

In [1]:

```
1 import numpy as np
2 import pandas as pd
3 from collections import Counter
4
5 import seaborn as sns
6 sns.set(rc={'figure.figsize':(6,4)})
7 import matplotlib.pyplot as plt
8 %matplotlib inline
9
10 from tqdm import tqdm
11 import random
12 import pickle
13 import time
14
15 from sklearn.model_selection import train_test_split
16 from sklearn.preprocessing import LabelEncoder
17
18 from sklearn.preprocessing import MinMaxScaler
19 from sklearn.preprocessing import StandardScaler
20 from sklearn.preprocessing import MaxAbsScaler
21 from sklearn.preprocessing import RobustScaler
22 from sklearn.preprocessing import QuantileTransformer
23 from sklearn.preprocessing import PowerTransformer
24 from sklearn.preprocessing import Normalizer
25
26 from sklearn.linear_model import LogisticRegression
27 from sklearn.neighbors import KNeighborsClassifier
28 from sklearn.naive_bayes import GaussianNB
29 from sklearn.tree import DecisionTreeClassifier
30 from sklearn.ensemble import RandomForestClassifier
31
32 from sklearn.model_selection import cross_val_score
33 from sklearn.metrics import accuracy_score
34 from sklearn.metrics import log_loss
35 from sklearn.metrics import cohen_kappa_score
36 from sklearn.metrics import confusion_matrix
37 from sklearn import metrics
38
39 # for ignore warnings
40 import warnings
41 warnings.filterwarnings("ignore")
42
43 plot_data_list = []
```

```
In [2]: 1 df = pd.read_csv('Dataset\LS Dataset(all)_RP.csv')
        2 df.head()
```

Out[2]:

	Age(years)	Weight(kg)	Height(cm)	BMI	Gender	Profession	smoke	Exercise	cereal	salad	...	Poor and unplanned work	Lack of career development	Feeling of powerlessness	Lack of financial security	Unable to satisfy all stakeholders	High_
0	42	61.0	165	22.41	Male	Sitting-Job	No	Moderate	Once	Everyday	...	Low	Low	Low	Low	Low	
1	30	49.0	165	18.00	Female	Other	No	Inactive	Twice	Rarely	...	Average	Average	Low	Low	Low	
2	52	60.0	159	23.73	Male	Moving-Job	Yes	Moderate	Twice	Rarely	...	Very low	Very low	Very low	Very low	Very low	
3	46	61.0	172	20.62	Male	Other	No	Low	Thrice	Sometimes	...	Average	Average	Very low	Average	Average	
4	45	65.0	155	27.06	Female	Moving-Job	No	Inactive	Twice	Everyday	...	Very low	Very High	Average	Average	Low	

5 rows × 35 columns

```
In [3]: 1 df.shape
```

Out[3]: (375, 35)

```
In [4]: 1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375 entries, 0 to 374
Data columns (total 35 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Age(years)                           375 non-null    int64
1   Weight(kg)                           375 non-null    float64
2   Height(cm)                           375 non-null    int64
3   BMI                                  375 non-null    float64
4   Gender                               375 non-null    object
5   Profession                           375 non-null    object
6   smoke                                375 non-null    object
7   Exercise                             375 non-null    object
8   cereal                               375 non-null    object
9   salad                                375 non-null    object
10  vegetables                            375 non-null    object
11  grains                               375 non-null    object
12  sweets                               375 non-null    object
13  Sweets_in_a_week                     375 non-null    object
14  Refined_Sugar                        375 non-null    object
15  Milk_Products_consumption            375 non-null    object
16  Milk_Quantity(ml/day)                375 non-null    object
17  Pregnancy                            375 non-null    object
18  History_of_Diabetes                  375 non-null    object
19  Anxiety                              375 non-null    object
20  Stress_Workload                      375 non-null    object
21  Poor                                  375 non-null    object
    family income
22  Pressure of working                  375 non-null    object
23  Frequent travel                      375 non-null    object
24  Monotonous work                     375 non-null    object
25  Poor and unplanned work              375 non-null    object
26  Lack of career development            375 non-null    object
27  Feeling of powerlessness              375 non-null    object
28  Lack of financial security            375 non-null    object
29  Unable to satisfy all stakeholders    375 non-null    object
30  High_Blood_Pressure                  375 non-null    object
31  Blood_Sugar_Fasting                  375 non-null    object
32  Blood_Sugar_Post_Meal                 375 non-null    object
33  HbA1c_Glycated_Haemoglobin           375 non-null    object
34  Class                                375 non-null    object
dtypes: float64(2), int64(2), object(31)
memory usage: 102.7+ KB
```

In [5]:

1 round(df.describe(),2)

Out[5]:

	Age(years)	Weight(kg)	Height(cm)	BMI
count	375.00	375.00	375.00	375.00
mean	42.41	64.94	162.57	24.63
std	15.29	11.99	10.08	4.45
min	4.00	20.00	100.00	10.97
25%	30.00	57.00	155.50	22.20
50%	40.00	65.00	163.00	24.22
75%	53.00	72.00	168.00	27.15
max	90.00	102.00	193.00	52.00

```
In [6]: 1 # check null
        2 df.isnull().sum()
```

```
Out[6]: Age(years)          0
Weight(kg)          0
Height(cm)          0
BMI                0
Gender              0
Profession          0
smoke               0
Exercise            0
cereal              0
salad               0
vegetables          0
grains              0
sweets              0
Sweets_in_a_week    0
Refined_Sugar        0
Milk_Products_consumption  0
Milk Quantity(ml/day)  0
Pregnancy            0
History_of_Diabetes  0
Anxiety              0
Stress_Workload      0
Poor \nfamily income  0
Pressure of\n working  0
Frequent \ntravel    0
Monotonous\n work     0
Poor and unplanned work  0
Lack of career development  0
Feeling of powerlessness  0
Lack of financial security  0
Unable to satisfy all stakeholders  0
High_Blood_Pressure  0
Blood_Sugar_Fasting  0
Blood_Sugar_Post_Meal  0
HbA1c_Glycated_Haemoglobin  0
Class                0
dtype: int64
```

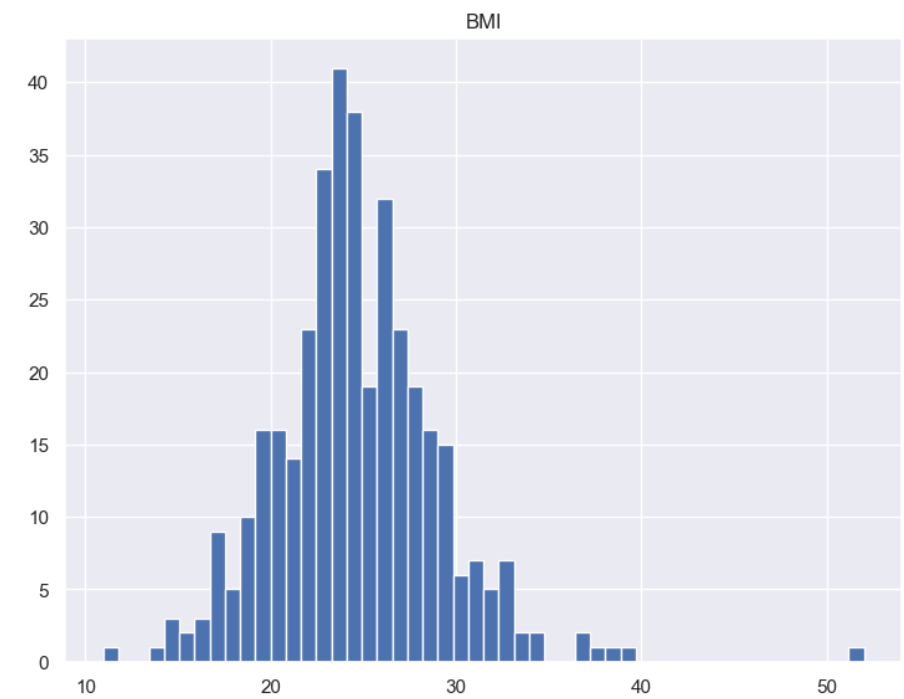
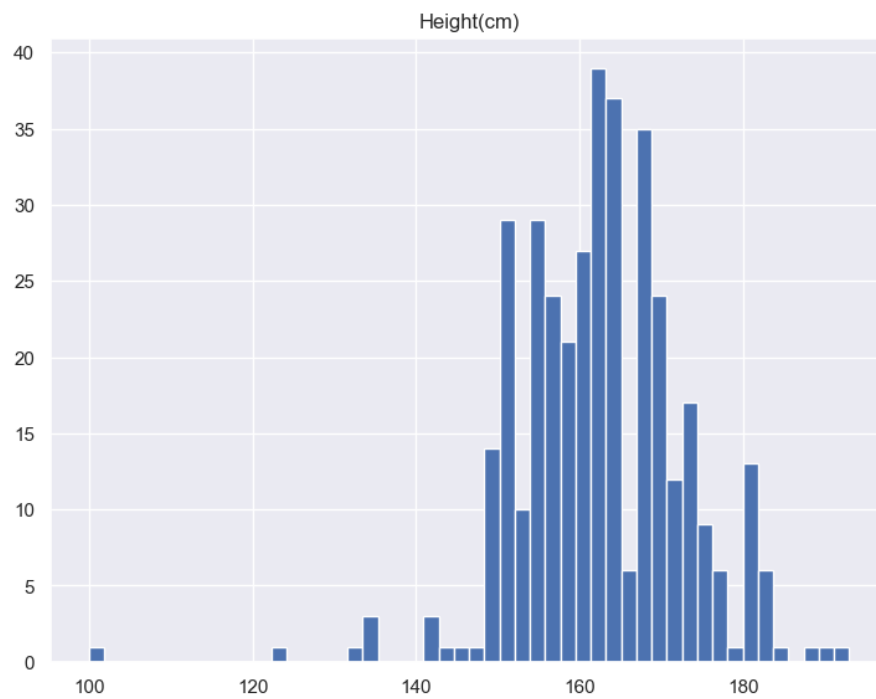
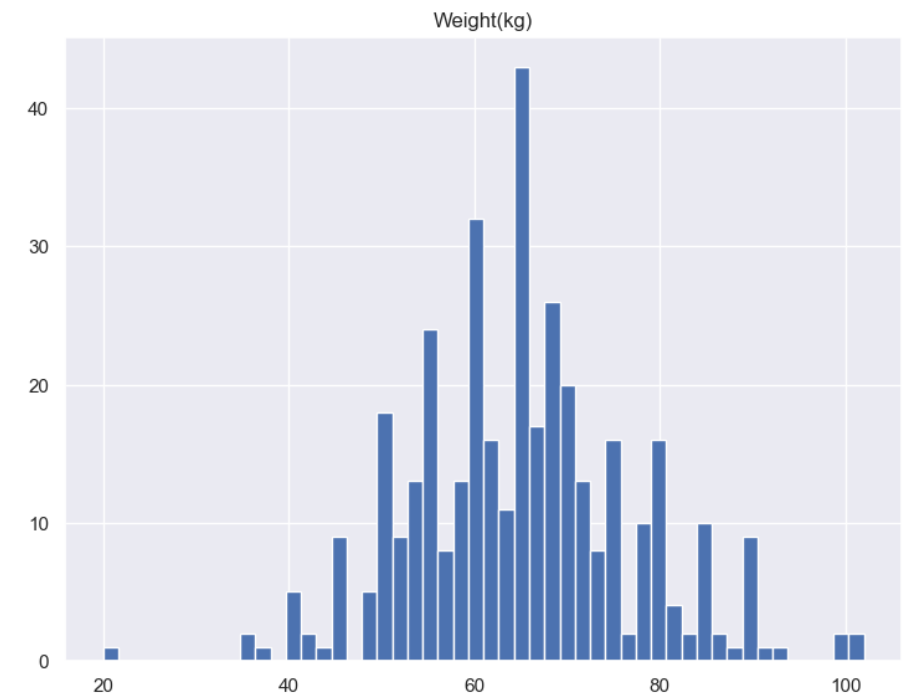
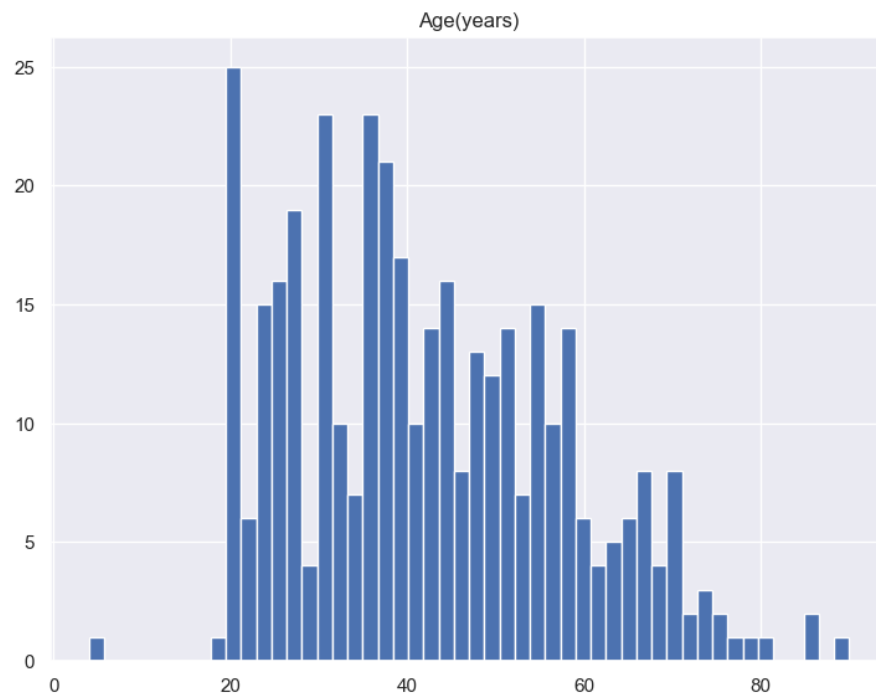
```
In [7]: 1 # List of numerical variables
2 numerical_features = [feature for feature in df.columns if df[feature].dtypes != 'O']
3
4 print('Number of numerical variables: ', len(numerical_features))
5
6 numerical_variables_df = df[numerical_features]
7
8 # visualise the numerical variables
9 df[numerical_features].head()
```

