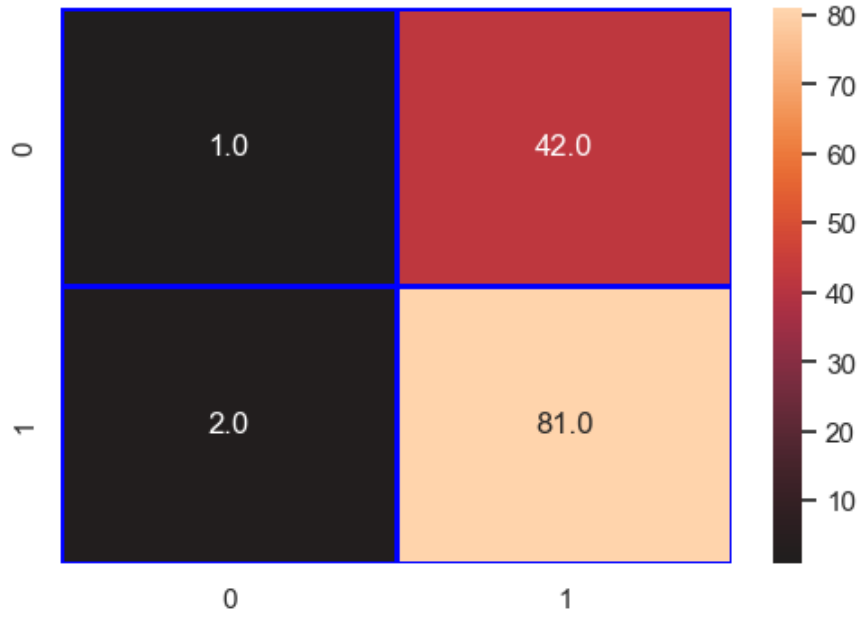
```python
def run_pipeline(X_train, X_test, y_train, y_test, scaler, classifier):
    # Model Information
    print(f"Modele name : {type(classifier).__name__}")
    print(f"Scaler name : {type(scaler).__name__}")

    # process 1 : fit and transform X_train data
    scaled_X_train = scaler.fit_transform(X_train)

    # process 2 : train model
    classifier.fit(scaled_X_train, y_train)

    # process 3 : transform X_test data
    scaled_X_test = scaler.transform(X_test)

    # process 4 : test model
    y_pred = classifier.predict(scaled_X_test)
    # print(y_pred, le.inverse_transform(y_pred))

    # process 5 : Perform k-fold cross-validation using cross_val_score
    scores = cross_val_score(classifier, scaled_X_train, y_train, cv=10, scoring='accuracy')
    print(f"10 K-Fold Accuracy_score : {np.round_(scores,4)}")
    print(f"10 K-Fold Average Accuracy_score : {round(np.average(scores)*100,2)} %")

    # process 6 : model evalution
    print("Accuracy_score:", round((accuracy_score(y_test, y_pred))*100,2),'%')
    print("Loss:", round((1-accuracy_score(y_test, y_pred))*100,2),'%')
    print("Cohen_kappa_score:", round((cohen_kappa_score(y_test, y_pred))*100,2),'%')
    print("Classification_report:\n",metrics.classification_report(y_test, y_pred))
    print("confusion_matrix:\n", confusion_matrix(y_test, y_pred))
    # plot confusion_matrix
    fig, ax = plt.subplots()
    fig.set_size_inches(6,4) # WH
    sns.heatmap(confusion_matrix(y_test, y_pred),
                annot=True,
                fmt=".1f",
                linewidths = 2,
                linecolor = "blue",
                center=0)
    plt.show()

    # process 7 : save model in pkl file
    filename = f'Moduls\\{str(type(classifier).__name__)}_{str(type(scaler).__name__)}_03_Disease_Prediction.pkl'
    pickle.dump(classifier, open(filename, 'wb'))

    # collect data for bar plot
    global plot_data_list
    plot_data_list.append([str(type(classifier).__name__),
                           str(type(scaler).__name__),
```

```
                        round((accuracy_score(y_test, y_pred))*100,2)])

    # end
    print("==="*30)
    print("\n\n")
    time.sleep(0.5)
```

In [57]:

```
for model in model_list:
    for scaler in scaler_list:
        run_pipeline(X_train, X_test, y_train, y_test, scaler, model)

    # plot data
    plot_df = pd.DataFrame(plot_data_list, columns=['classifier', 'scaler', 'accuracy_score'])
    plot_df.to_csv(f"Dataset\\{str(type(model).__name__)}_accuracy_score_plot_data_03_Disease_Prediction.csv", index=False)
    sns.set(rc={'figure.figsize':(18,6)})
    ax = sns.barplot(data=plot_df, x="classifier", y="accuracy_score", hue="scaler")
    plt.title('Accuracy Score Plot')
    plt.xlabel('Classifier')
    plt.ylabel('Accuracy Score')
    for i in ax.containers:
        ax.bar_label(i,)
    plt.show()

    # empty list
    plot_data_list = []
    print("\n\n")

print("Done...")
```

```
Modele name : LogisticRegression
Scaler name : MinMaxScaler
10 K-Fold Accuracy_score : [0.6471 0.6275 0.6471 0.68   0.66   0.64   0.66   0.64   0.66   0.62  ]
10 K-Fold Average Accuracy_score : 64.82 %
Accuracy_score: 65.08 %
Loss: 34.92 %
Cohen_kappa_score: -0.11 %
Classification_report:
              precision    recall  f1-score   support

           0       0.33      0.02      0.04        43
           1       0.66      0.98      0.79        83

    accuracy                           0.65       126
   macro avg       0.50      0.50      0.41       126
weighted avg       0.55      0.65      0.53       126

confusion_matrix:
 [[ 1 42]
 [ 2 81]]
```

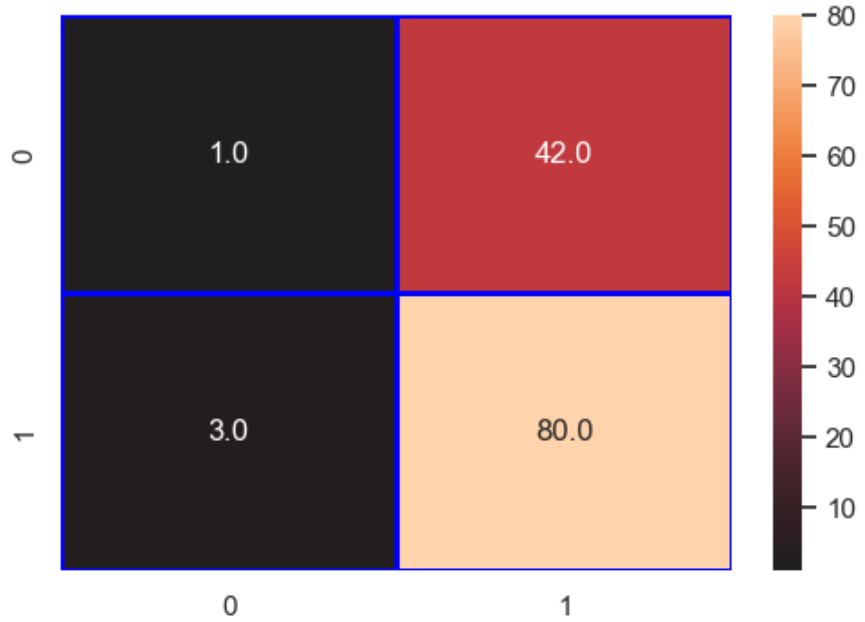


==========

Modele name : LogisticRegression
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.6275 0.6275 0.5882 0.64 0.66 0.62 0.66 0.64 0.64 0.62]
10 K-Fold Average Accuracy_score : 63.23 %
Accuracy_score: 64.29 %
Loss: 35.71 %
Cohen_kappa_score: -1.65 %
Classification_report:

	precision	recall	f1-score	support
0	0.25	0.02	0.04	43
1	0.66	0.96	0.78	83
accuracy			0.64	126
macro avg	0.45	0.49	0.41	126
weighted avg	0.52	0.64	0.53	126

confusion_matrix:
[[1 42]
[3 80]]

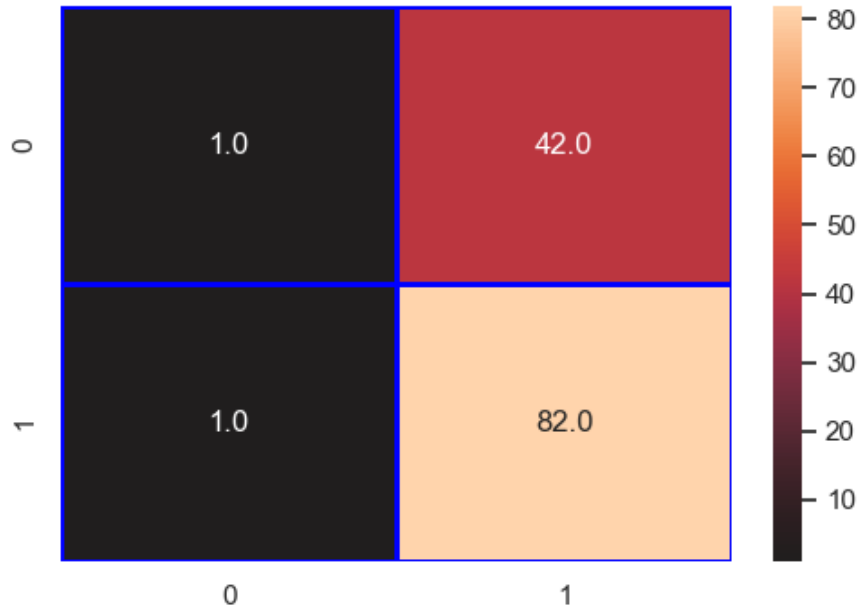


==========

Modele name : LogisticRegression
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.6471 0.6275 0.6471 0.68 0.66 0.64 0.68 0.64 0.66 0.64]
10 K-Fold Average Accuracy_score : 65.22 %
Accuracy_score: 65.87 %
Loss: 34.13 %
Cohen_kappa_score: 1.46 %
Classification_report:

	precision	recall	f1-score	support
0	0.50	0.02	0.04	43
1	0.66	0.99	0.79	83
accuracy			0.66	126
macro avg	0.58	0.51	0.42	126
weighted avg	0.61	0.66	0.54	126

confusion_matrix:
[[1 42]
[1 82]]

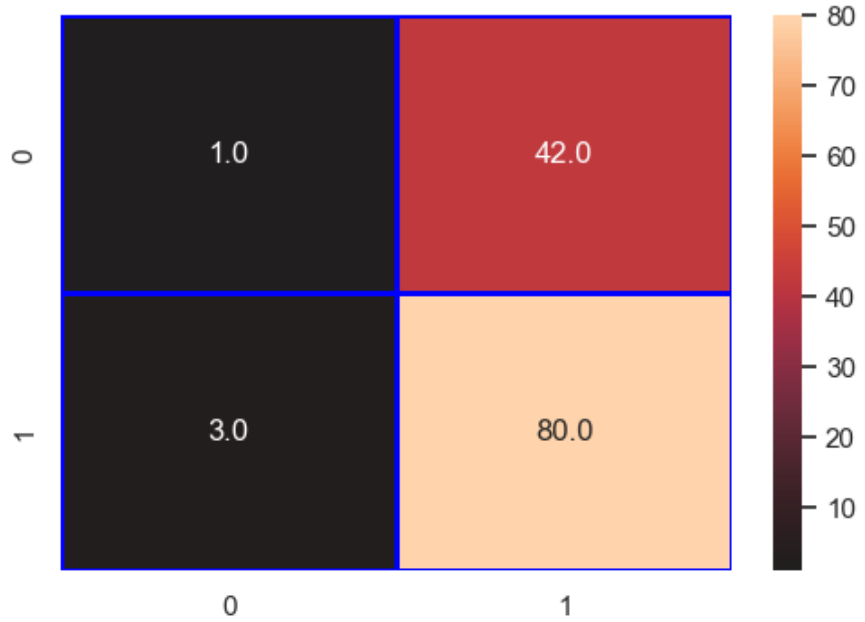


==========

Modele name : LogisticRegression
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.6275 0.6275 0.5882 0.64 0.66 0.62 0.66 0.64 0.66 0.62]
10 K-Fold Average Accuracy_score : 63.43 %
Accuracy_score: 64.29 %
Loss: 35.71 %
Cohen_kappa_score: -1.65 %
Classification_report:

	precision	recall	f1-score	support
0	0.25	0.02	0.04	43
1	0.66	0.96	0.78	83
accuracy			0.64	126
macro avg	0.45	0.49	0.41	126
weighted avg	0.52	0.64	0.53	126

confusion_matrix:
[[1 42]
[3 80]]