Number of numerical variables: 4

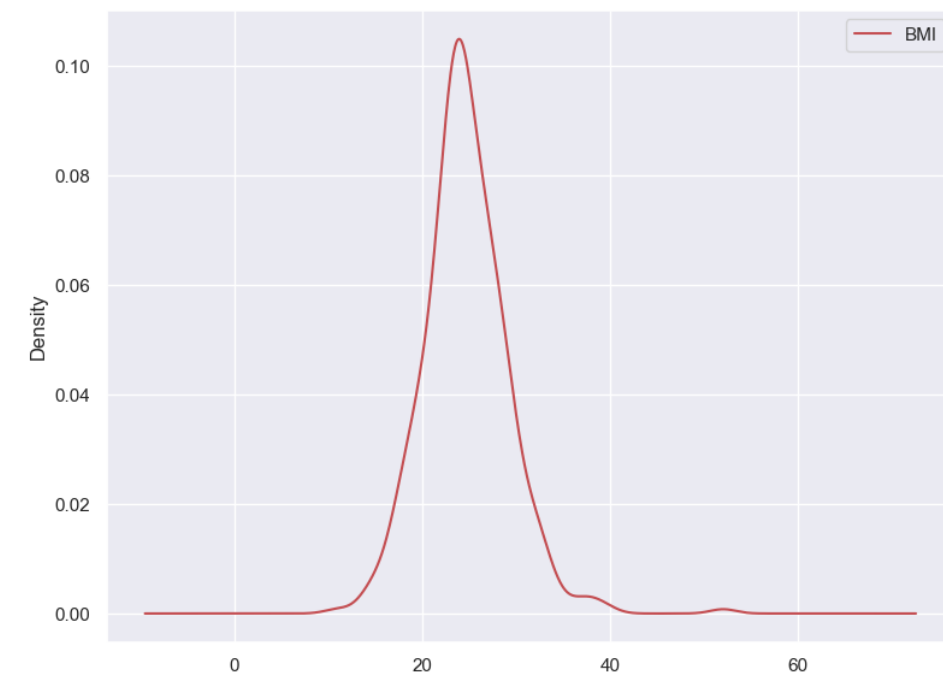
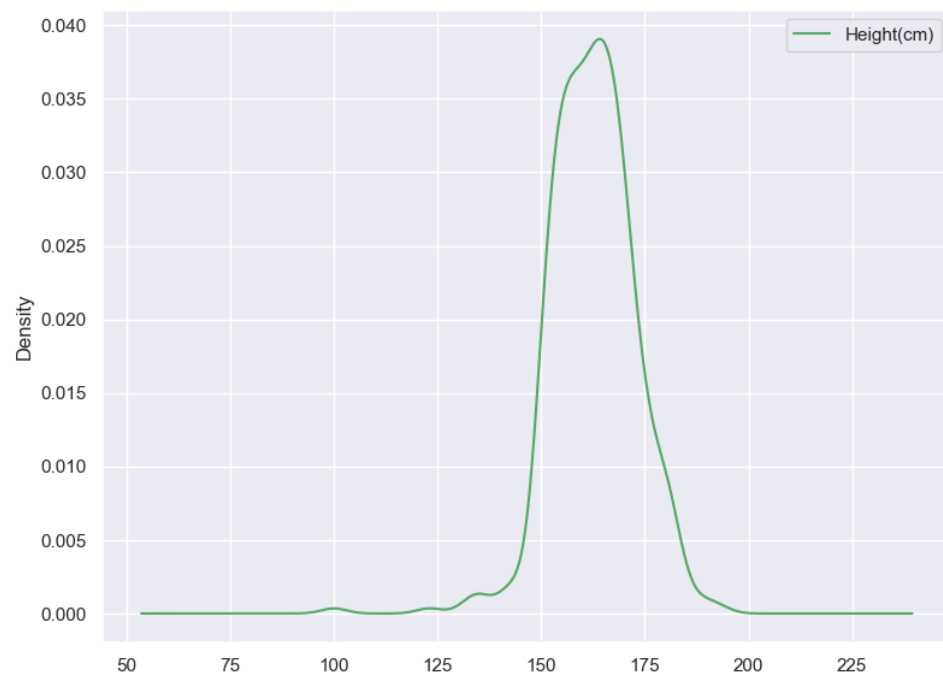
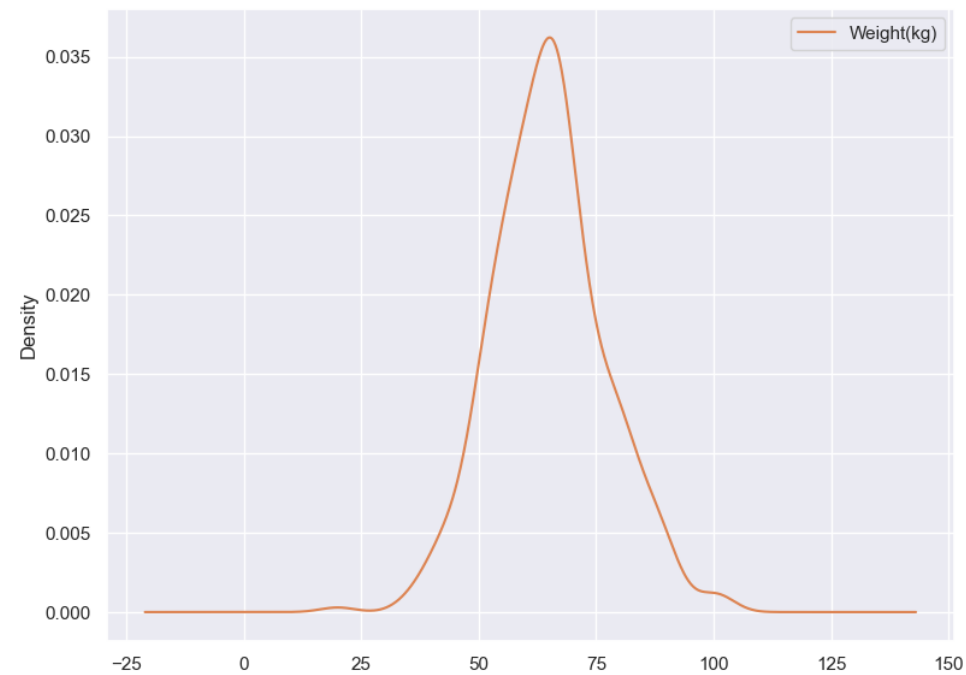
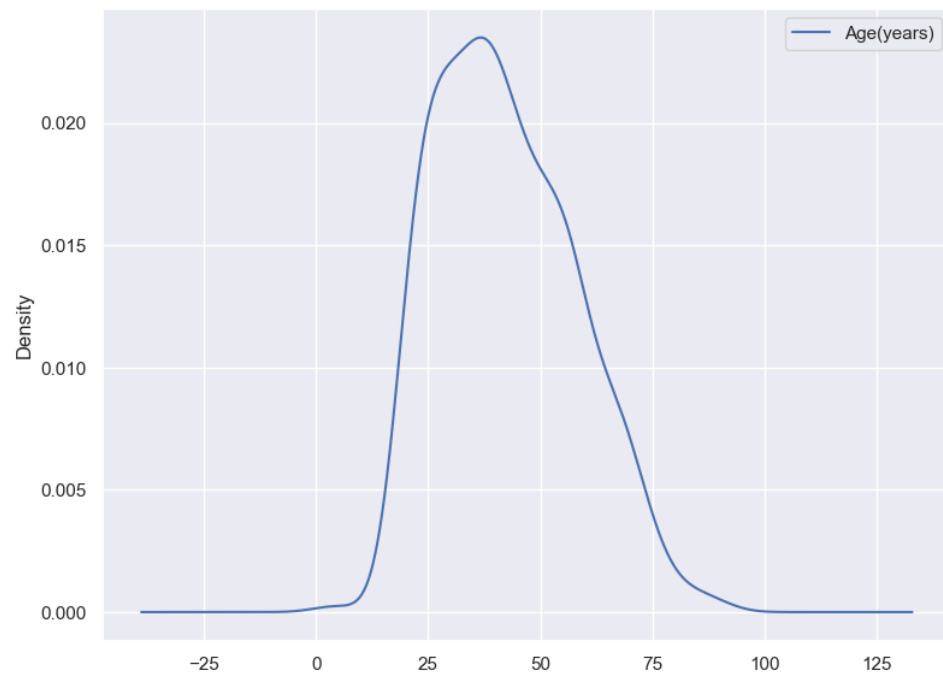
Out[7]:

	Age(years)	Weight(kg)	Height(cm)	BMI
0	42	61.0	165	22.41
1	30	49.0	165	18.00
2	52	60.0	159	23.73
3	46	61.0	172	20.62
4	45	65.0	155	27.06

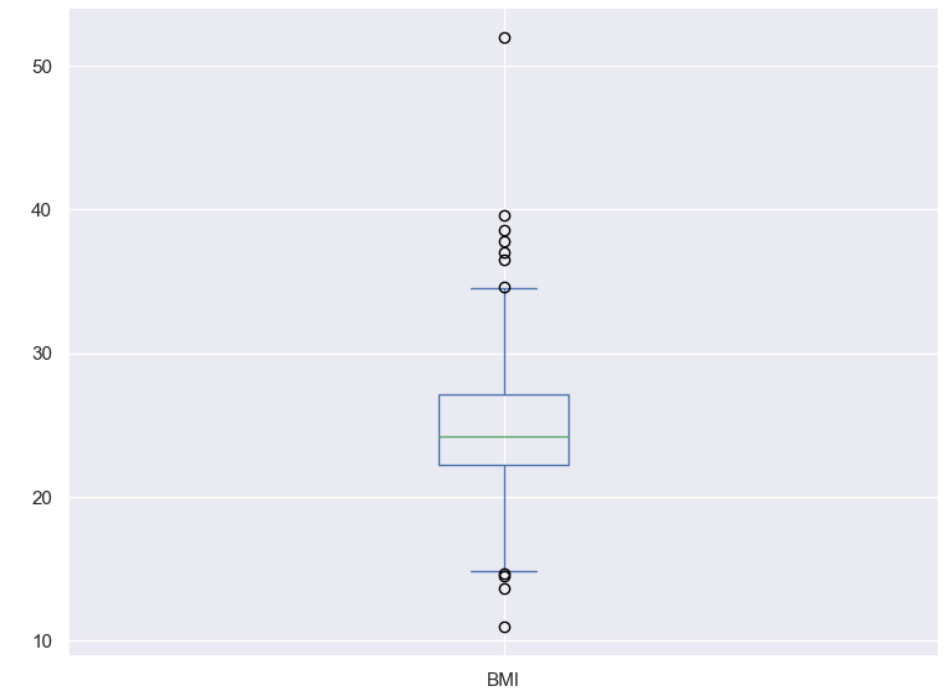
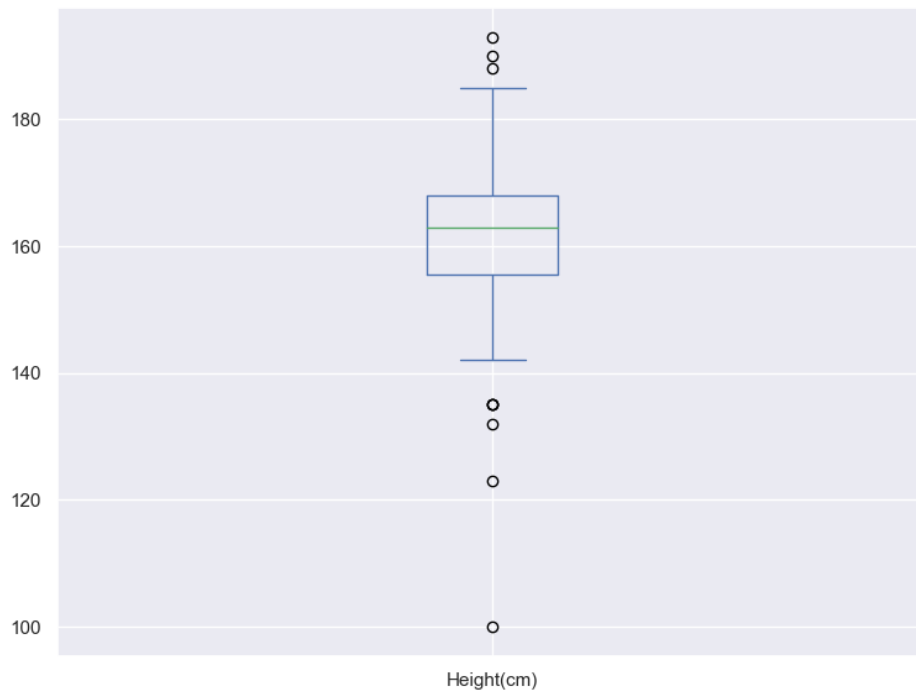
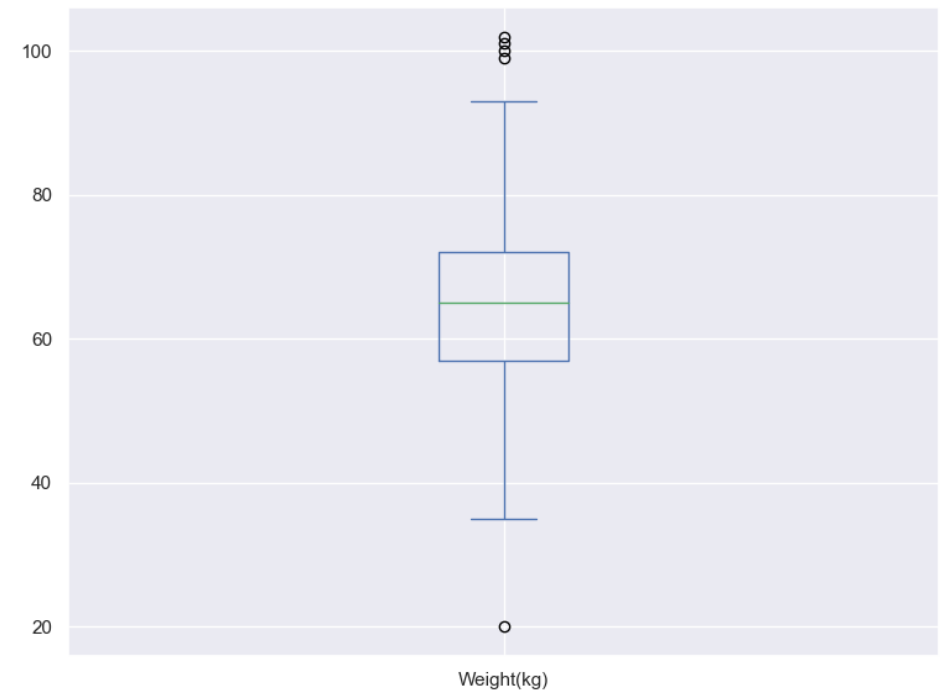
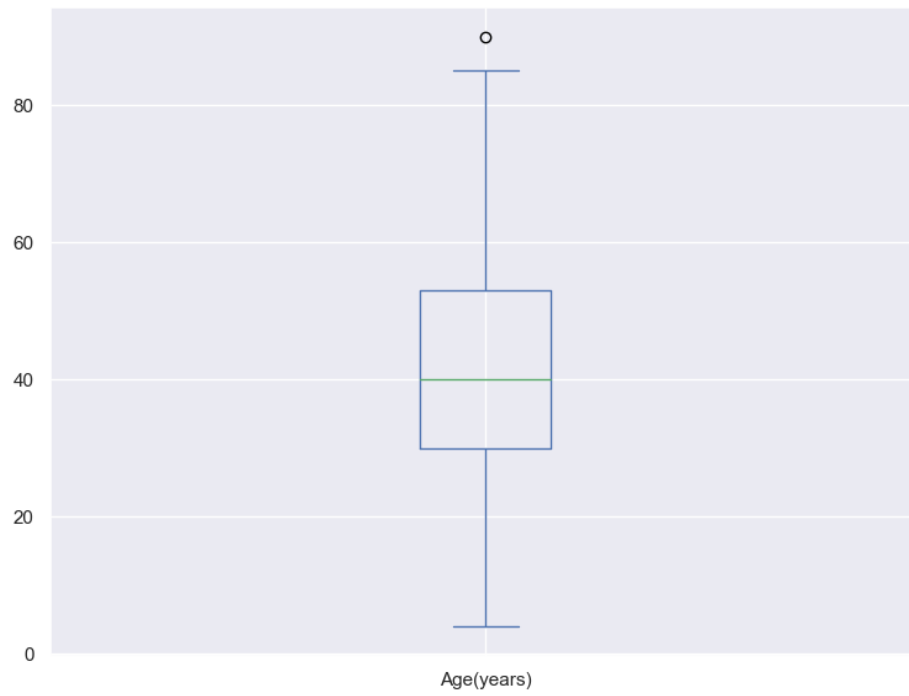
```
In [8]: 1 # Histogram:distribution of a single numerical variable
        2 numerical_variables_df.hist(bins=50, figsize=(20, 15))
        3 plt.show()
```


```
In [9]: 1 # Density plots for all attributes to visualize the distribution of each attribute
        2 numerical_variables_df.plot(kind='density', subplots=True, layout=(2,2), figsize=(20, 15), sharex=False)
        3 plt.show()
```



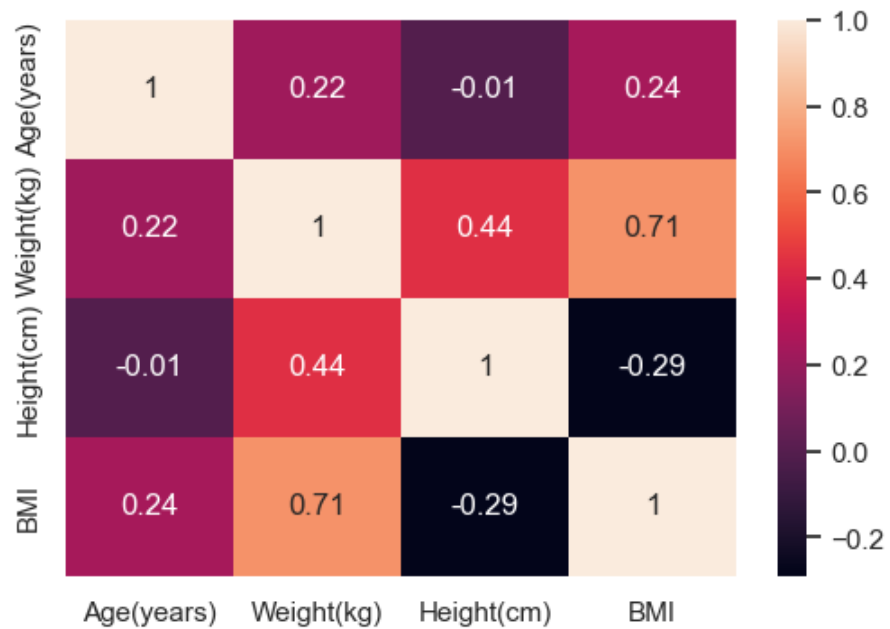
```
In [10]: 1 # Box Plot (Box-and-Whisker Plot): Summarizes the distribution of a numerical variable's
          2 numerical_variables_df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False, figsize=(20,15))
          3 plt.show()
```



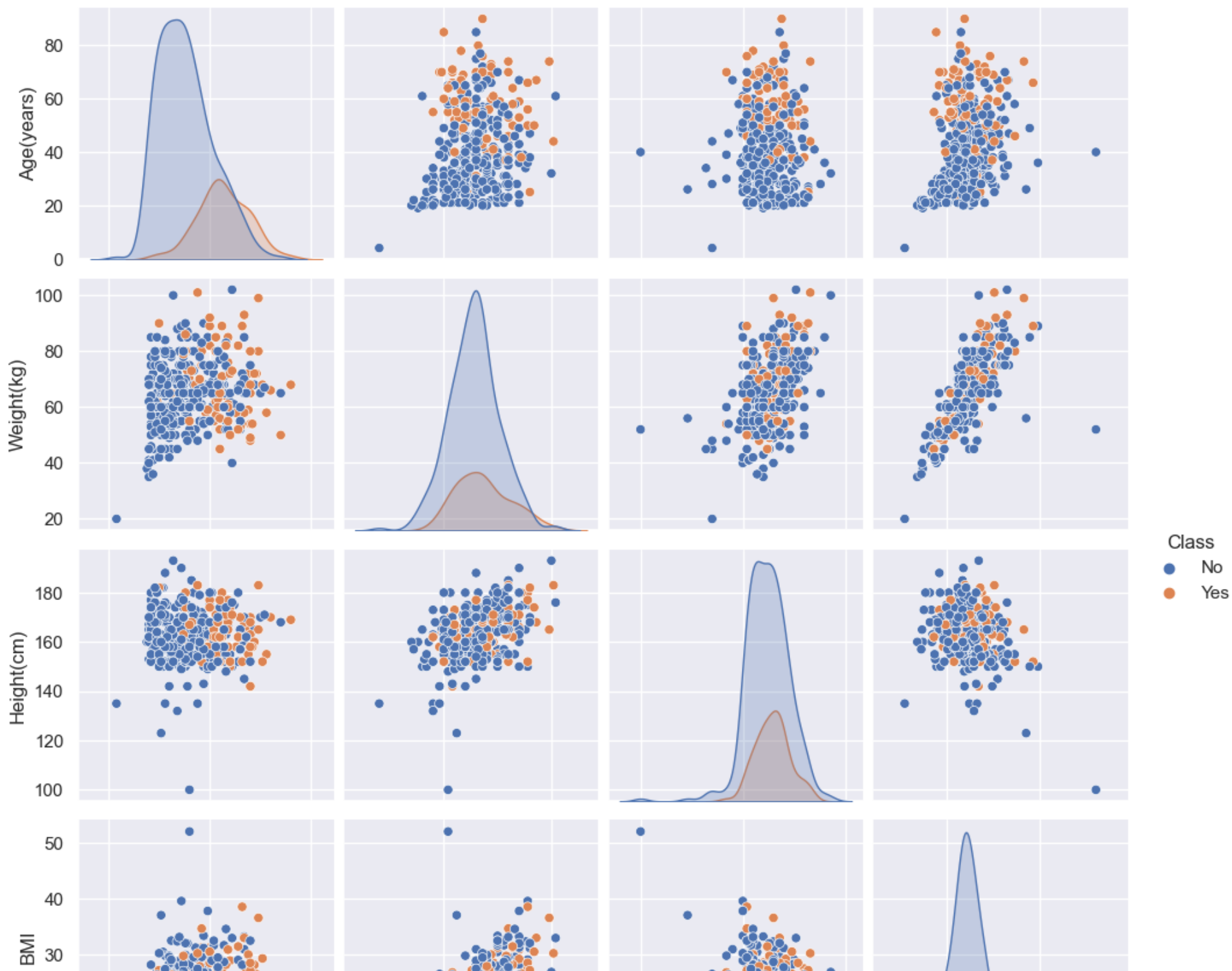
observation : outlier is detected in all features

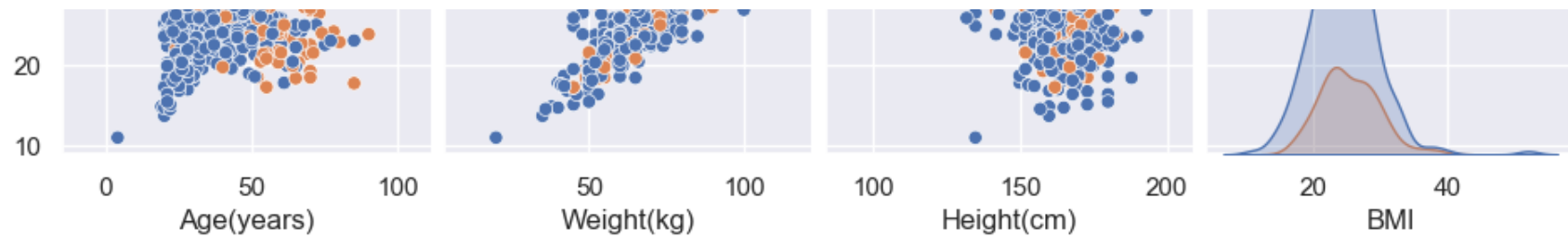
In [11]:

```
1 # Correlation between the different characteristics. Closer to 1 better is the correlation.  
2 sns.heatmap(round(numerical_variables_df.corr(method='pearson'),2), annot = True)  
3 plt.show()
```



```
In [14]: 1 numerical_variables_df_class = pd.concat([numerical_variables_df, df['Class']], axis=1)
          2 # Pairplot
          3 sns.pairplot(numerical_variables_df_class, hue='Class')
          4 plt.show()
```



```
In [37]: 1 # list of categorical variables
2 categorical_features = [feature for feature in df.columns if df[feature].dtypes == 'O']
3
4 print('Number of categorical variables: ', len(categorical_features))
5
6 categorical_variables_df = df[categorical_features]
7
8 # visualise the numerical variables
9 df[categorical_features].head()
```

Number of categorical variables: 31

Out[37]:

	Gender	Profession	smoke	Exercise	cereal	salad	vegetables	grains	sweets	Sweets_in_a_week	...	Poor and unplanned work	Lack of career development	Feeling of powerlessness	Lack of financial security	Unable to satisfy all stakeholders
0	Male	Sitting-Job	No	Moderate	Once	Everyday	Twice	Rice-Wheat	Yes	Weekly	...	Low	Low	Low	Low	Low
1	Female	Other	No	Inactive	Twice	Rarely	Twice	Rice-Wheat-Maida	Yes	Weekly	...	Average	Average	Low	Low	Low
2	Male	Moving-Job	Yes	Moderate	Twice	Rarely	Twice	Polished-Rice	Yes	Rarely	...	Very low	Very low	Very low	Very low	Very low
3	Male	Other	No	Low	Thrice	Sometimes	Twice	Wheat	Yes	Weekly	...	Average	Average	Very low	Average	Average
4	Female	Moving-Job	No	Inactive	Twice	Everyday	Twice	Brown-Rice	Yes	Weekly	...	Very low	Very High	Average	Average	Low

5 rows × 31 columns



In [38]:

```
1 for col in categorical_variables_df.columns:
2     print(f"Unique values in {col} is : {sorted(categorical_variables_df[col].unique())}, \
3         Count : {len(sorted(categorical_variables_df[col].unique()))} \n")
```

Unique values in Gender is : ['Female', 'Male'], Count : 2

Unique values in Profession is : ['Business', 'Business-sitting-job ', 'Housewife', 'Moving-Job', 'Other', 'Sitting-Job', 'other'], Count : 7

Unique values in smoke is : ['No', 'YES', 'Yes', 'yes'], Count : 4

Unique values in Exercise is : ['High', 'Inactive', 'Intense', 'Low', 'Moderate', 'Very intense'], Count : 6

Unique values in cereal is : ['More than three times', 'None', 'Once', 'Thrice', 'Thrice in a day', 'Twice'], Count : 6

Unique values in salad is : ['1 bowl in lunch and/or dinner', '1/2 bowl in lunch and/or dinner', 'Everyday', 'Frequently', 'Never', 'None', 'Rarely', 'Rarely (Weekly)', 'Sometimes'], Count : 9

Unique values in vegetables is : ['Daily', 'Never', 'None', 'Once', 'Once in a week', 'Sometimes', 'Twice', 'Twice a day', 'Weekly'], Count : 9

Unique values in grains is : ['Brown-Rice', 'Brown-Rice-Millet', 'Brown-Rice-Millet-Maida', 'Brown-Rice-Wheat', 'Brown-Rice-Wheat-Millet', 'Millet', 'Polished-Rice', 'Polished-Rice-Brown-Rice-Wheat', 'Polished-Rice-Brown-Rice-Wheat-Millet', 'Polished-Rice-Maida', 'Polished-Rice-Millet', 'Polished-Rice-Wheat', 'Polished-Rice-Wheat-Maida-Millet', 'Polished-Rice-Wheat-Millet', 'Rice-Wheat', 'Rice-Wheat-Brown-Rice', 'Rice-Wheat-Brown-Rice-Maida', 'Rice-Wheat-Maida', 'Rice-Wheat-Millet', 'Rice-Wheat-Millet-Maida', 'Wheat', 'Wheat-Brown-Rice', 'Wheat-Brown-Rice-Millet', 'Wheat-Maida', 'Wheat-Millet', 'Wheat-Millet-Maida'], Count : 26