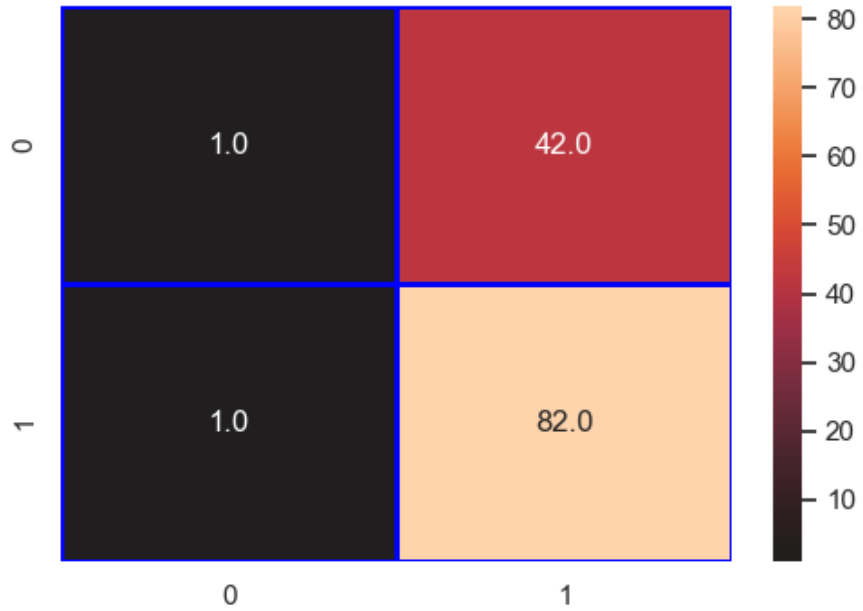


==========

Modele name : LogisticRegression
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.7255 0.6471 0.6078 0.7 0.68 0.64 0.66 0.66 0.66 0.62]
10 K-Fold Average Accuracy_score : 66.0 %
Accuracy_score: 65.87 %
Loss: 34.13 %
Cohen_kappa_score: 1.46 %
Classification_report:

	precision	recall	f1-score	support
0	0.50	0.02	0.04	43
1	0.66	0.99	0.79	83
accuracy			0.66	126
macro avg	0.58	0.51	0.42	126
weighted avg	0.61	0.66	0.54	126

confusion_matrix:
[[1 42]
[1 82]]

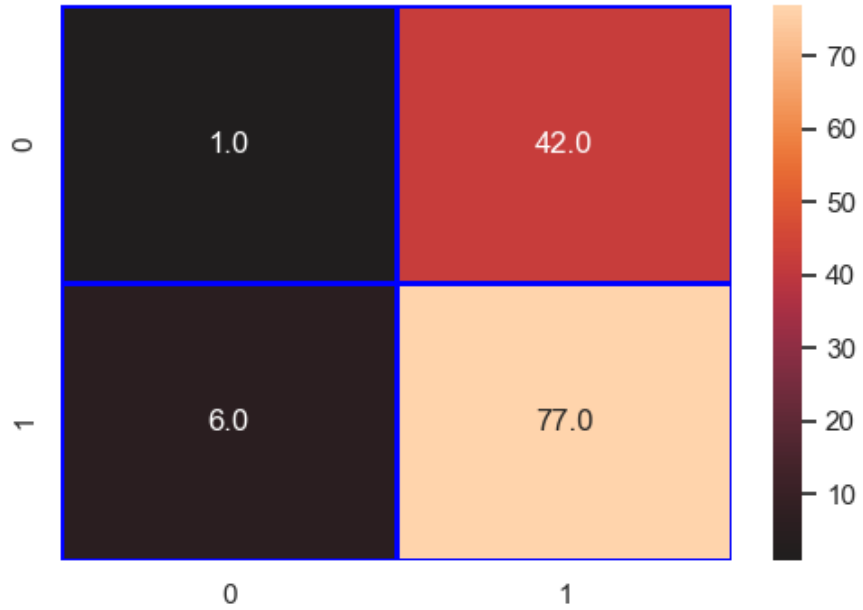


==========

Modele name : LogisticRegression
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.6667 0.6078 0.5882 0.72 0.68 0.62 0.66 0.6 0.68 0.62]
10 K-Fold Average Accuracy_score : 64.43 %
Accuracy_score: 61.9 %
Loss: 38.1 %
Cohen_kappa_score: -6.14 %
Classification_report:

	precision	recall	f1-score	support
0	0.14	0.02	0.04	43
1	0.65	0.93	0.76	83
accuracy			0.62	126
macro avg	0.39	0.48	0.40	126
weighted avg	0.47	0.62	0.52	126

confusion_matrix:
[[1 42]
[6 77]]

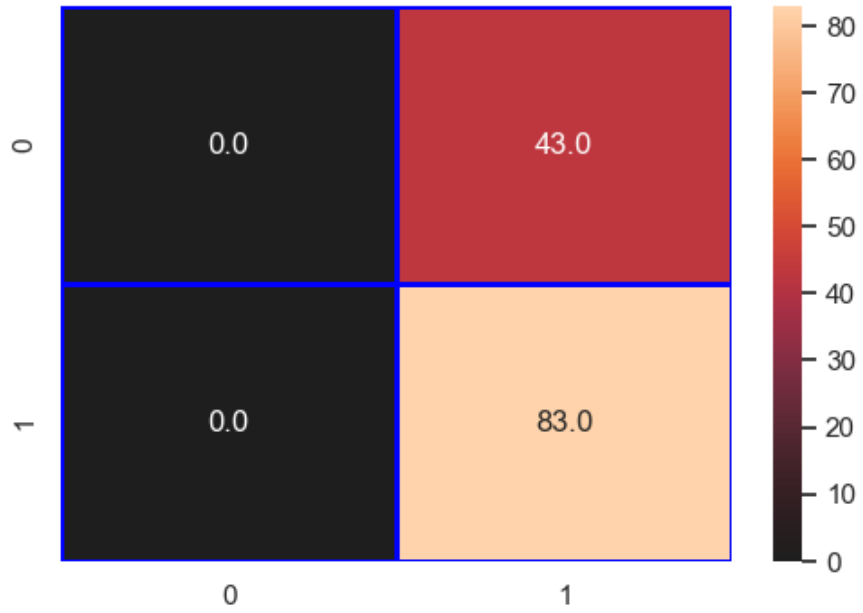


==========

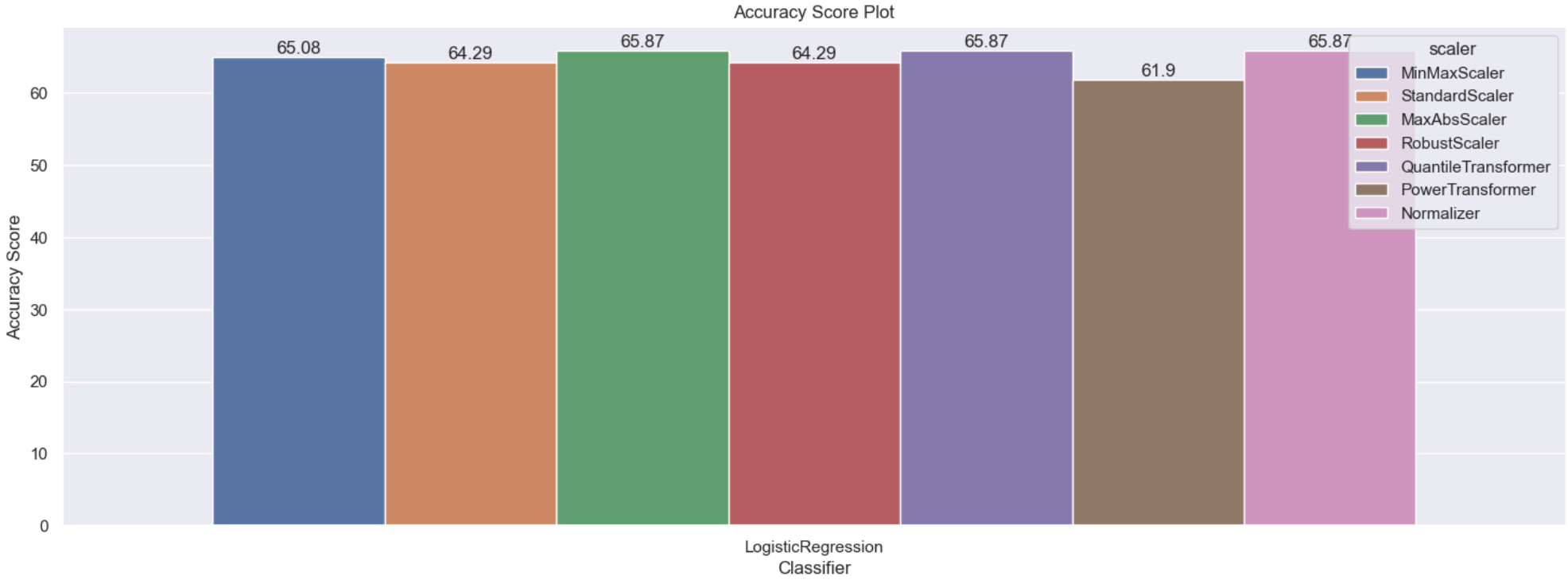
Modele name : LogisticRegression
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.6471 0.6471 0.6471 0.66   0.66   0.66   0.66   0.66   0.66   0.66]
10 K-Fold Average Accuracy_score : 65.61 %
Accuracy_score: 65.87 %
Loss: 34.13 %
Cohen_kappa_score: 0.0 %
Classification_report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	43
1	0.66	1.00	0.79	83
accuracy			0.66	126
macro avg	0.33	0.50	0.40	126
weighted avg	0.43	0.66	0.52	126

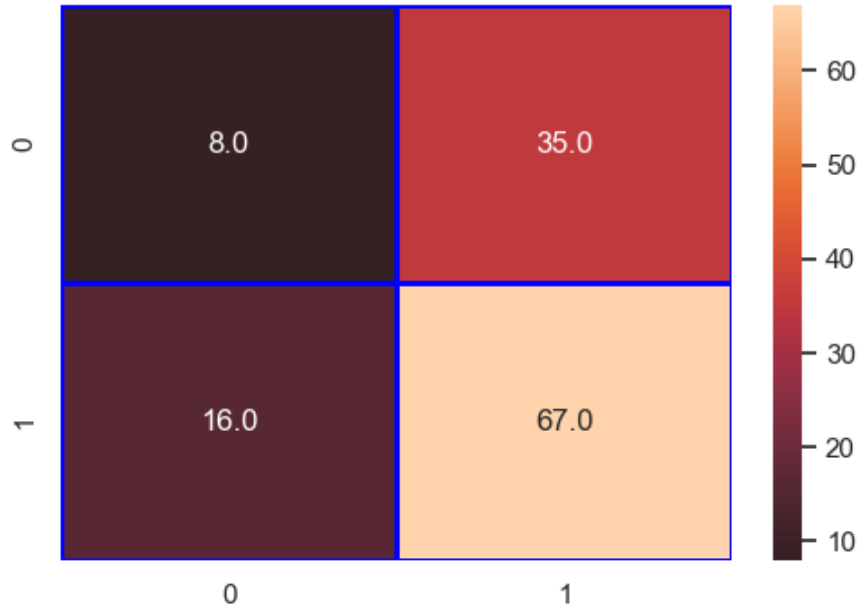
confusion_matrix:
 [[0 43]
 [0 83]]


0.0
43.0
0.0
83.0
0
1
0
1
80
70
60
50
40
30
20
10
0

==========




```
Modele name : KNeighborsClassifier
Scaler name : MinMaxScaler
10 K-Fold Accuracy_score : [0.549  0.549  0.4902 0.48   0.66   0.54   0.64   0.74   0.64   0.54  ]
10 K-Fold Average Accuracy_score : 58.28 %
Accuracy_score: 59.52 %
Loss: 40.48 %
Cohen_kappa_score: -0.75 %
Classification_report:
              precision    recall  f1-score   support

           0       0.33      0.19      0.24        43
           1       0.66      0.81      0.72        83

    accuracy                           0.60       126
   macro avg       0.50      0.50      0.48       126
weighted avg       0.55      0.60      0.56       126

confusion_matrix:
 [[ 8 35]
 [16 67]]
```