Unique values in sweets is : ['No', 'Yes'], Count : 2

Unique values in Sweets_in_a_week is : ['Daily', 'Monthly', 'Never', 'Rarely', 'Weekly'], Count : 5

Unique values in Refined_Sugar is : ['No', 'Yes'], Count : 2

Unique values in Milk_Products_consumption is : ['No', 'Yes'], Count : 2

Unique values in Milk Quantity(ml/day) is : ['100-200', '200-400', '400-600', 'More than 600 '], Count : 4

Unique values in Pregnancy is : ['No', 'Not-Applicable', 'Yes'], Count : 3

Unique values in History_of_Diabetes is : ['No', 'Yes'], Count : 2

Unique values in Anxiety is : ['No', 'Yes'], Count : 2

Unique values in Stress_Workload is : ['Low', 'Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 7

Unique values in Poor family income is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Pressure of working is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Frequent

travel is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Monotonous
work is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Poor and unplanned work is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Lack of career development is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Feeling of powerlessness is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Lack of financial security is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in Unable to satisfy all stakeholders is : ['Average', 'High', 'Low', 'Very High', 'Very high', 'Very low'], Count : 6

Unique values in High_Blood_Pressure is : ['No', 'Yes'], Count : 2

Unique values in Blood_Sugar_Fasting is : ['High', 'Low', 'Moderate'], Count : 3

Unique values in Blood_Sugar_Post_Meal is : ['High', 'Low', 'Moderate'], Count : 3

Unique values in HbA1c_Glycated_Haemoglobin is : ['High', 'Low', 'Moderate'], Count : 3

Unique values in Class is : ['No', 'Yes'], Count : 2

In [39]:

```
1 replacement_dict = {'other': 'Other'}
2 categorical_variables_df['Profession'].replace(replacement_dict, inplace=True)
3
4 replacement_dict = {'YES': 'Yes', 'yes': 'Yes'}
5 categorical_variables_df['smoke'].replace(replacement_dict, inplace=True)
6
7 replacement_dict = {'Thrice in a day': 'Thrice'}
8 categorical_variables_df['cereal'].replace(replacement_dict, inplace=True)
9
10 replacement_dict = {'None': 'Never', 'Rarely (Weekly)': 'Rarely', 'Sometimes': 'Rarely'}
11 categorical_variables_df['salad'].replace(replacement_dict, inplace=True)
12
13 replacement_dict = {'None': 'Never', 'Once in a week': 'Once', 'Sometimes': 'Rarely', 'Twice a day': 'Twice'}
14 categorical_variables_df['vegetables'].replace(replacement_dict, inplace=True)
15
16 replacement_dict = {' Low': 'Low', 'Very High': 'Very high'}
17 categorical_variables_df['Stress_Workload'].replace(replacement_dict, inplace=True)
18
19 replacement_dict = {'Very High': 'Very high'}
20 categorical_variables_df['Poor \nfamil y income'].replace(replacement_dict, inplace=True)
21
22 replacement_dict = {'Very High': 'Very high'}
23 categorical_variables_df['Pressure of\n working'].replace(replacement_dict, inplace=True)
24
25 replacement_dict = {'Very High': 'Very high'}
26 categorical_variables_df['Frequent \ntravel'].replace(replacement_dict, inplace=True)
27
28 replacement_dict = {'Very High': 'Very high'}
29 categorical_variables_df['Monotonous\n work'].replace(replacement_dict, inplace=True)
30
31 replacement_dict = {'Very High': 'Very high'}
32 categorical_variables_df['Poor and unplanned work'].replace(replacement_dict, inplace=True)
33
34 replacement_dict = {'Very High': 'Very high'}
35 categorical_variables_df['Lack of career development'].replace(replacement_dict, inplace=True)
36
37 replacement_dict = {'Very High': 'Very high'}
38 categorical_variables_df['Feeling of powerlessness'].replace(replacement_dict, inplace=True)
39
40 replacement_dict = {'Very High': 'Very high'}
41 categorical_variables_df['Lack of financial security'].replace(replacement_dict, inplace=True)
42
43 replacement_dict = {'Very High': 'Very high'}
44 categorical_variables_df['Unable to satisfy all stakeholders'].replace(replacement_dict, inplace=True)
45
46 for col in categorical_variables_df.columns:
47     print(f"Unique values in {col} is : {sorted(categorical_variables_df[col].unique())}, \
```



```
Count : {len(sorted(categorical_variables_df[col].unique()))} \n"
```

Unique values in Gender is : ['Female', 'Male'], Count : 2

Unique values in Profession is : ['Business', 'Business-sitting-job ', 'Housewife', 'Moving-Job', 'Other', 'Sitting-Job'], Count : 6

Unique values in smoke is : ['No', 'Yes'], Count : 2

Unique values in Exercise is : ['High', 'Inactive', 'Intense', 'Low', 'Moderate', 'Very intense'], Count : 6

Unique values in cereal is : ['More than three times', 'None', 'Once', 'Thrice', 'Twice'], Count : 5

Unique values in salad is : ['1 bowl in lunch and/or dinner', '1/2 bowl in lunch and/or dinner', 'Everyday', 'Frequently', 'Never', 'Rarely'], Count : 6

Unique values in vegetables is : ['Daily', 'Never', 'Once', 'Rarely', 'Twice', 'Weekly'], Count : 6

Unique values in grains is : ['Brown-Rice', 'Brown-Rice-Millet', 'Brown-Rice-Millet-Maida', 'Brown-Rice-Wheat', 'Brown-Rice-Wheat-Millet', 'Millet', 'Polished-Rice', 'Polished-Rice-Brown-Rice-Wheat', 'Polished-Rice-Brown-Rice-Wheat-Millet', 'Polished-Rice-Maida', 'Polished-Rice-Millet', 'Polished-Rice-Wheat', 'Polished-Rice-Wheat-Maida-Millet', 'Polished-Rice-Wheat-Millet', 'Rice-Wheat', 'Rice-Wheat-Brown-Rice', 'Rice-Wheat-Brown-Rice-Maida', 'Rice-Wheat-Maida', 'Rice-Wheat-Millet', 'Rice-Wheat-Millet-Maida', 'Wheat', 'Wheat-Brown-Rice', 'Wheat-Brown-Rice-Millet', 'Wheat-Maida', 'Wheat-Millet', 'Wheat-Millet-Maida'], Count : 26

Unique values in sweets is : ['No', 'Yes'], Count : 2

Unique values in Sweets_in_a_week is : ['Daily', 'Monthly', 'Never', 'Rarely', 'Weekly'], Count : 5

Unique values in Refined_Sugar is : ['No', 'Yes'], Count : 2

Unique values in Milk_Products_consumption is : ['No', 'Yes'], Count : 2

Unique values in Milk Quantity(ml/day) is : ['100-200', '200-400', '400-600', 'More than 600 '], Count : 4

Unique values in Pregnancy is : ['No', 'Not-Applicable', 'Yes'], Count : 3

Unique values in History_of_Diabetes is : ['No', 'Yes'], Count : 2

Unique values in Anxiety is : ['No', 'Yes'], Count : 2

Unique values in Stress_Workload is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Poor family income is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Pressure of working is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Frequent travel is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Monotonus

work is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Poor and unplanned work is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Lack of career development is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Feeling of powerlessness is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Lack of financial security is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in Unable to satisfy all stakeholders is : ['Average', 'High', 'Low', 'Very high', 'Very low'], Count : 5

Unique values in High_Blood_Pressure is : ['No', 'Yes'], Count : 2

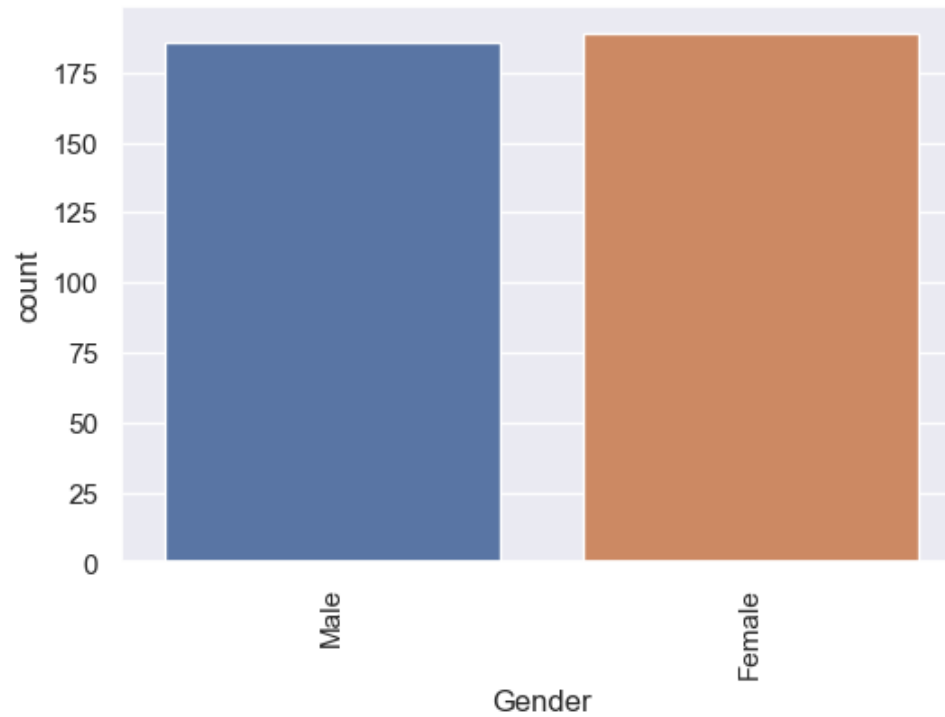
Unique values in Blood_Sugar_Fasting is : ['High', 'Low', 'Moderate'], Count : 3

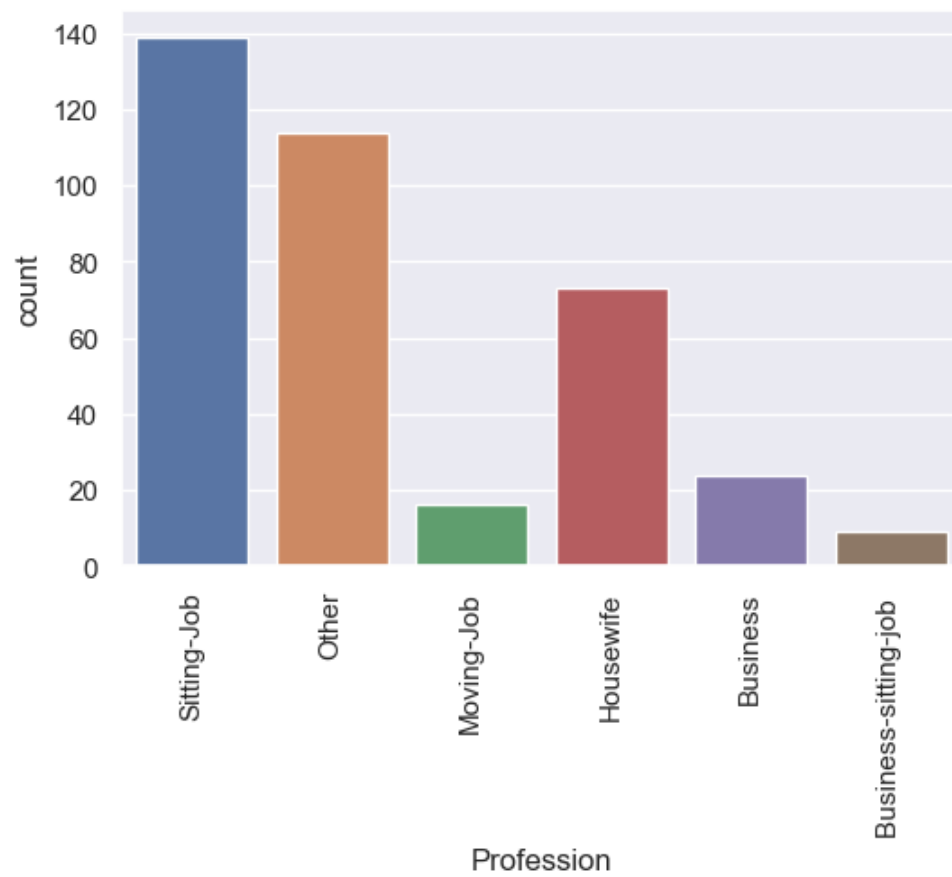
Unique values in Blood_Sugar_Post_Meal is : ['High', 'Low', 'Moderate'], Count : 3

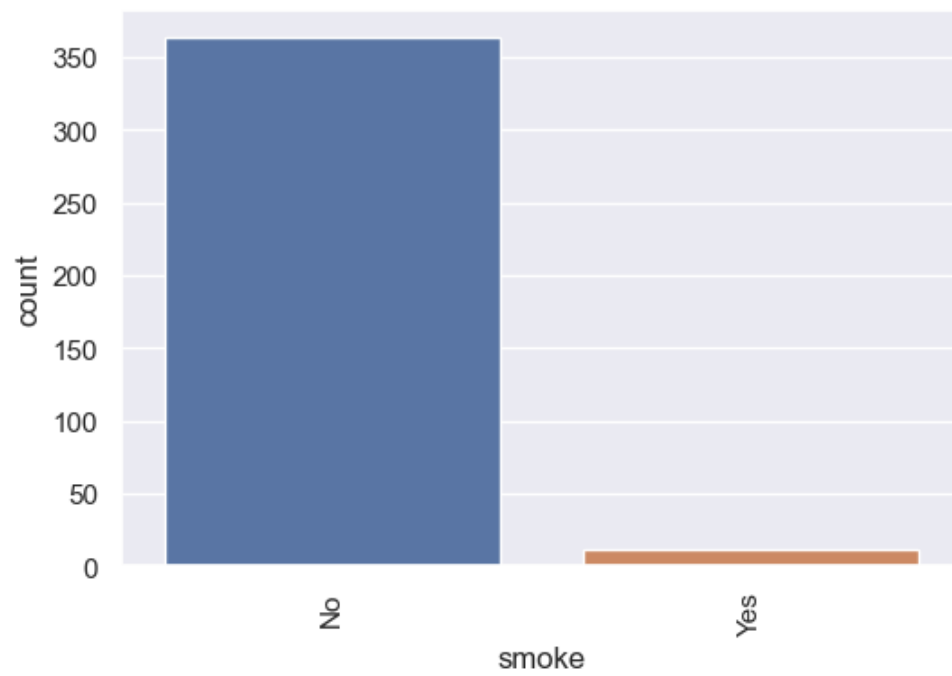
Unique values in HbA1c_Glycated_Haemoglobin is : ['High', 'Low', 'Moderate'], Count : 3