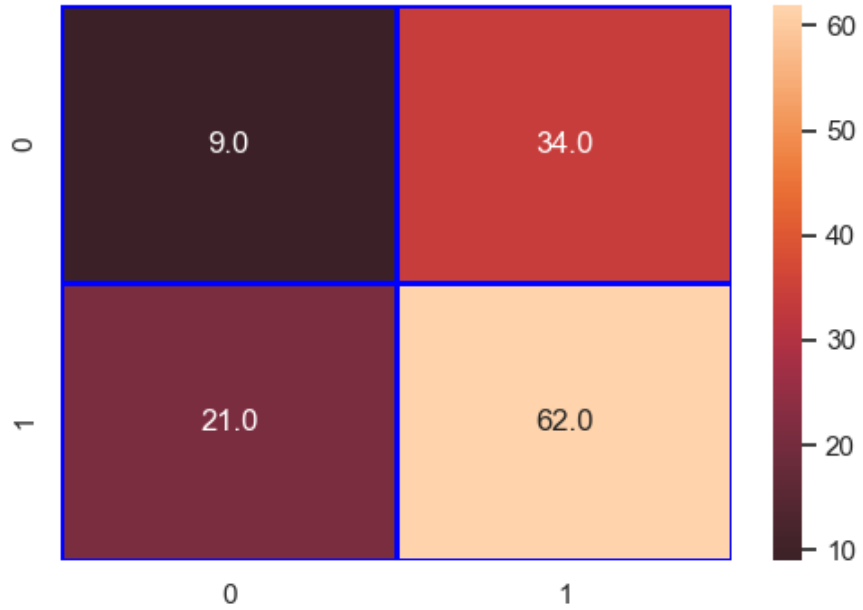


==========

Modele name : KNeighborsClassifier
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.5686 0.549 0.4902 0.46   0.64   0.54   0.62   0.68   0.62   0.46]
10 K-Fold Average Accuracy_score : 56.28 %
Accuracy_score: 56.35 %
Loss: 43.65 %
Cohen_kappa_score: -4.71 %
Classification_report:

	precision	recall	f1-score	support
0	0.30	0.21	0.25	43
1	0.65	0.75	0.69	83
accuracy			0.56	126
macro avg	0.47	0.48	0.47	126
weighted avg	0.53	0.56	0.54	126

confusion_matrix:
 [[9 34]
 [21 62]]

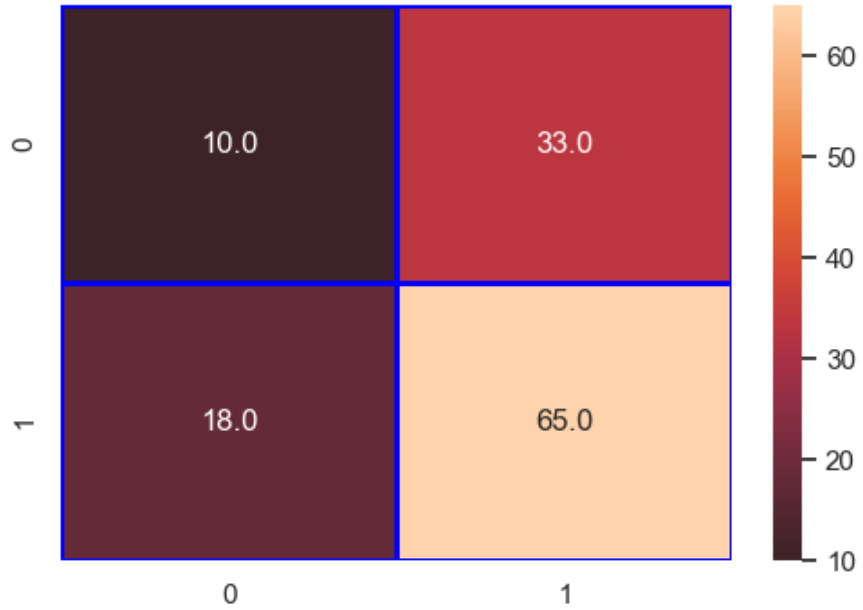


==========

Modele name : KNeighborsClassifier
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.6078 0.5686 0.5098 0.56 0.62 0.52 0.7 0.66 0.72 0.52]
10 K-Fold Average Accuracy_score : 59.86 %
Accuracy_score: 59.52 %
Loss: 40.48 %
Cohen_kappa_score: 1.71 %
Classification_report:

	precision	recall	f1-score	support
0	0.36	0.23	0.28	43
1	0.66	0.78	0.72	83
accuracy			0.60	126
macro avg	0.51	0.51	0.50	126
weighted avg	0.56	0.60	0.57	126

confusion_matrix:
[[10 33]
[18 65]]

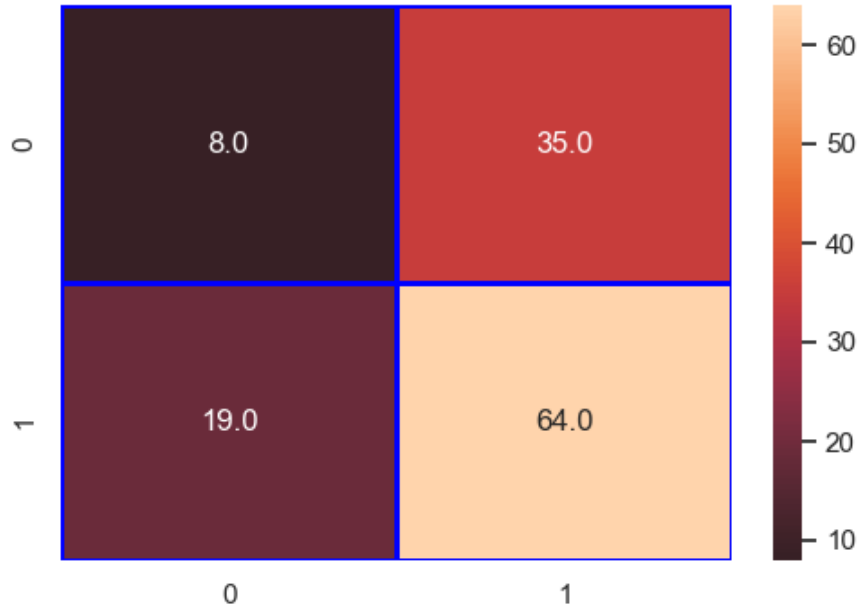


==========

Modele name : KNeighborsClassifier
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.5294 0.5294 0.451 0.48   0.68   0.56   0.6   0.62   0.68   0.48]
10 K-Fold Average Accuracy_score : 56.1 %
Accuracy_score: 57.14 %
Loss: 42.86 %
Cohen_kappa_score: -4.71 %
Classification_report:

	precision	recall	f1-score	support
0	0.30	0.19	0.23	43
1	0.65	0.77	0.70	83
accuracy			0.57	126
macro avg	0.47	0.48	0.47	126
weighted avg	0.53	0.57	0.54	126

confusion_matrix:
 [[8 35]
 [19 64]]

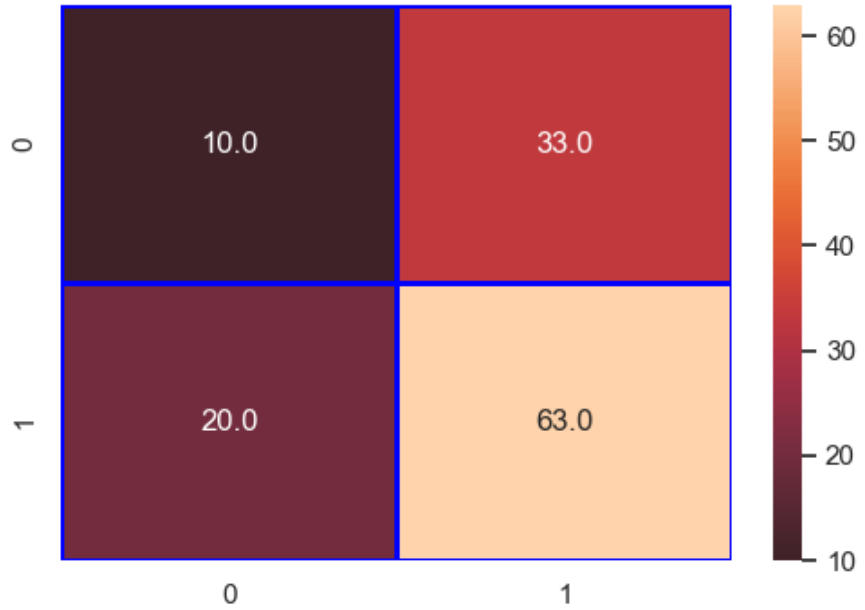


==========

Modele name : KNeighborsClassifier
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.6078 0.5686 0.4902 0.5 0.7 0.64 0.66 0.68 0.6 0.56]
10 K-Fold Average Accuracy_score : 60.07 %
Accuracy_score: 57.94 %
Loss: 42.06 %
Cohen_kappa_score: -0.91 %
Classification_report:

	precision	recall	f1-score	support
0	0.33	0.23	0.27	43
1	0.66	0.76	0.70	83
accuracy			0.58	126
macro avg	0.49	0.50	0.49	126
weighted avg	0.55	0.58	0.56	126

confusion_matrix:
[[10 33]
[20 63]]

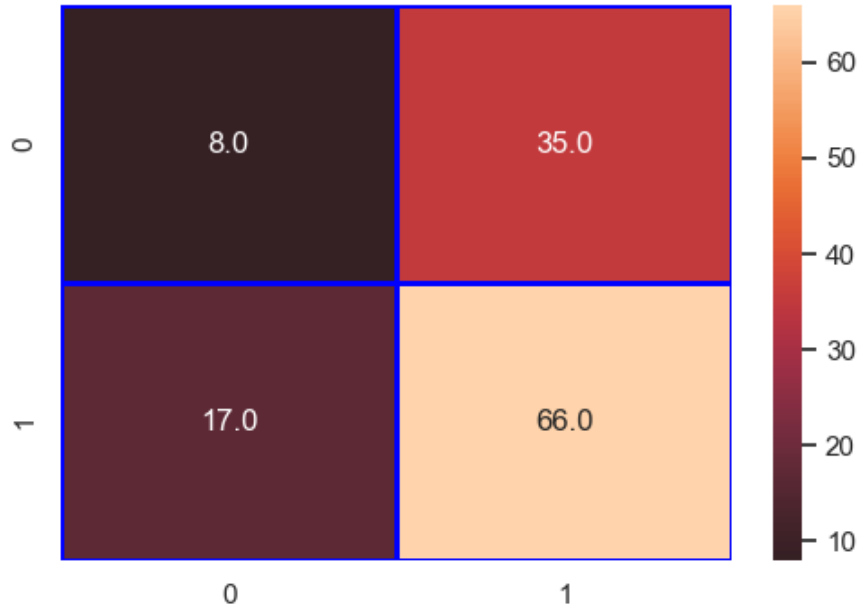


==========

Modele name : KNeighborsClassifier
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.5294 0.549 0.6078 0.42   0.72   0.62   0.7   0.62   0.56   0.52]
10 K-Fold Average Accuracy_score : 58.46 %
Accuracy_score: 58.73 %
Loss: 41.27 %
Cohen_kappa_score: -2.09 %
Classification_report:

	precision	recall	f1-score	support
0	0.32	0.19	0.24	43
1	0.65	0.80	0.72	83
accuracy			0.59	126
macro avg	0.49	0.49	0.48	126
weighted avg	0.54	0.59	0.55	126

confusion_matrix:
 [[8 35]
 [17 66]]