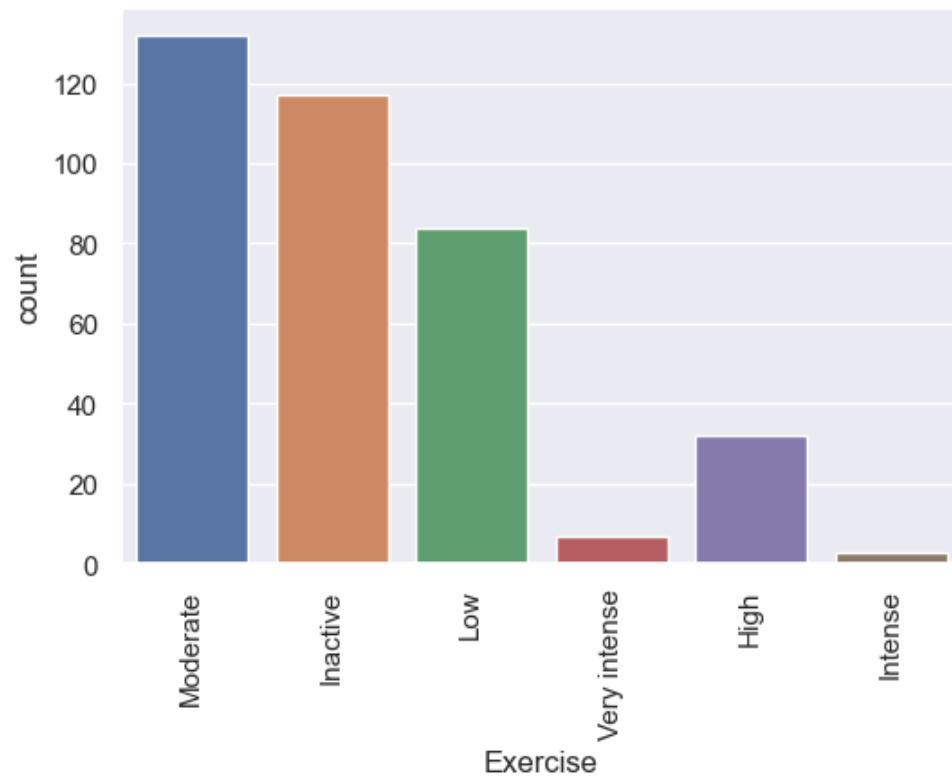
Unique values in Class is : ['No', 'Yes'], Count : 2

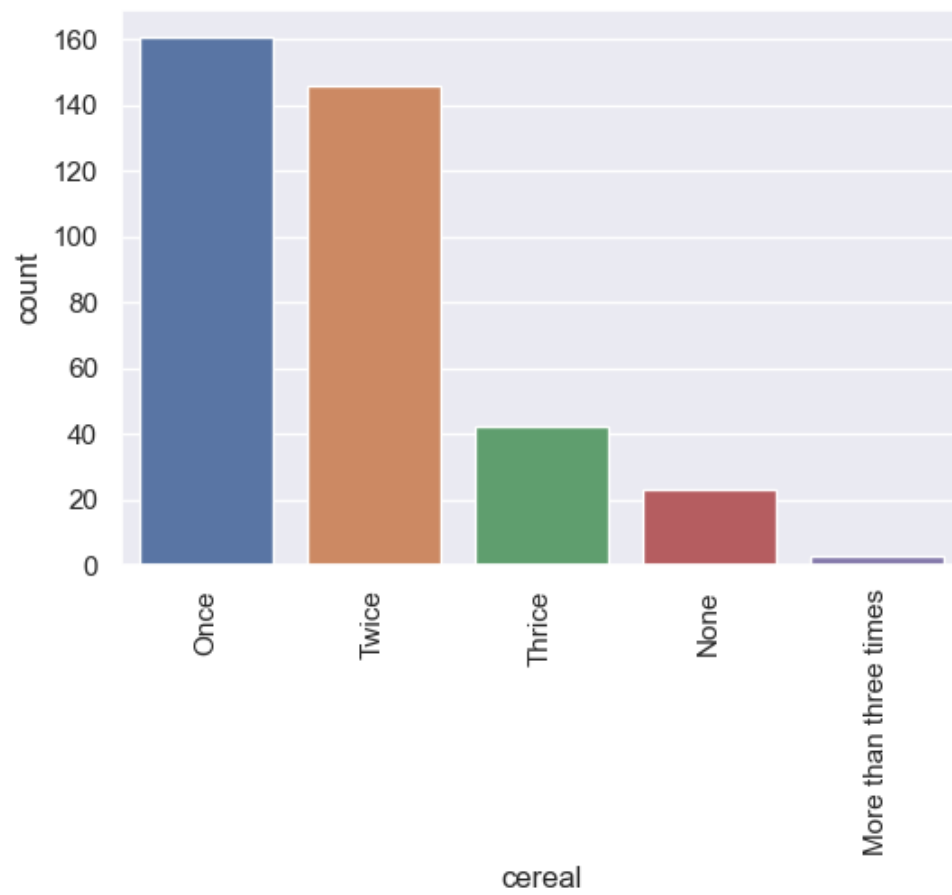
```
In [40]: 1 # Count Plot: Displays the count of occurrences for each category as bars.  
2 for col in categorical_variables_df.columns:  
3     plt.xticks(rotation=90)  
4     sns.countplot(data=categorical_variables_df, x=col)  
5     plt.show()
```

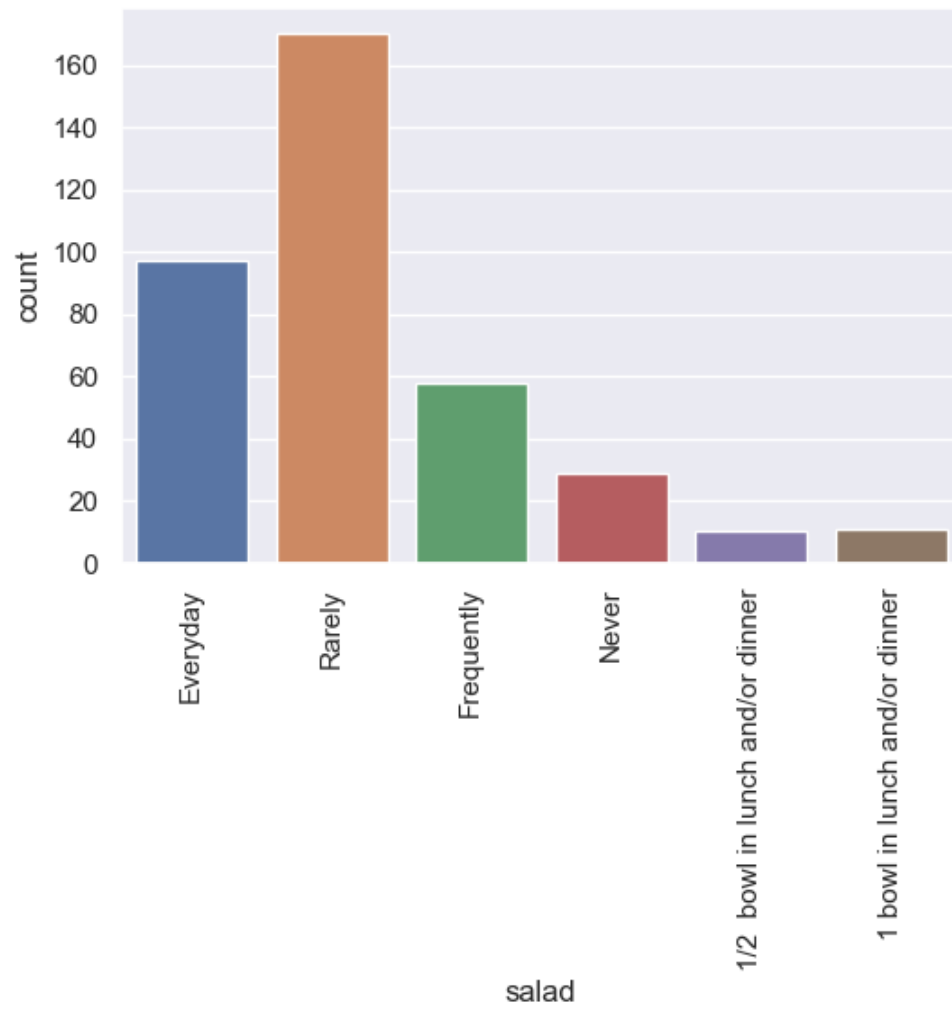




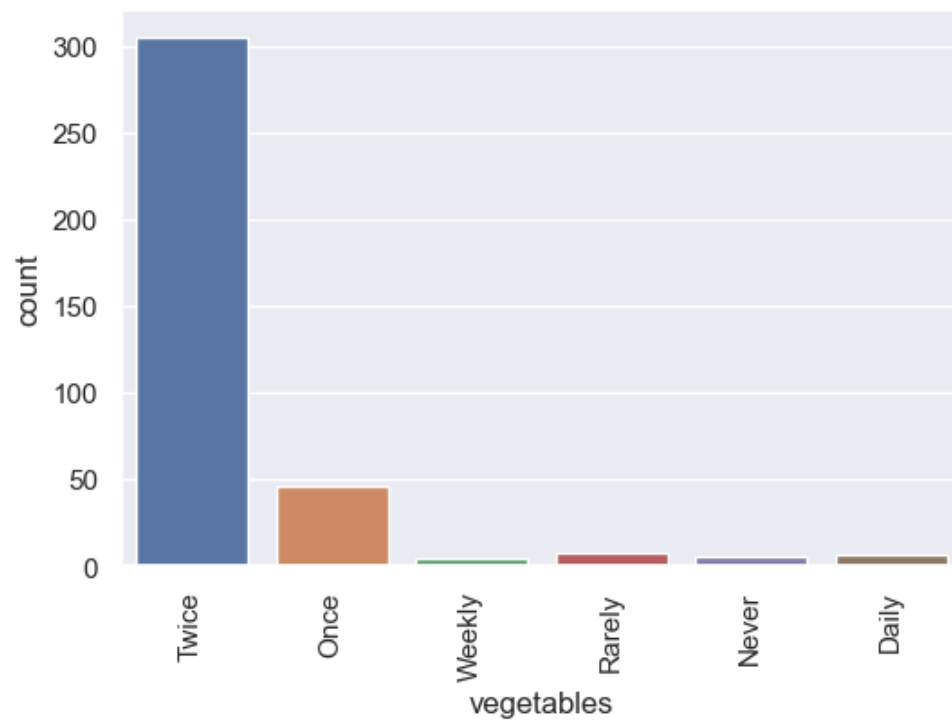


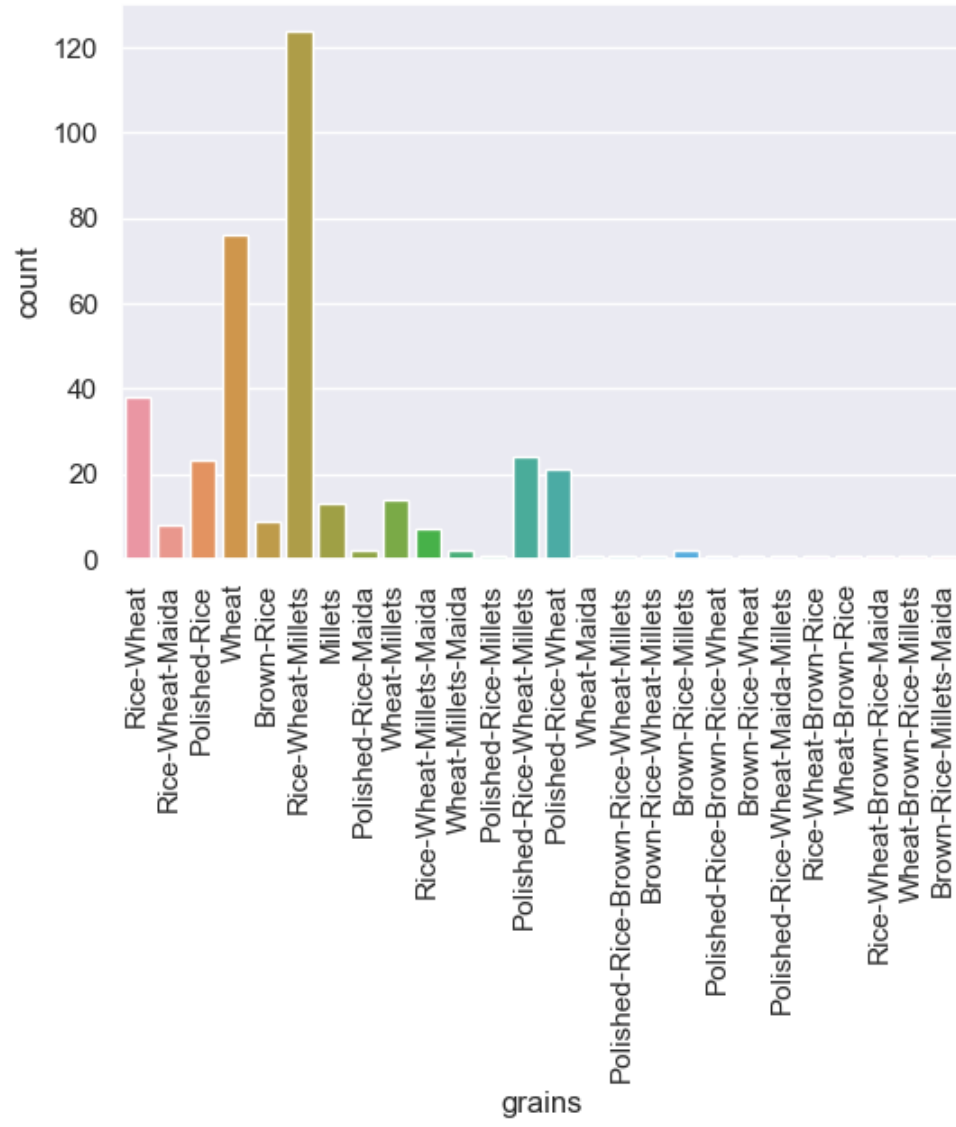


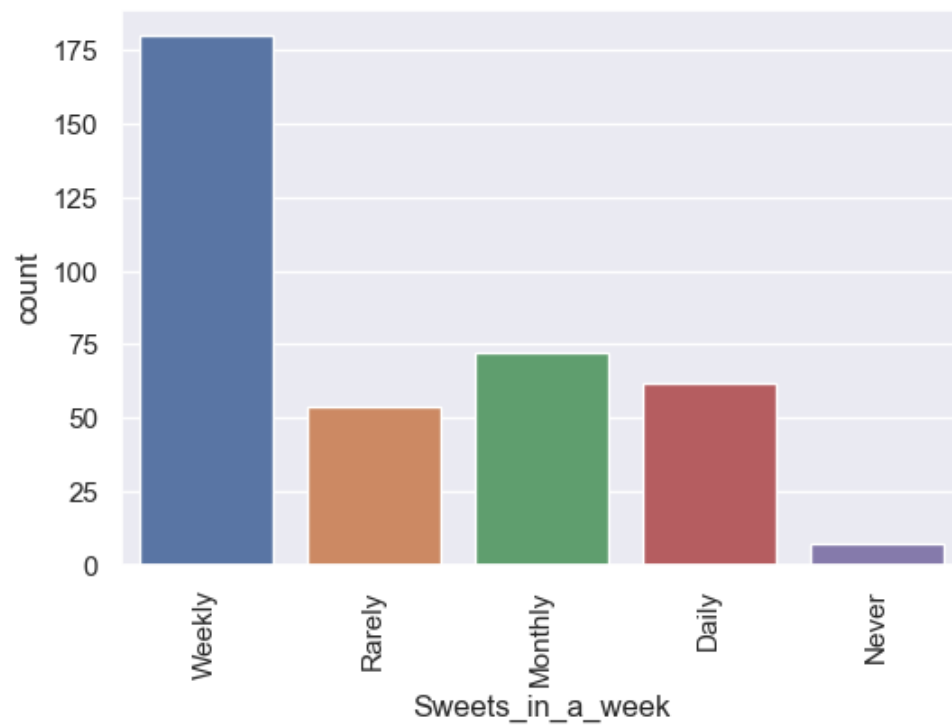
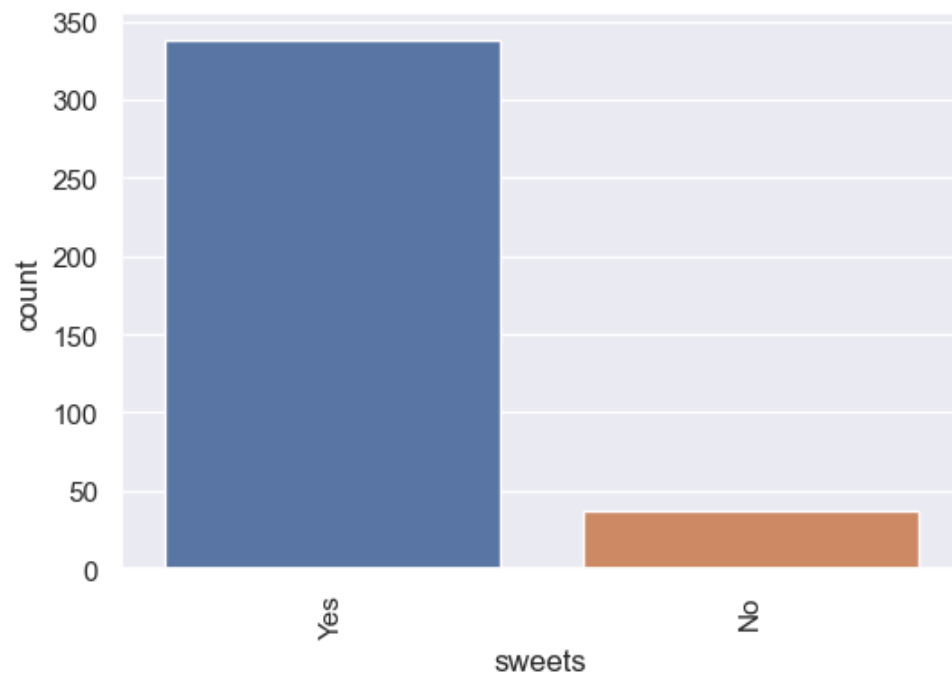


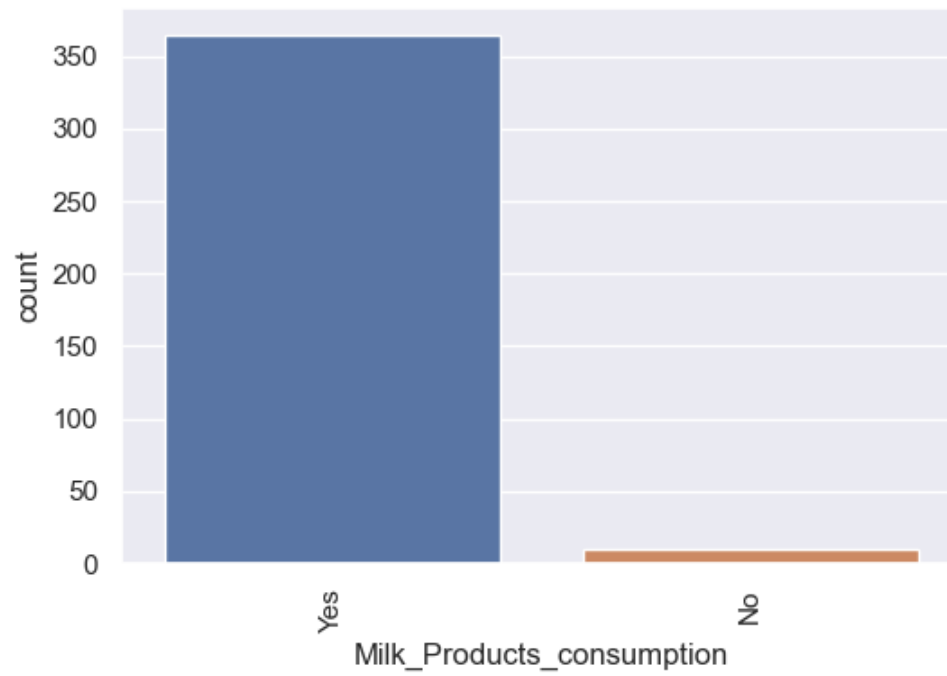
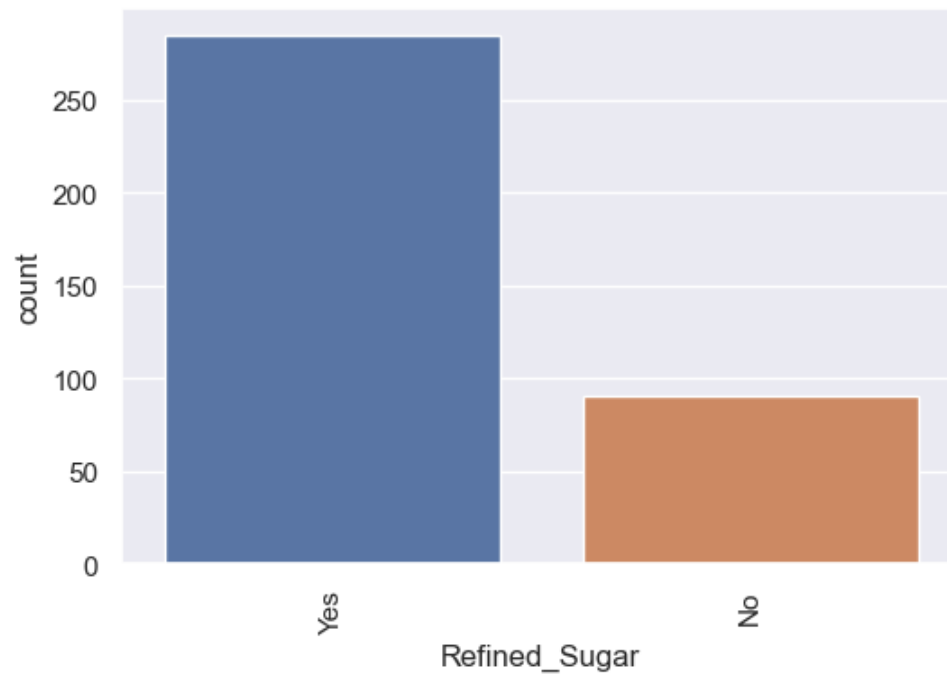