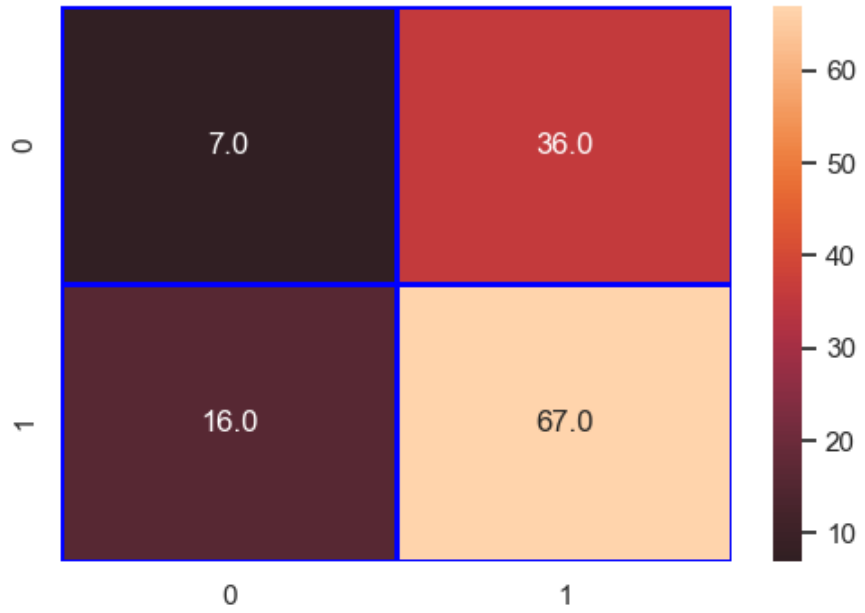


==========

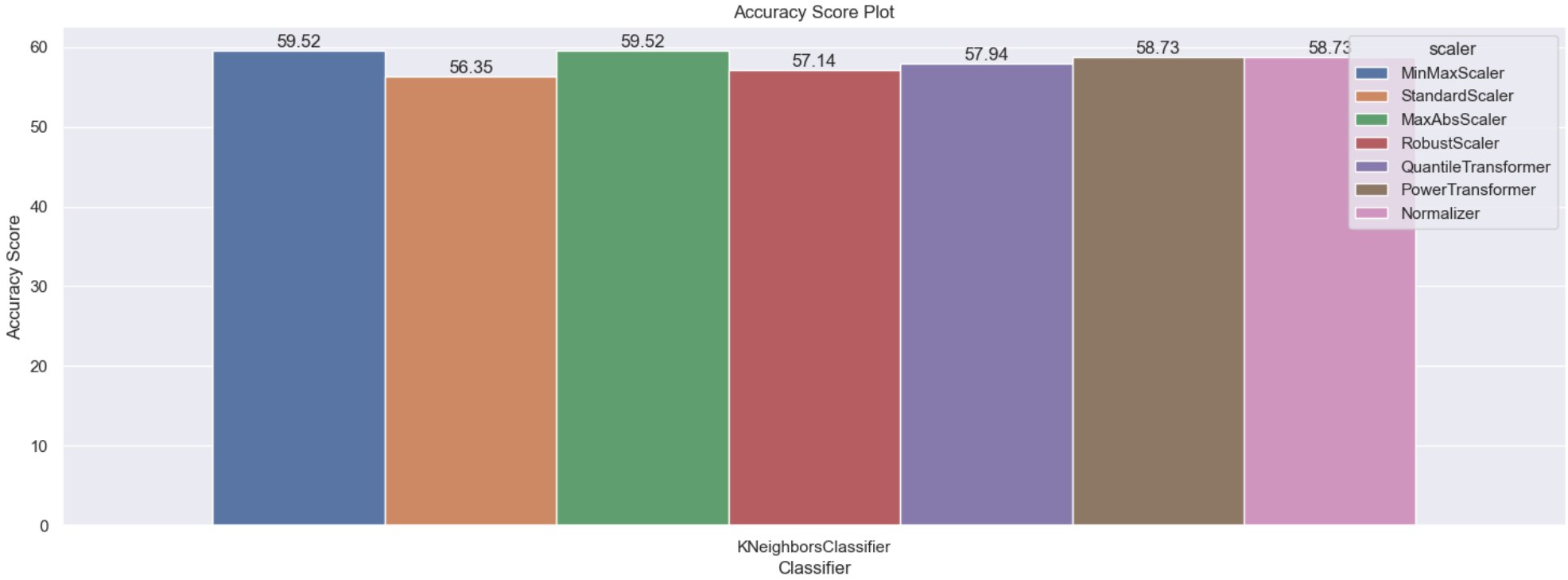
Modele name : KNeighborsClassifier
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.5882 0.549 0.549 0.58   0.58   0.54   0.6   0.66   0.54   0.56]
10 K-Fold Average Accuracy_score : 57.46 %
Accuracy_score: 58.73 %
Loss: 41.27 %
Cohen_kappa_score: -3.38 %
Classification_report:

	precision	recall	f1-score	support
0	0.30	0.16	0.21	43
1	0.65	0.81	0.72	83
accuracy			0.59	126
macro avg	0.48	0.49	0.47	126
weighted avg	0.53	0.59	0.55	126

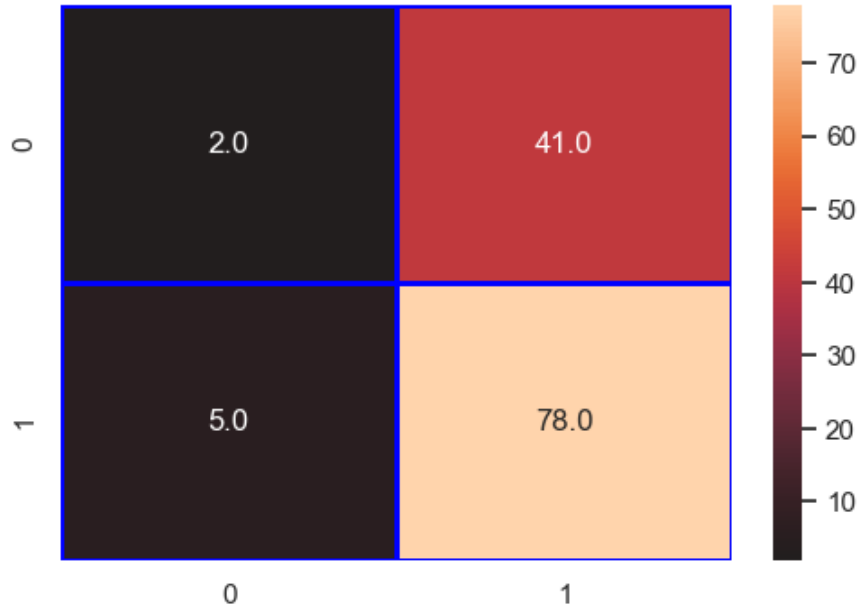
confusion_matrix:
 [[7 36]
 [16 67]]


7.0
36.0
16.0
67.0
0
1
0
1
60
50
40
30
20
10

==========




```
Modele name : GaussianNB
Scaler name : MinMaxScaler
10 K-Fold Accuracy_score : [0.6078 0.6275 0.6078 0.64   0.68   0.64   0.6    0.64   0.68   0.58  ]
10 K-Fold Average Accuracy_score : 63.03 %
Accuracy_score: 63.49 %
Loss: 36.51 %
Cohen_kappa_score: -1.72 %
Classification_report:
              precision    recall  f1-score   support

           0       0.29      0.05      0.08        43
           1       0.66      0.94      0.77        83

    accuracy                           0.63       126
   macro avg       0.47      0.49      0.43       126
weighted avg       0.53      0.63      0.54       126

confusion_matrix:
 [[ 2 41]
 [ 5 78]]
```

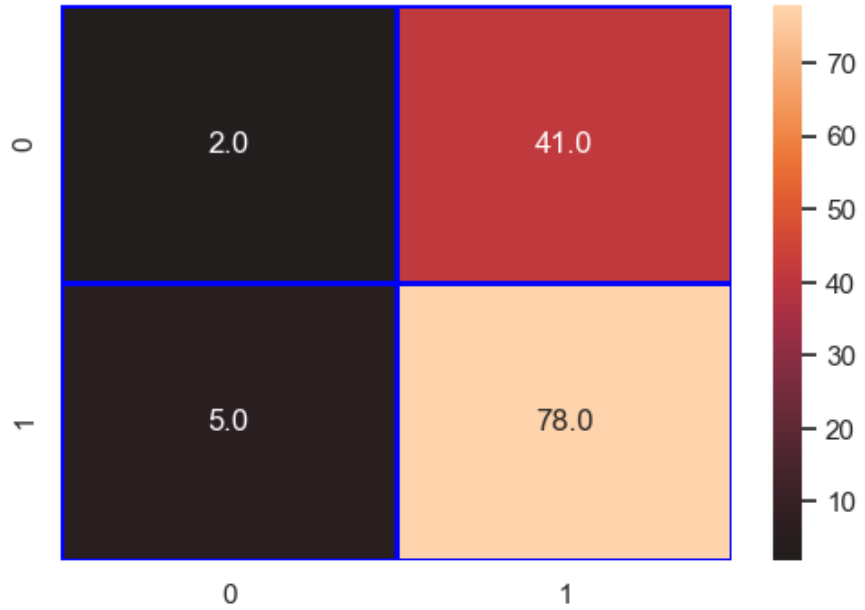


==========

Modele name : GaussianNB
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.6078 0.6275 0.6078 0.64 0.68 0.64 0.6 0.64 0.68 0.58]
10 K-Fold Average Accuracy_score : 63.03 %
Accuracy_score: 63.49 %
Loss: 36.51 %
Cohen_kappa_score: -1.72 %
Classification_report:

	precision	recall	f1-score	support
0	0.29	0.05	0.08	43
1	0.66	0.94	0.77	83
accuracy			0.63	126
macro avg	0.47	0.49	0.43	126
weighted avg	0.53	0.63	0.54	126

confusion_matrix:
[[2 41]
[5 78]]

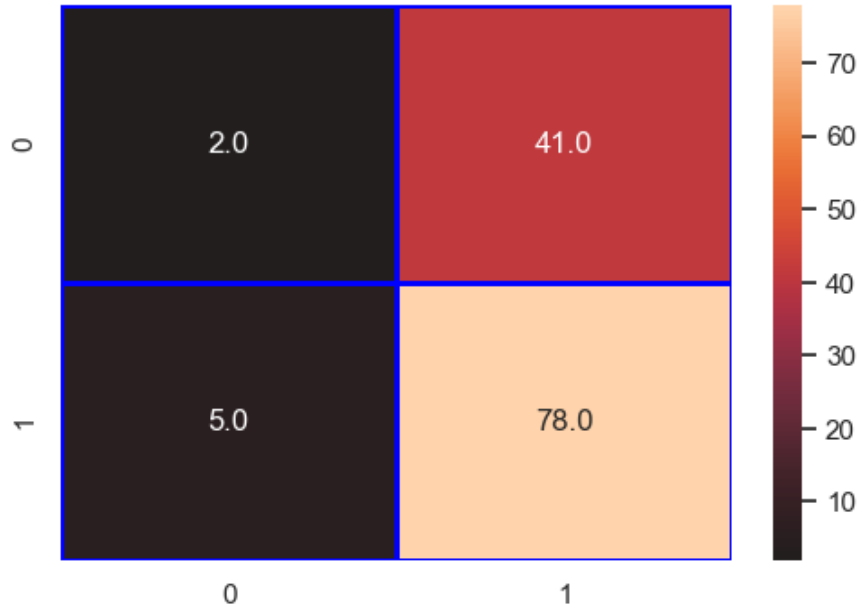


==========

Modele name : GaussianNB
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.6078 0.6275 0.6078 0.64 0.68 0.64 0.6 0.64 0.68 0.58]
10 K-Fold Average Accuracy_score : 63.03 %
Accuracy_score: 63.49 %
Loss: 36.51 %
Cohen_kappa_score: -1.72 %
Classification_report:

	precision	recall	f1-score	support
0	0.29	0.05	0.08	43
1	0.66	0.94	0.77	83
accuracy			0.63	126
macro avg	0.47	0.49	0.43	126
weighted avg	0.53	0.63	0.54	126

confusion_matrix:
[[2 41]
[5 78]]

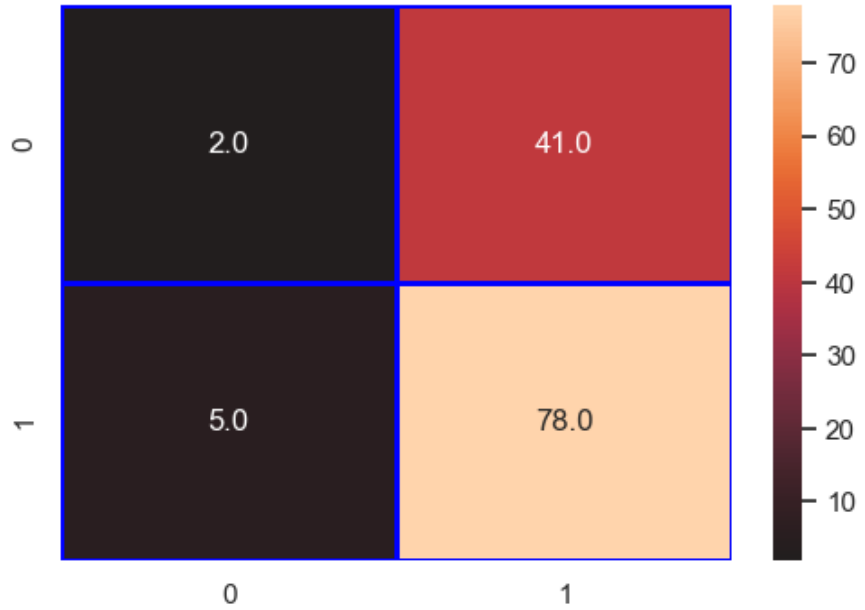


==========

Modele name : GaussianNB
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.6078 0.6275 0.6078 0.64 0.68 0.64 0.6 0.64 0.68 0.58]
10 K-Fold Average Accuracy_score : 63.03 %
Accuracy_score: 63.49 %
Loss: 36.51 %
Cohen_kappa_score: -1.72 %
Classification_report:

	precision	recall	f1-score	support
0	0.29	0.05	0.08	43
1	0.66	0.94	0.77	83
accuracy			0.63	126
macro avg	0.47	0.49	0.43	126
weighted avg	0.53	0.63	0.54	126

confusion_matrix:
[[2 41]
[5 78]]