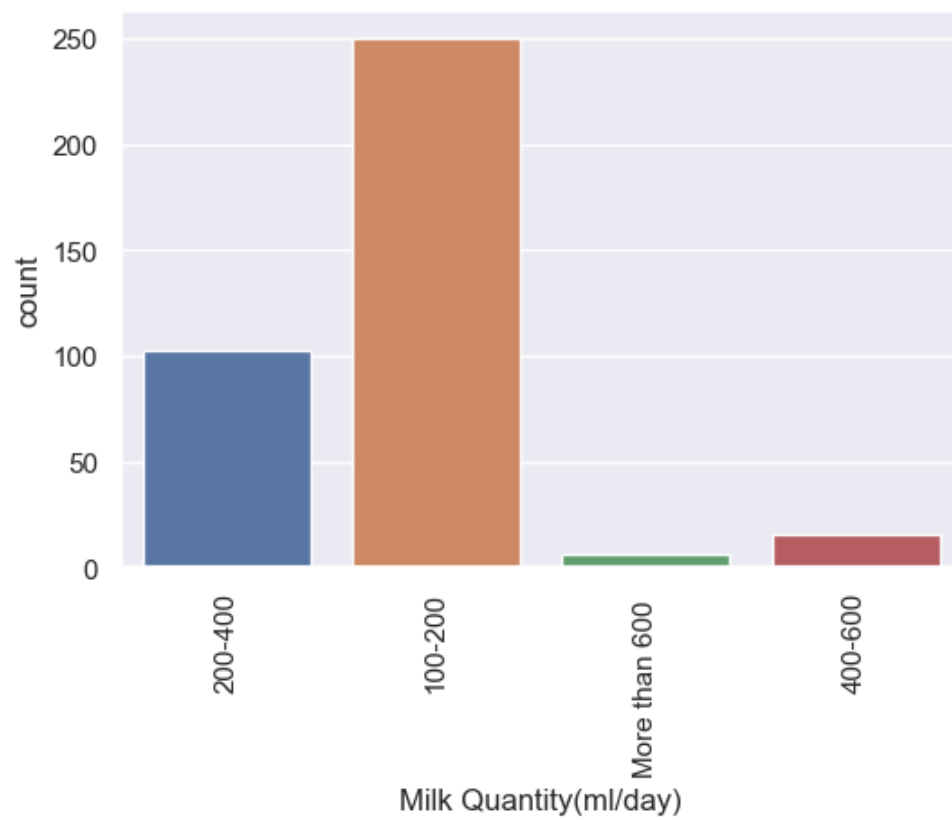
salad

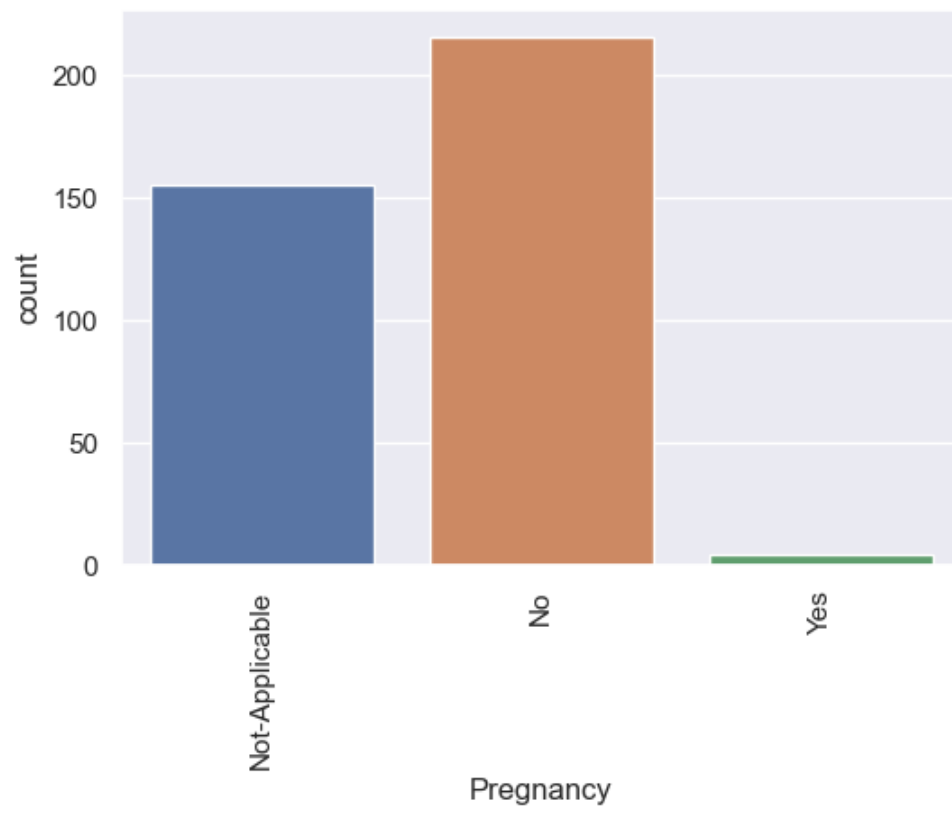


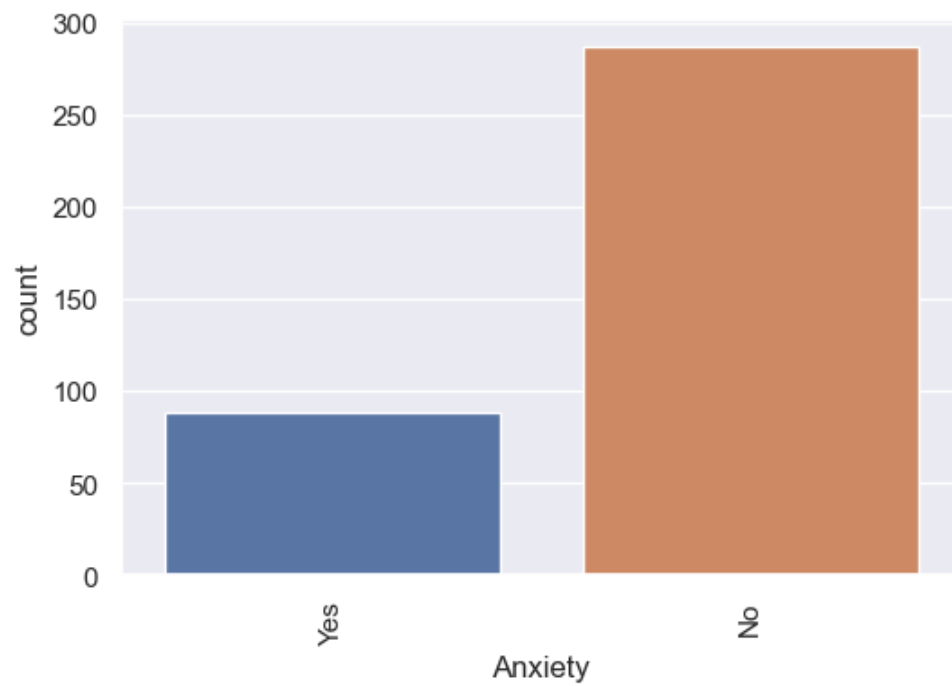
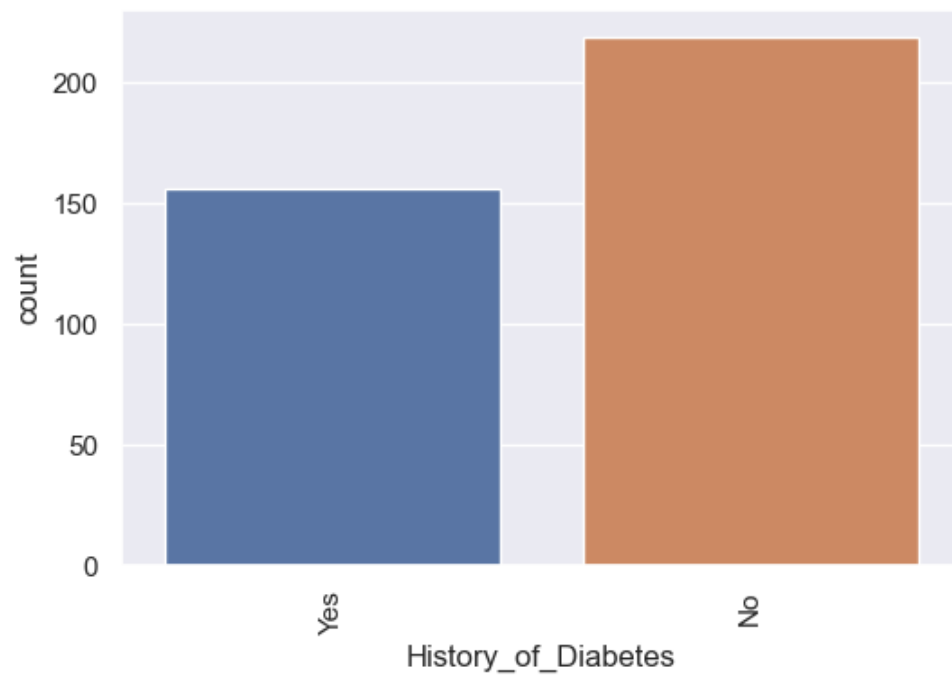