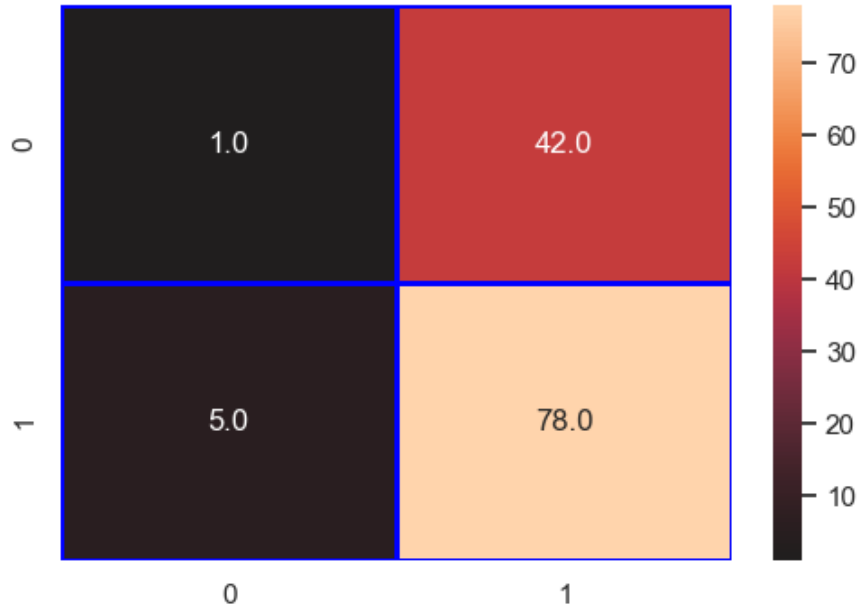


==========

Modele name : GaussianNB
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.6863 0.6275 0.5882 0.66   0.7   0.6   0.66   0.62   0.7   0.58]
10 K-Fold Average Accuracy_score : 64.22 %
Accuracy_score: 62.7 %
Loss: 37.3 %
Cohen_kappa_score: -4.67 %
Classification_report:

	precision	recall	f1-score	support
0	0.17	0.02	0.04	43
1	0.65	0.94	0.77	83
accuracy			0.63	126
macro avg	0.41	0.48	0.40	126
weighted avg	0.49	0.63	0.52	126

confusion_matrix:
 [[1 42]
 [5 78]]

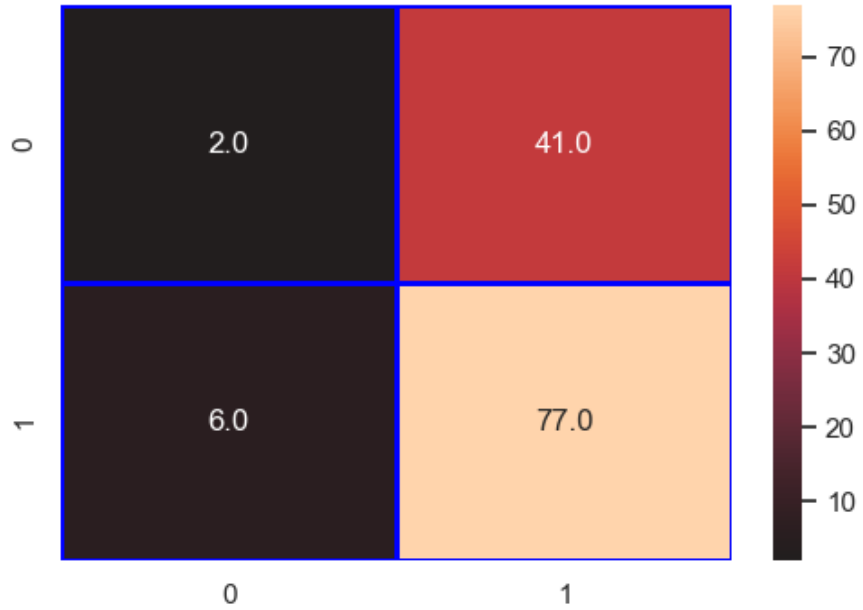


==========

Modele name : GaussianNB
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.6667 0.6667 0.6078 0.64 0.7 0.6 0.66 0.66 0.74 0.58]
10 K-Fold Average Accuracy_score : 65.21 %
Accuracy_score: 62.7 %
Loss: 37.3 %
Cohen_kappa_score: -3.21 %
Classification_report:

	precision	recall	f1-score	support
0	0.25	0.05	0.08	43
1	0.65	0.93	0.77	83
accuracy			0.63	126
macro avg	0.45	0.49	0.42	126
weighted avg	0.52	0.63	0.53	126

confusion_matrix:
[[2 41]
[6 77]]

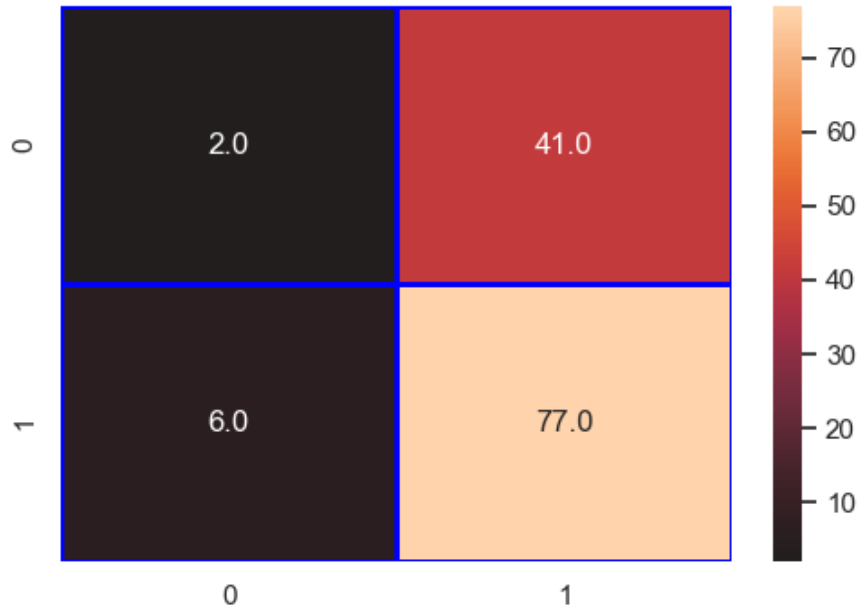


==========

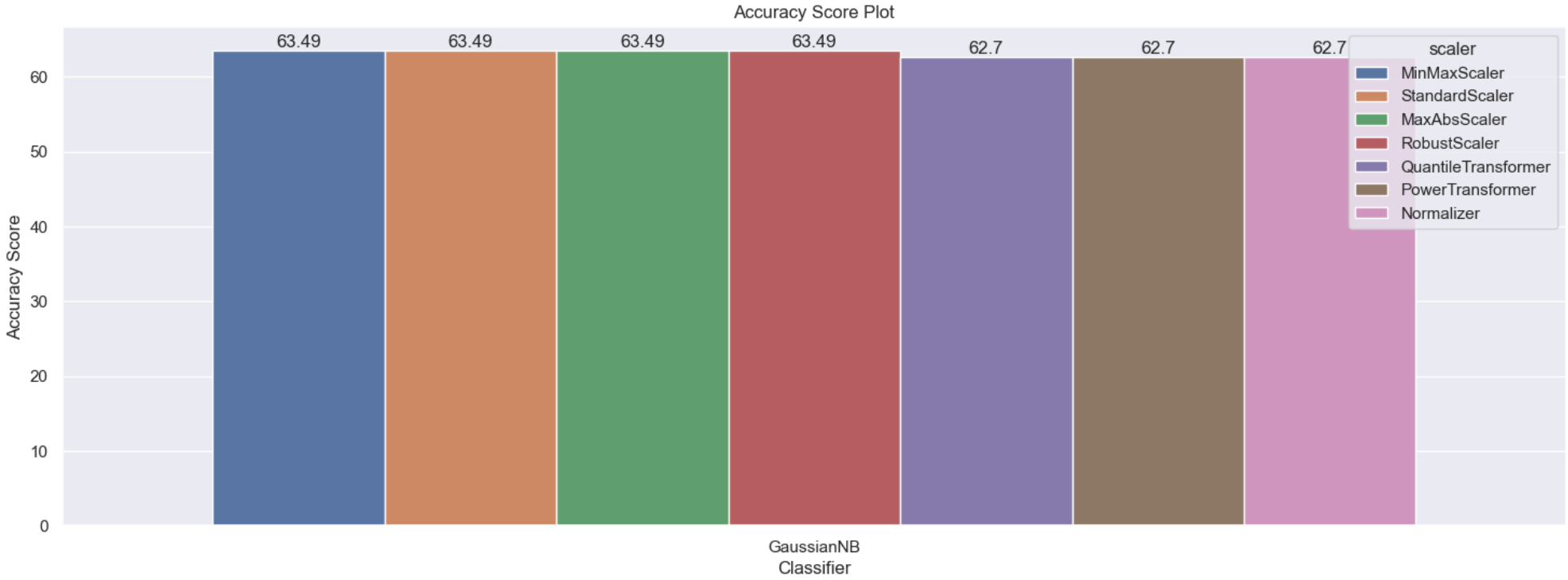
Modele name : GaussianNB
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.6078 0.6471 0.6275 0.62 0.68 0.52 0.66 0.66 0.72 0.54]
10 K-Fold Average Accuracy_score : 62.82 %
Accuracy_score: 62.7 %
Loss: 37.3 %
Cohen_kappa_score: -3.21 %
Classification_report:

	precision	recall	f1-score	support
0	0.25	0.05	0.08	43
1	0.65	0.93	0.77	83
accuracy			0.63	126
macro avg	0.45	0.49	0.42	126
weighted avg	0.52	0.63	0.53	126

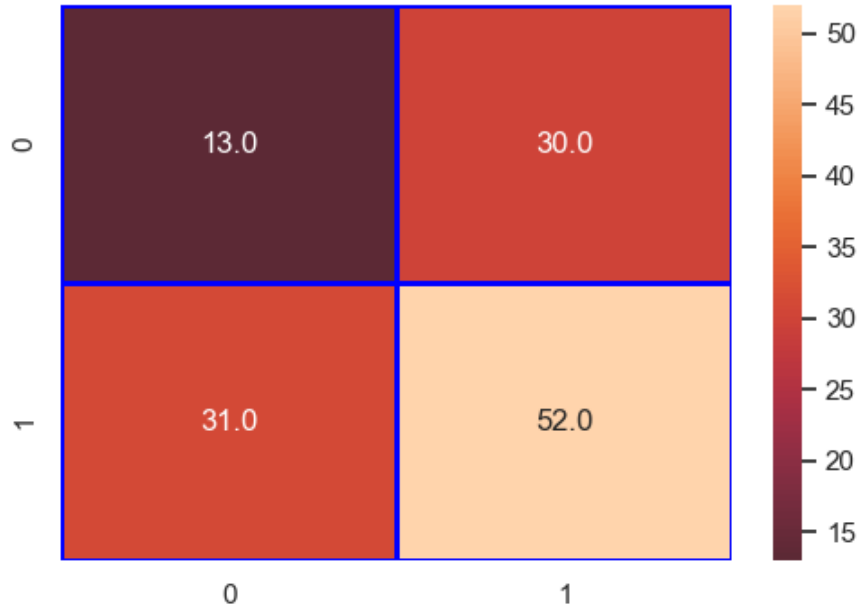
confusion_matrix:
[[2 41]
[6 77]]


2.0
41.0
6.0
77.0
0
1
0
1
70
60
50
40
30
20
10

==========




```
Modele name : DecisionTreeClassifier
Scaler name : MinMaxScaler
10 K-Fold Accuracy_score : [0.5294 0.5098 0.5294 0.58   0.56   0.42   0.52   0.58   0.5    0.5   ]
10 K-Fold Average Accuracy_score : 52.29 %
Accuracy_score: 51.59 %
Loss: 48.41 %
Cohen_kappa_score: -7.08 %
Classification_report:
              precision    recall  f1-score   support

           0       0.30      0.30      0.30        43
           1       0.63      0.63      0.63        83

    accuracy                           0.52       126
   macro avg       0.46      0.46      0.46       126
weighted avg       0.52      0.52      0.52       126

confusion_matrix:
 [[13 30]
 [31 52]]
```

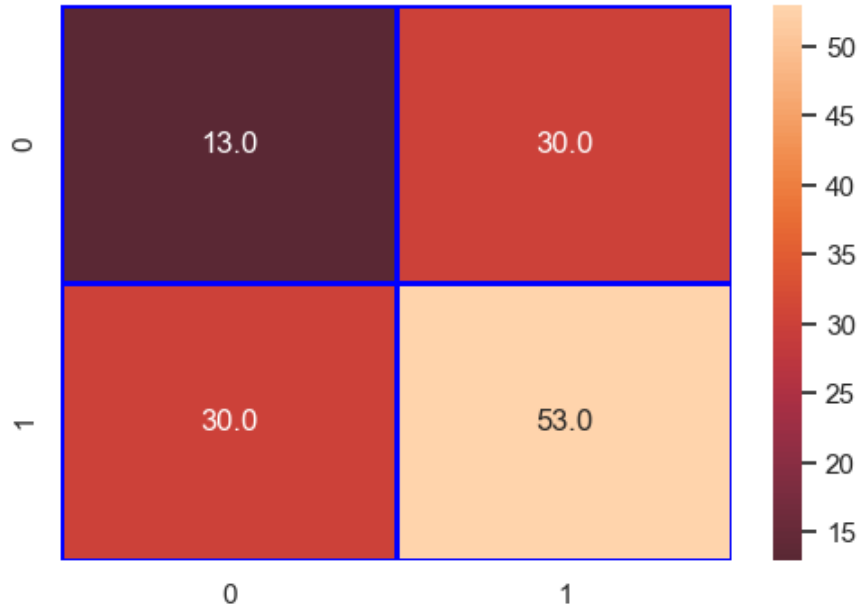


==========

Modele name : DecisionTreeClassifier
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.5882 0.5686 0.5098 0.56 0.52 0.48 0.6 0.54 0.54 0.48]
10 K-Fold Average Accuracy_score : 53.87 %
Accuracy_score: 52.38 %
Loss: 47.62 %
Cohen_kappa_score: -5.91 %
Classification_report:

	precision	recall	f1-score	support
0	0.30	0.30	0.30	43
1	0.64	0.64	0.64	83
accuracy			0.52	126
macro avg	0.47	0.47	0.47	126
weighted avg	0.52	0.52	0.52	126

confusion_matrix:
[[13 30]
[30 53]]

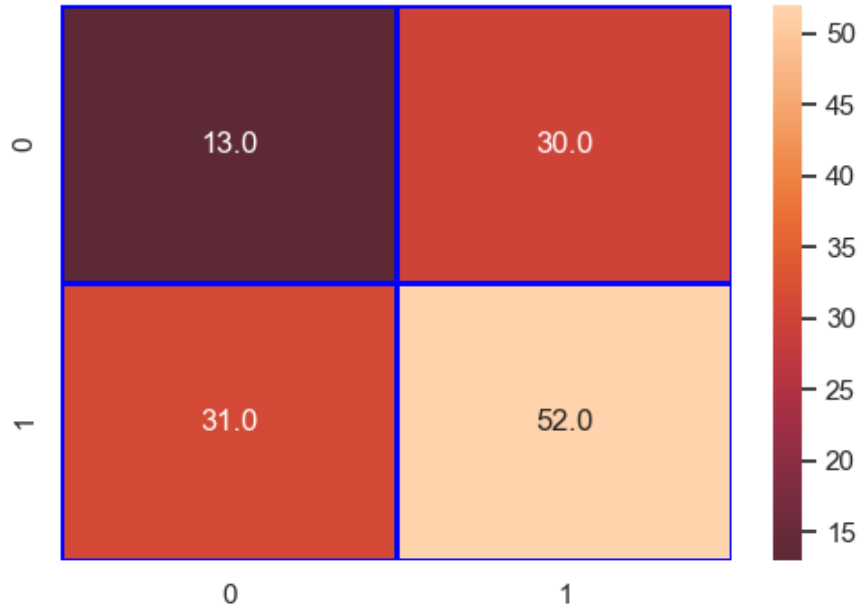


==========

Modele name : DecisionTreeClassifier
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.5686 0.5686 0.5686 0.52 0.54 0.5 0.58 0.6 0.54 0.5]
10 K-Fold Average Accuracy_score : 54.86 %
Accuracy_score: 51.59 %
Loss: 48.41 %
Cohen_kappa_score: -7.08 %
Classification_report:

	precision	recall	f1-score	support
0	0.30	0.30	0.30	43
1	0.63	0.63	0.63	83
accuracy			0.52	126
macro avg	0.46	0.46	0.46	126
weighted avg	0.52	0.52	0.52	126

confusion_matrix:
[[13 30]
[31 52]]

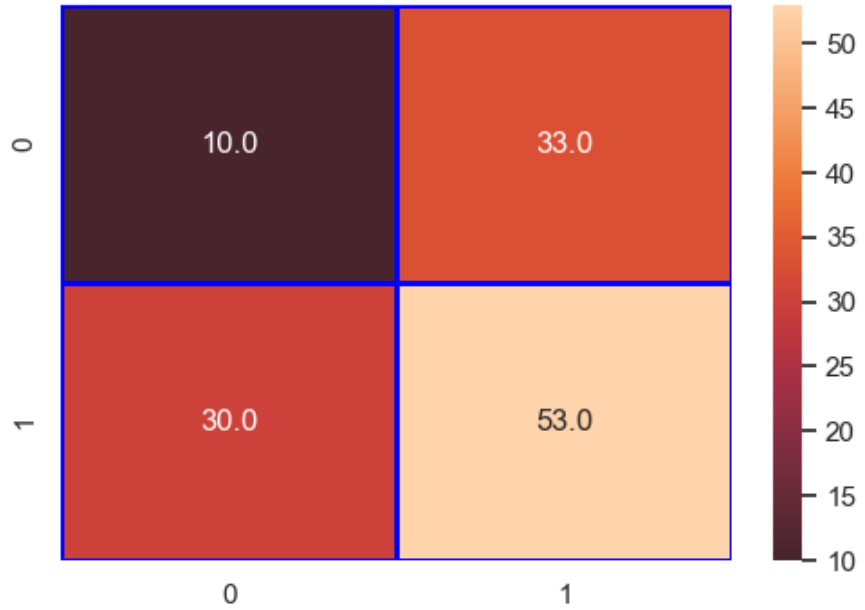


==========

Modele name : DecisionTreeClassifier
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.549 0.549 0.5098 0.58   0.54   0.44   0.5   0.6   0.54   0.46]
10 K-Fold Average Accuracy_score : 52.68 %
Accuracy_score: 50.0 %
Loss: 50.0 %
Cohen_kappa_score: -13.11 %
Classification_report:

	precision	recall	f1-score	support
0	0.25	0.23	0.24	43
1	0.62	0.64	0.63	83
accuracy			0.50	126
macro avg	0.43	0.44	0.43	126
weighted avg	0.49	0.50	0.50	126

confusion_matrix:
 [[10 33]
 [30 53]]

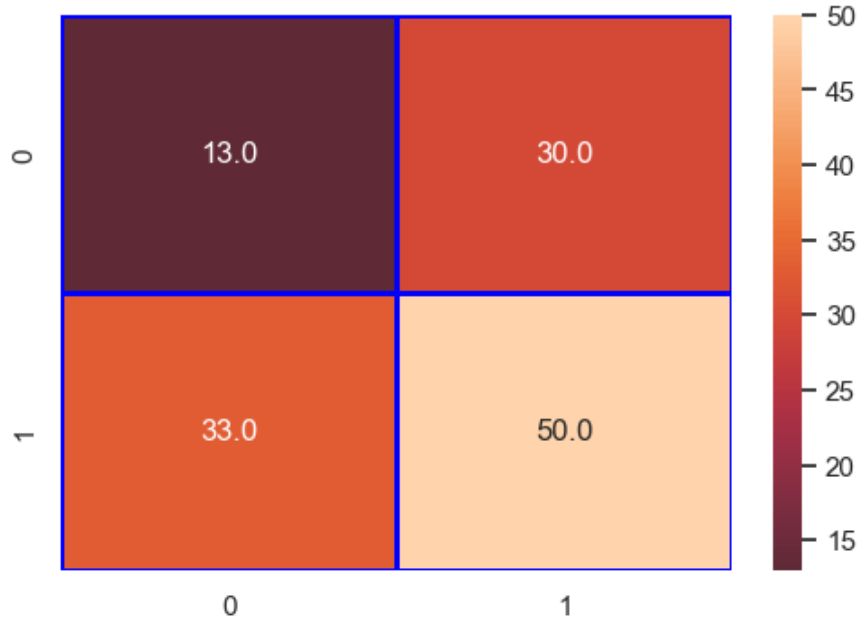


==========

Modele name : DecisionTreeClassifier
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.5294 0.549 0.5294 0.54   0.56   0.5   0.56   0.58   0.56   0.44]
10 K-Fold Average Accuracy_score : 53.48 %
Accuracy_score: 50.0 %
Loss: 50.0 %
Cohen_kappa_score: -9.37 %
Classification_report:

	precision	recall	f1-score	support
0	0.28	0.30	0.29	43
1	0.62	0.60	0.61	83
accuracy			0.50	126
macro avg	0.45	0.45	0.45	126
weighted avg	0.51	0.50	0.50	126

confusion_matrix:
 [[13 30]
 [33 50]]

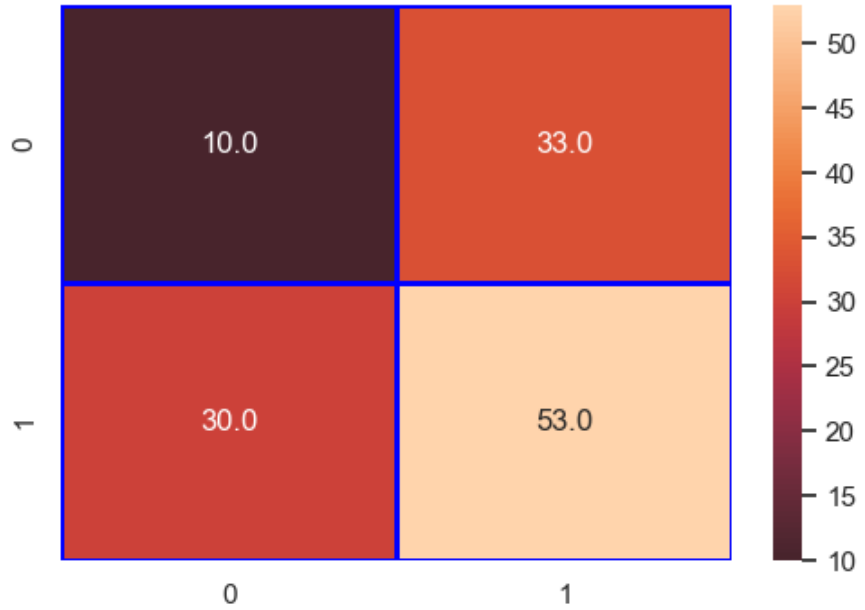


==========

Modele name : DecisionTreeClassifier
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.5882 0.549 0.5294 0.52   0.52   0.48   0.52   0.6   0.54   0.46]
10 K-Fold Average Accuracy_score : 53.07 %
Accuracy_score: 50.0 %
Loss: 50.0 %
Cohen_kappa_score: -13.11 %
Classification_report:

	precision	recall	f1-score	support
0	0.25	0.23	0.24	43
1	0.62	0.64	0.63	83
accuracy			0.50	126
macro avg	0.43	0.44	0.43	126
weighted avg	0.49	0.50	0.50	126

confusion_matrix:
 [[10 33]
 [30 53]]

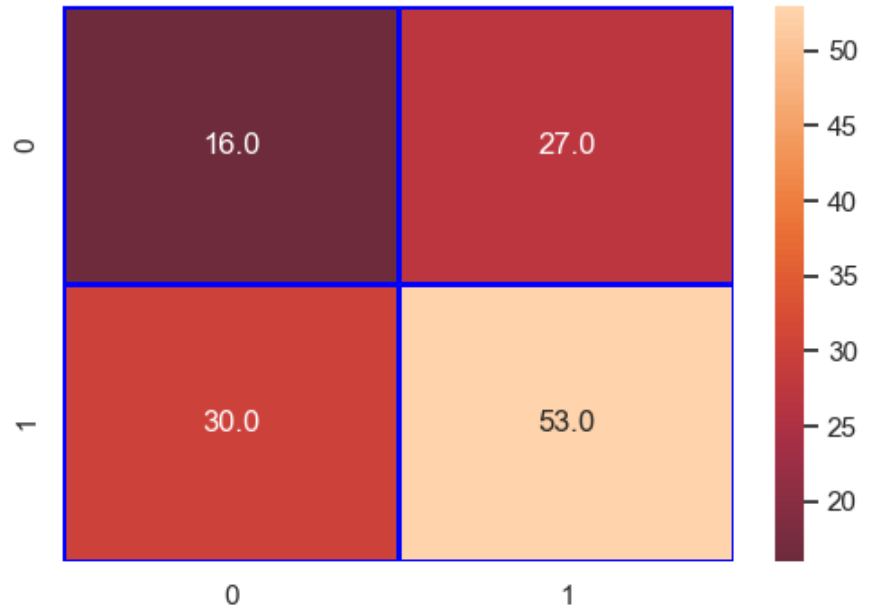


==========

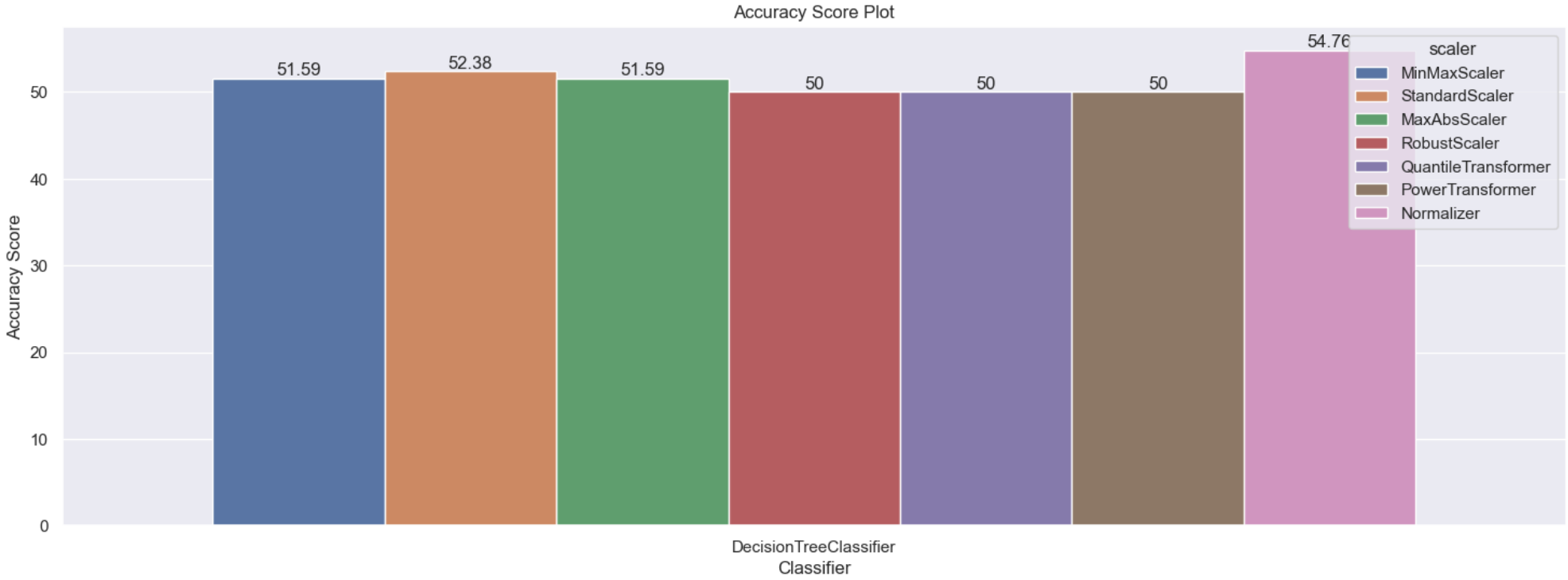
Modele name : DecisionTreeClassifier
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.5294 0.4902 0.5882 0.52 0.58 0.4 0.5 0.52 0.5 0.52]
10 K-Fold Average Accuracy_score : 51.48 %
Accuracy_score: 54.76 %
Loss: 45.24 %
Cohen_kappa_score: 1.05 %
Classification_report:

	precision	recall	f1-score	support
0	0.35	0.37	0.36	43
1	0.66	0.64	0.65	83
accuracy			0.55	126
macro avg	0.51	0.51	0.50	126
weighted avg	0.56	0.55	0.55	126

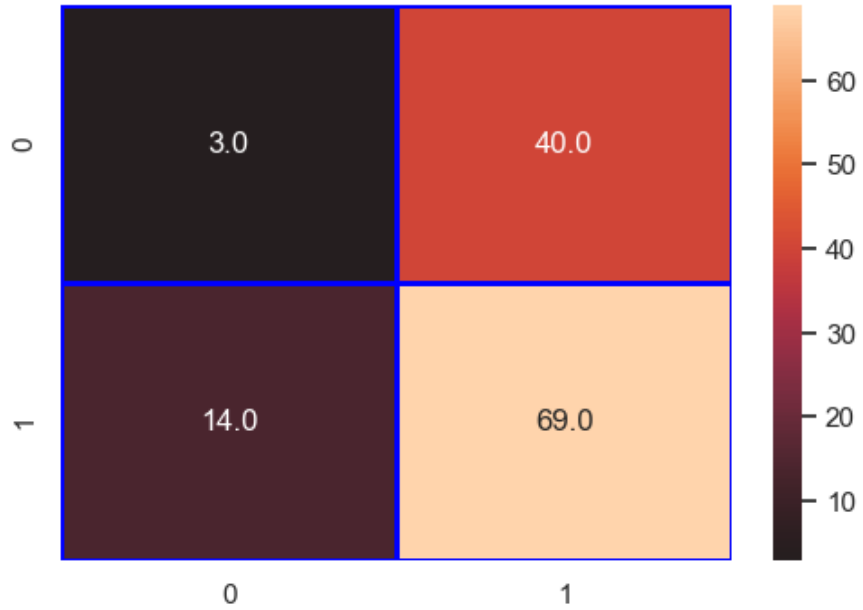
confusion_matrix:
[[16 27]
[30 53]]


16.0
27.0
30.0
53.0
0
1
0
1
50
45
40
35
30
25
20

==========




```
Modele name : RandomForestClassifier
Scaler name : MinMaxScaler
10 K-Fold Accuracy_score : [0.4902 0.6078 0.5882 0.58   0.64   0.6    0.6    0.68   0.64   0.64  ]
10 K-Fold Average Accuracy_score : 60.66 %
Accuracy_score: 57.14 %
Loss: 42.86 %
Cohen_kappa_score: -11.58 %
Classification_report:
              precision    recall  f1-score   support

           0       0.18      0.07      0.10        43
           1       0.63      0.83      0.72        83

    accuracy                           0.57       126
   macro avg       0.40      0.45      0.41       126
weighted avg       0.48      0.57      0.51       126

confusion_matrix:
 [[ 3 40]
 [14 69]]
```

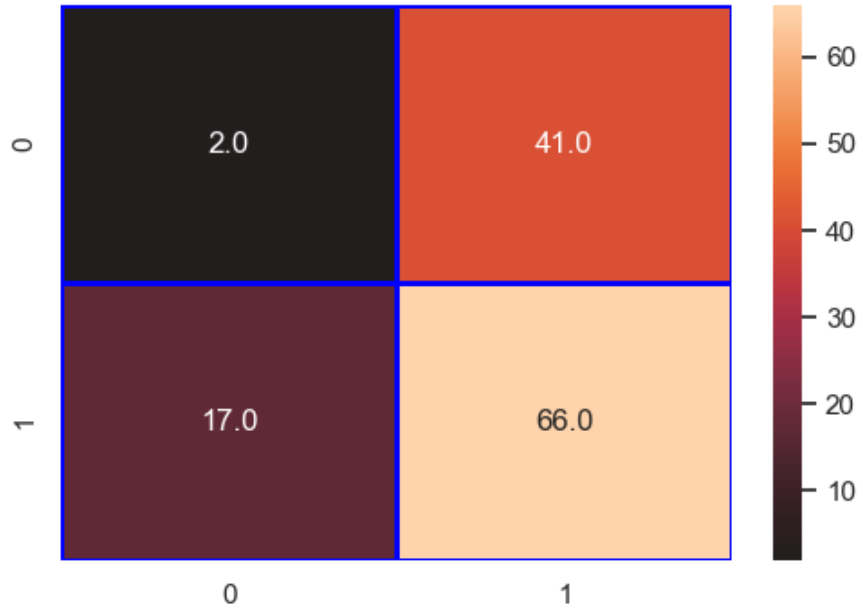


==========

Modele name : RandomForestClassifier
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.5882 0.6275 0.5882 0.62   0.7   0.58   0.64   0.76   0.6   0.54]
10 K-Fold Average Accuracy_score : 62.44 %
Accuracy_score: 53.97 %
Loss: 46.03 %
Cohen_kappa_score: -18.29 %
Classification_report:

	precision	recall	f1-score	support
0	0.11	0.05	0.06	43
1	0.62	0.80	0.69	83
accuracy			0.54	126
macro avg	0.36	0.42	0.38	126
weighted avg	0.44	0.54	0.48	126

confusion_matrix:
 [[2 41]
 [17 66]]

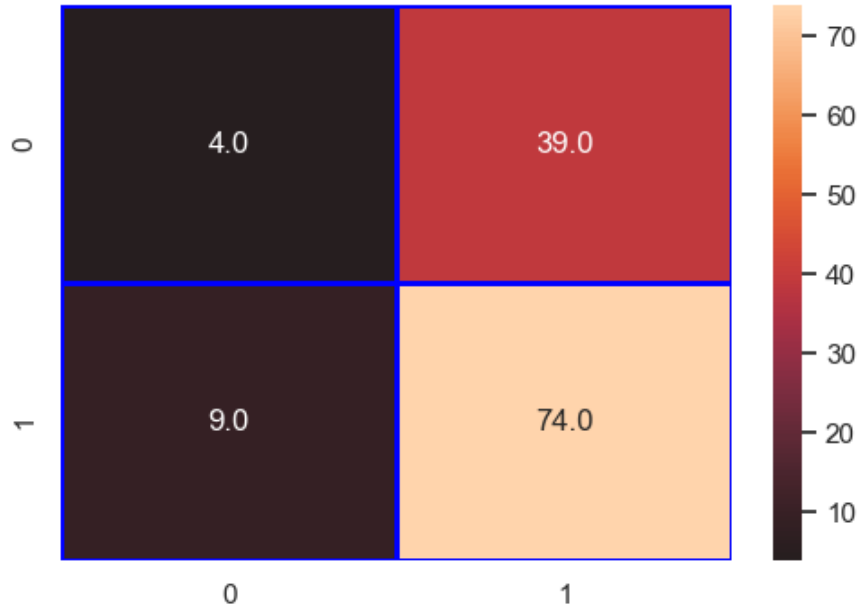


==========

Modele name : RandomForestClassifier
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.549 0.6078 0.5882 0.58   0.68   0.58   0.58   0.66   0.6   0.62]
10 K-Fold Average Accuracy_score : 60.45 %
Accuracy_score: 61.9 %
Loss: 38.1 %
Cohen_kappa_score: -1.85 %
Classification_report:

	precision	recall	f1-score	support
0	0.31	0.09	0.14	43
1	0.65	0.89	0.76	83
accuracy			0.62	126
macro avg	0.48	0.49	0.45	126
weighted avg	0.54	0.62	0.55	126

confusion_matrix:
 [[4 39]
 [9 74]]

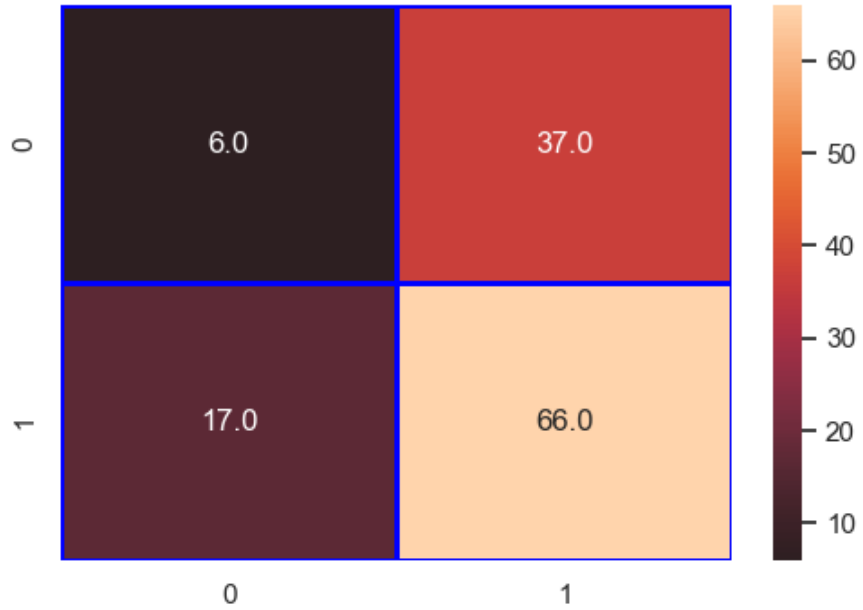


==========

Modele name : RandomForestClassifier
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.5882 0.5882 0.5294 0.62   0.68   0.62   0.56   0.68   0.64   0.6   ]
10 K-Fold Average Accuracy_score : 61.06 %
Accuracy_score: 57.14 %
Loss: 42.86 %
Cohen_kappa_score: -7.35 %
Classification_report:

	precision	recall	f1-score	support
0	0.26	0.14	0.18	43
1	0.64	0.80	0.71	83
accuracy			0.57	126
macro avg	0.45	0.47	0.45	126
weighted avg	0.51	0.57	0.53	126

confusion_matrix:
 [[6 37]
 [17 66]]

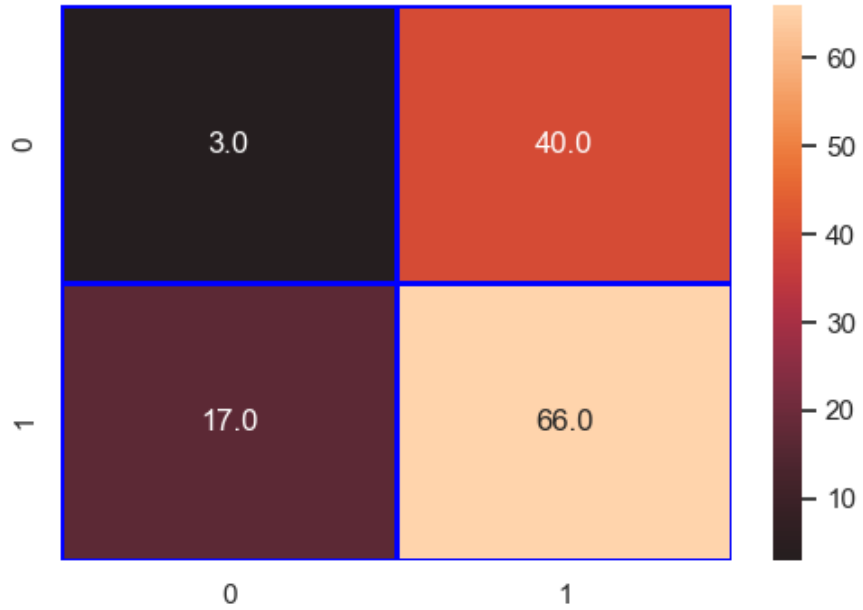


==========

Modele name : RandomForestClassifier
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.6078 0.5882 0.5294 0.58 0.68 0.58 0.6 0.7 0.62 0.54]
10 K-Fold Average Accuracy_score : 60.25 %
Accuracy_score: 54.76 %
Loss: 45.24 %
Cohen_kappa_score: -15.5 %
Classification_report:

	precision	recall	f1-score	support
0	0.15	0.07	0.10	43
1	0.62	0.80	0.70	83
accuracy			0.55	126
macro avg	0.39	0.43	0.40	126
weighted avg	0.46	0.55	0.49	126

confusion_matrix:
[[3 40]
[17 66]]

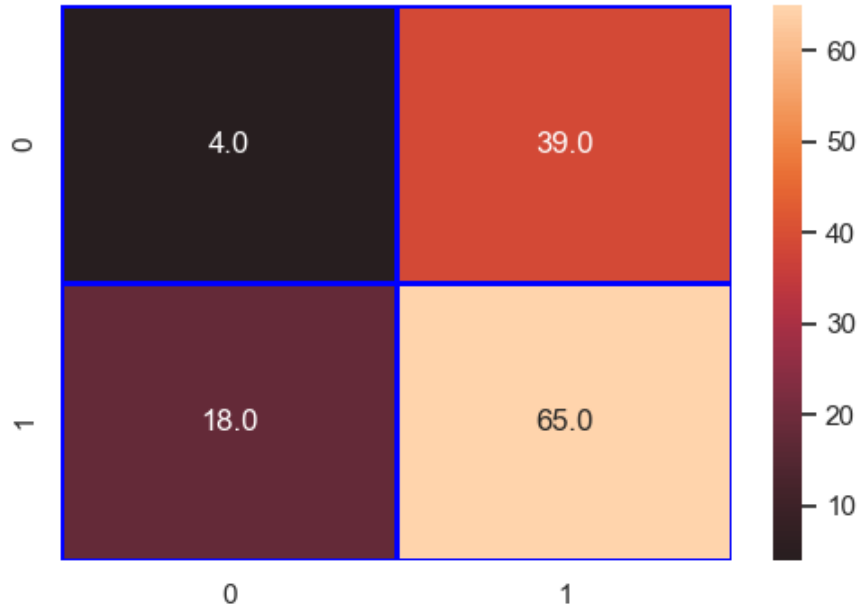


==========

Modele name : RandomForestClassifier
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.5686 0.6471 0.5294 0.6   0.64   0.62   0.52   0.74   0.62   0.6   ]
10 K-Fold Average Accuracy_score : 60.85 %
Accuracy_score: 54.76 %
Loss: 45.24 %
Cohen_kappa_score: -14.04 %
Classification_report:

	precision	recall	f1-score	support
0	0.18	0.09	0.12	43
1	0.62	0.78	0.70	83
accuracy			0.55	126
macro avg	0.40	0.44	0.41	126
weighted avg	0.47	0.55	0.50	126

confusion_matrix:
 [[4 39]
 [18 65]]

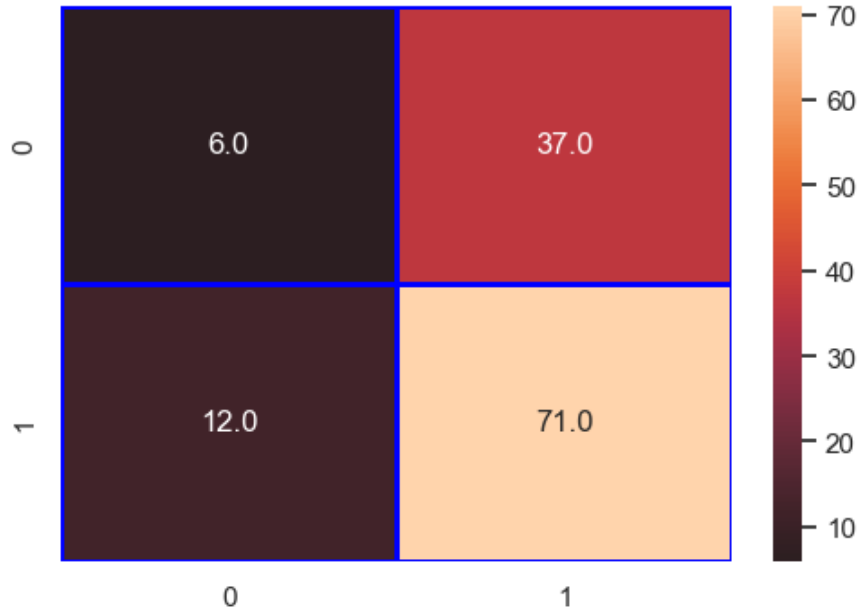


==========

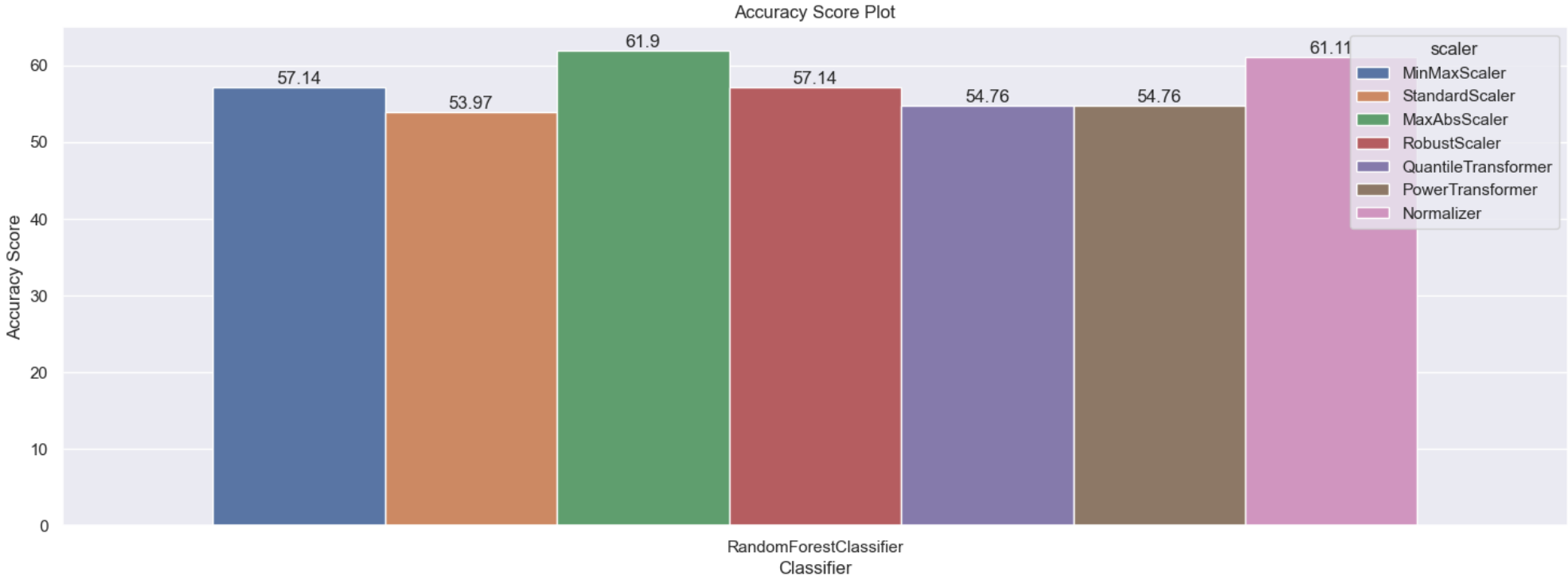
Modele name : RandomForestClassifier
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.5294 0.549 0.6078 0.62   0.6   0.62   0.7   0.66   0.62   0.58]
10 K-Fold Average Accuracy_score : 60.86 %
Accuracy_score: 61.11 %
Loss: 38.89 %
Cohen_kappa_score: -0.59 %
Classification_report:

	precision	recall	f1-score	support
0	0.33	0.14	0.20	43
1	0.66	0.86	0.74	83
accuracy			0.61	126
macro avg	0.50	0.50	0.47	126
weighted avg	0.55	0.61	0.56	126

confusion_matrix:
 [[6 37]
 [12 71]]


6.0
37.0
12.0
71.0
0
1
0
1
70
60
50
40
30
20
10

==========


Accuracy Score Plot
Accuracy Score
0
10
20
30
40
50
60
57.14
53.97
61.9
57.14
54.76
54.76
61.11
RandomForestClassifier
Classifier
scaler
MinMaxScaler
StandardScaler
MaxAbsScaler
RobustScaler
QuantileTransformer
PowerTransformer
Normalizer

Done...

In [ ]:
```
1
```