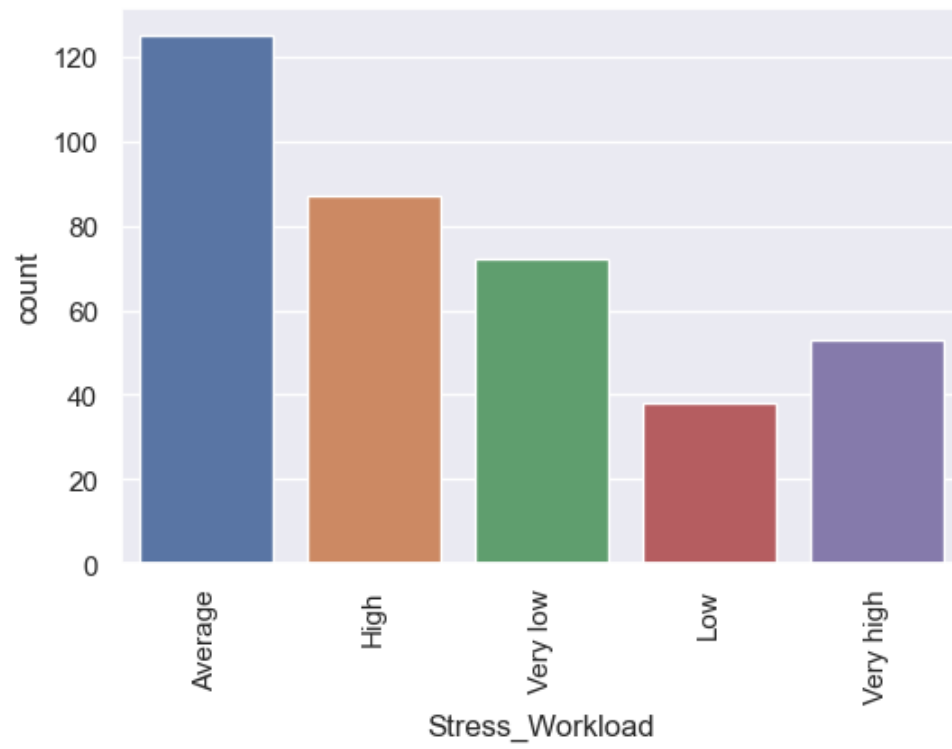


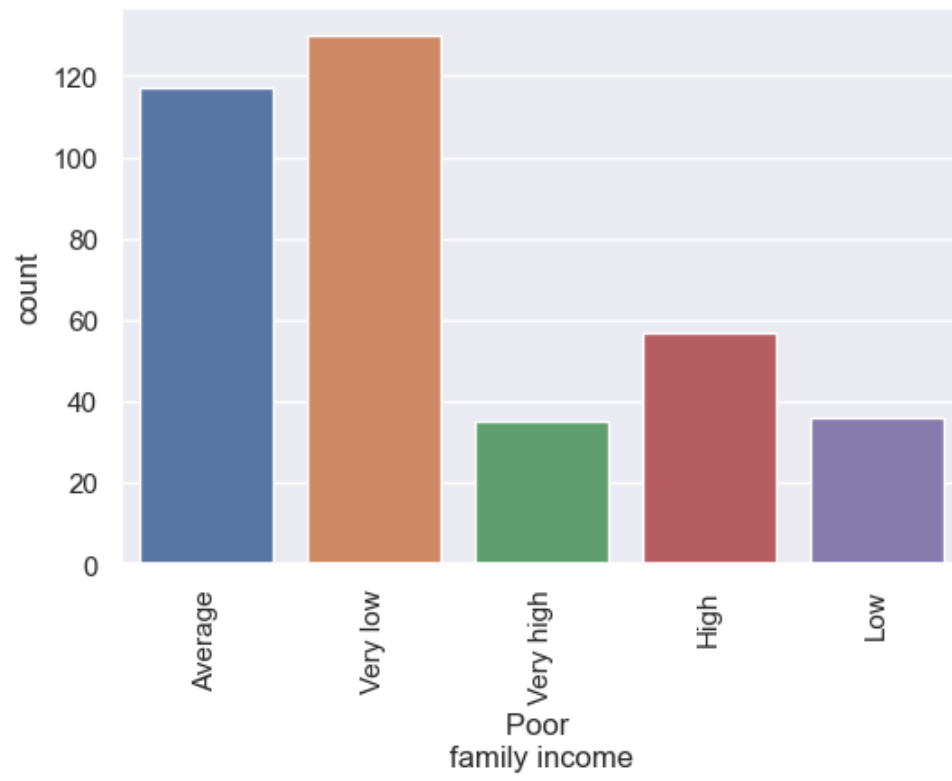


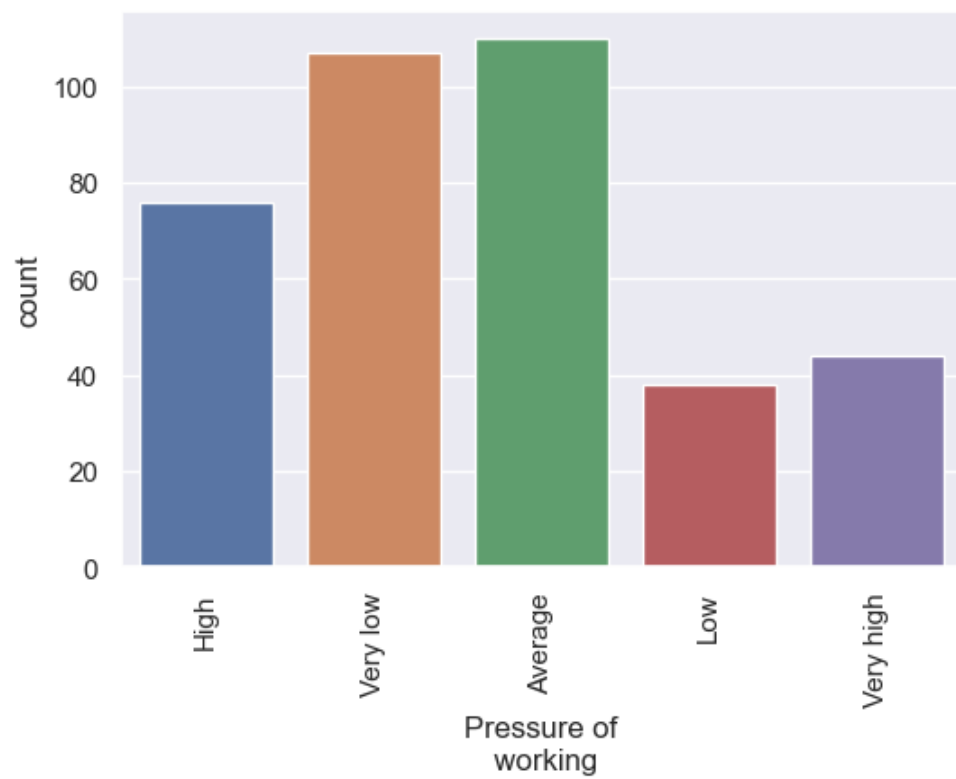


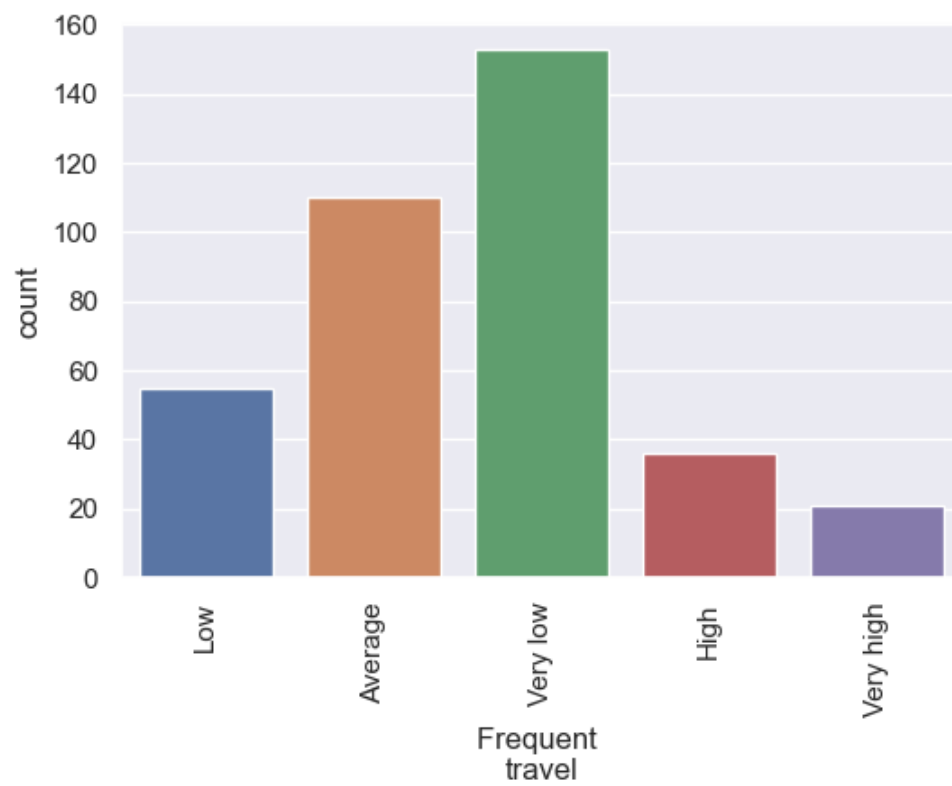


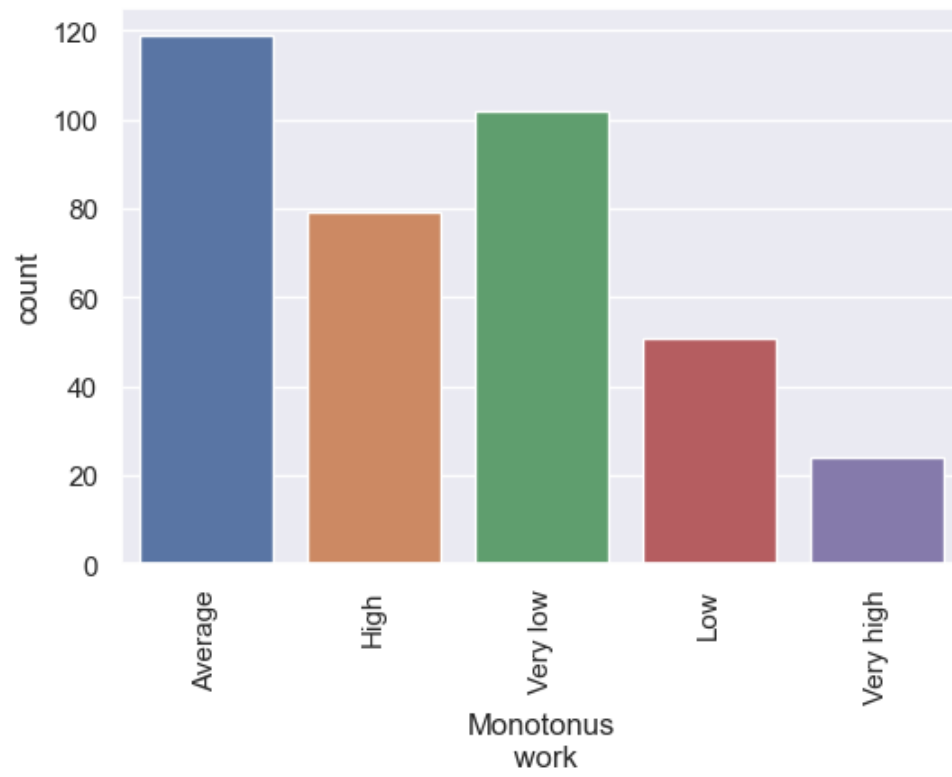


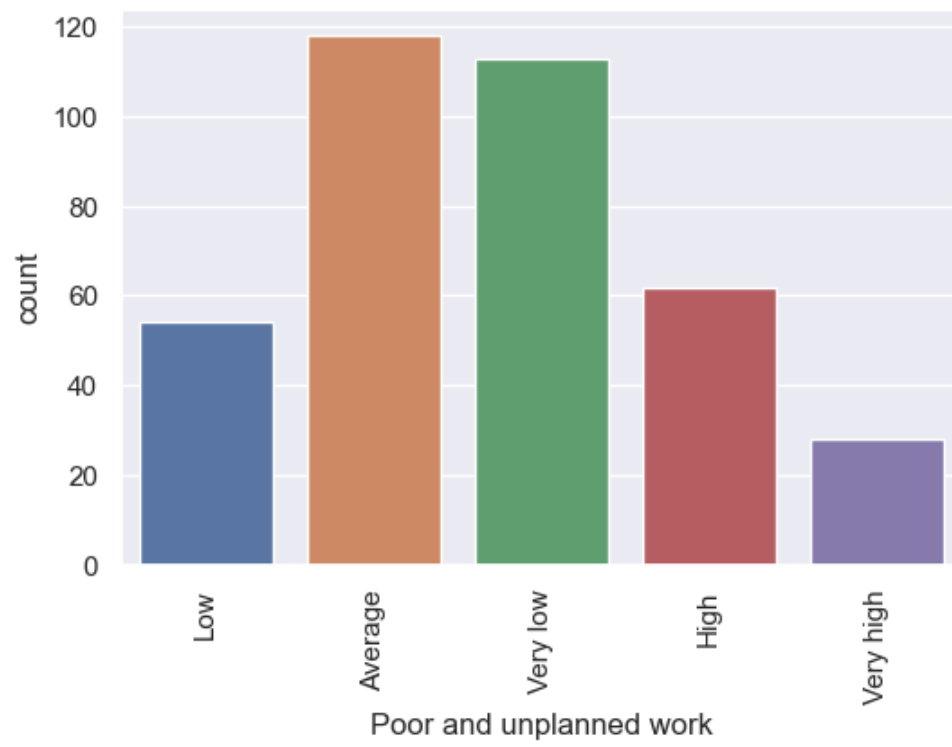


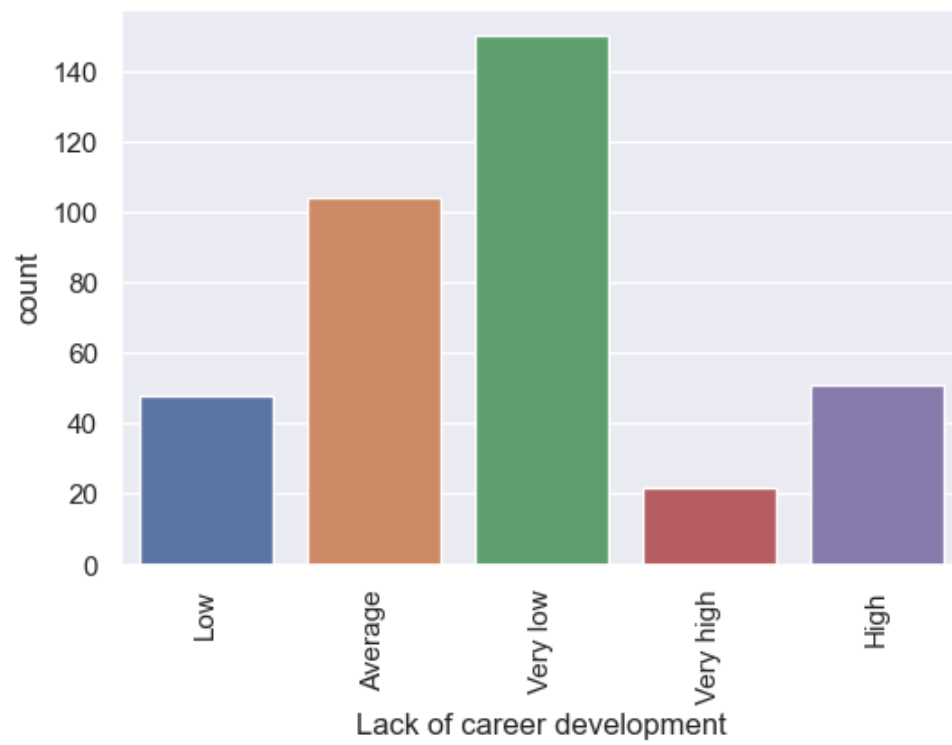


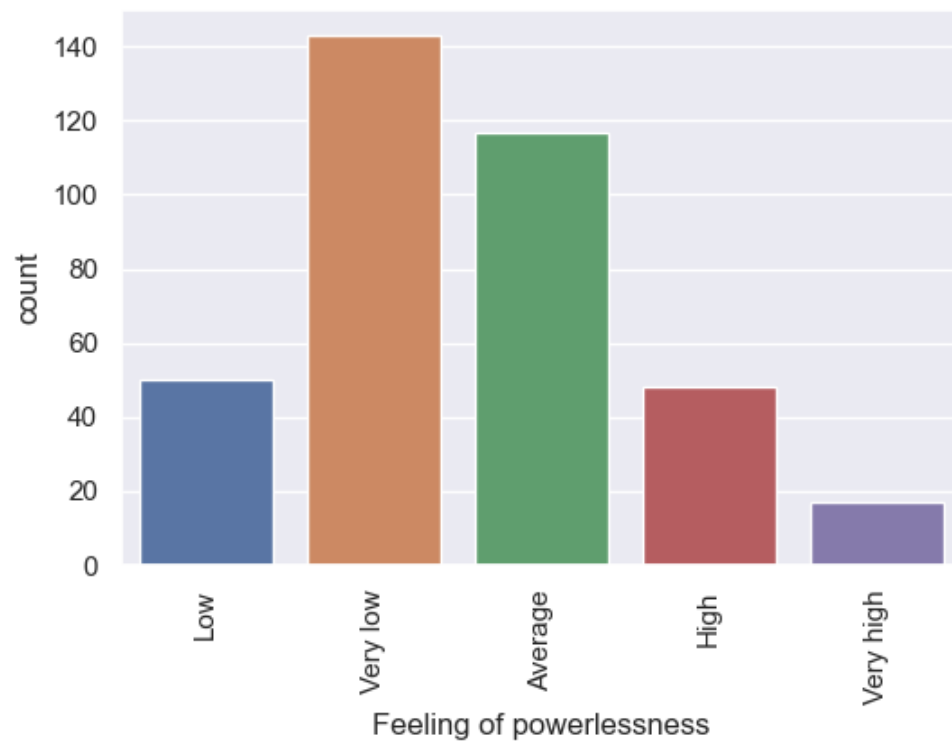


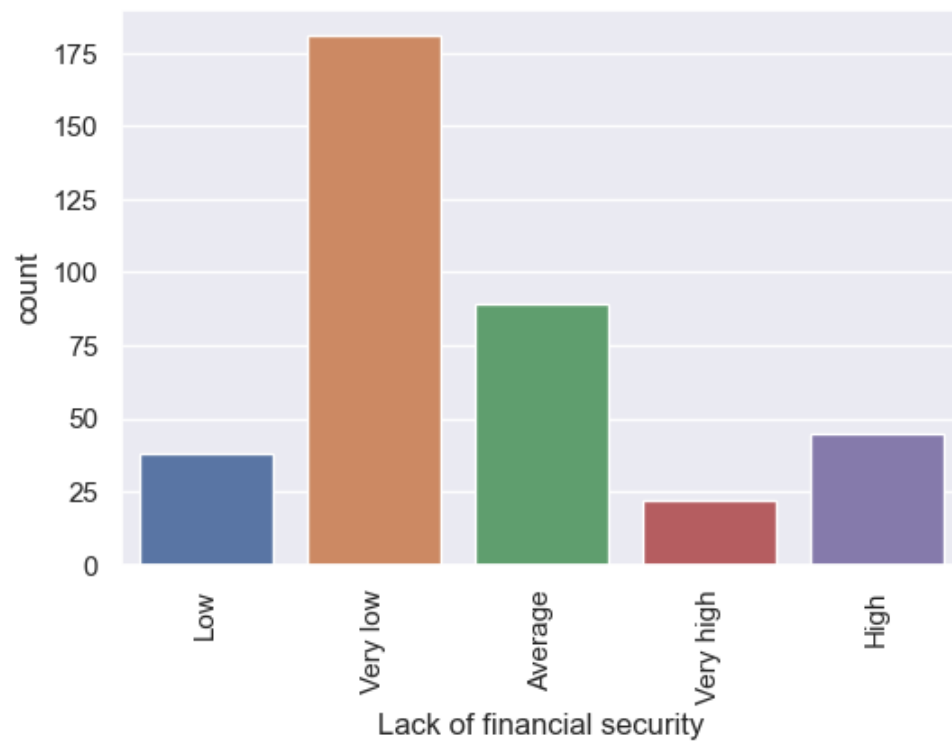


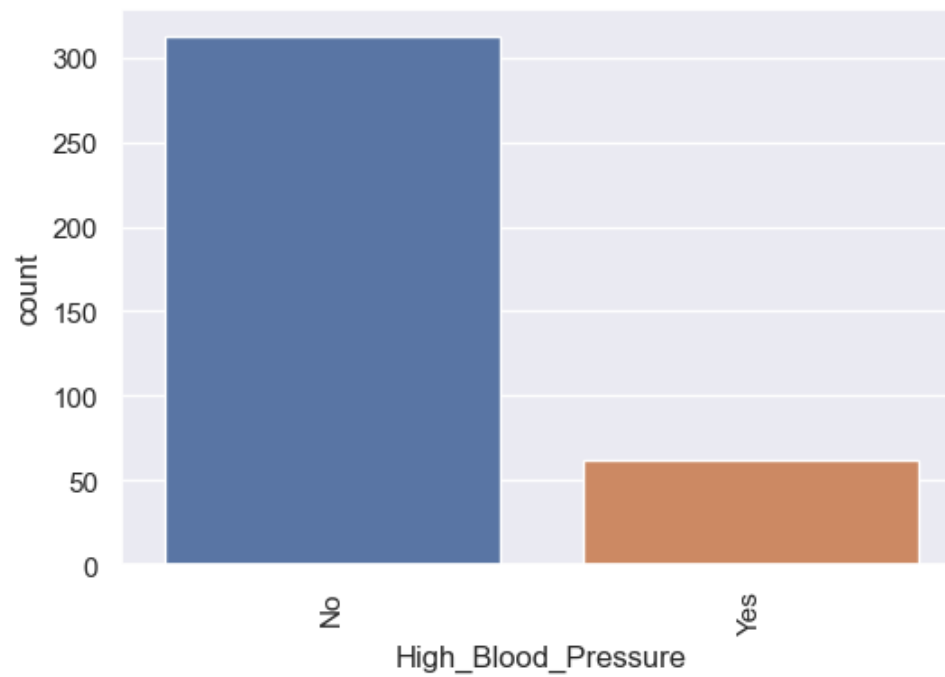
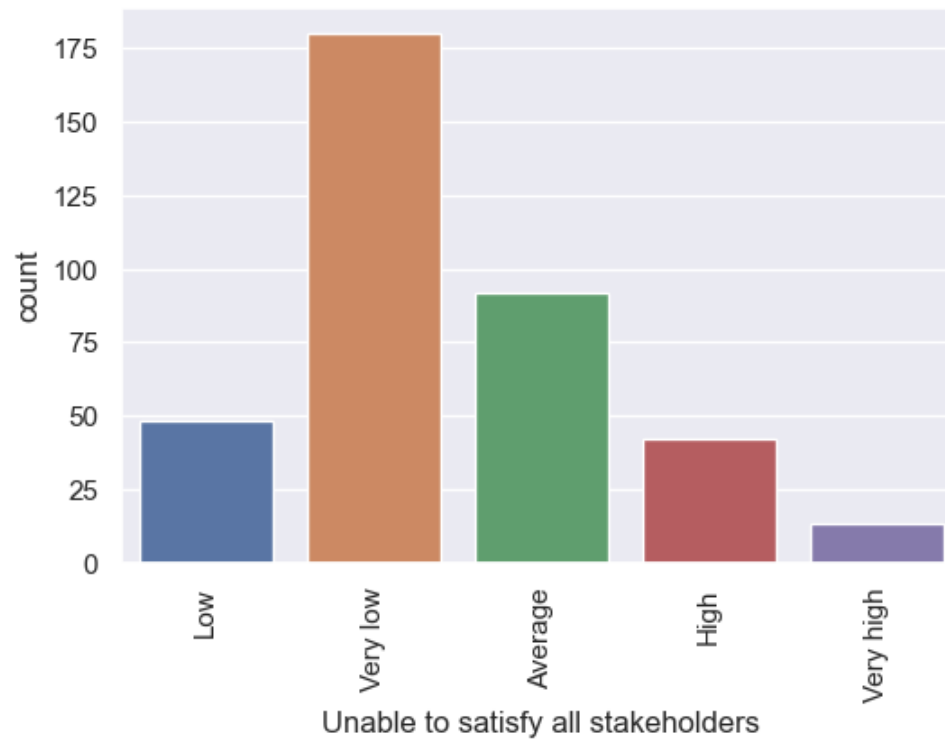


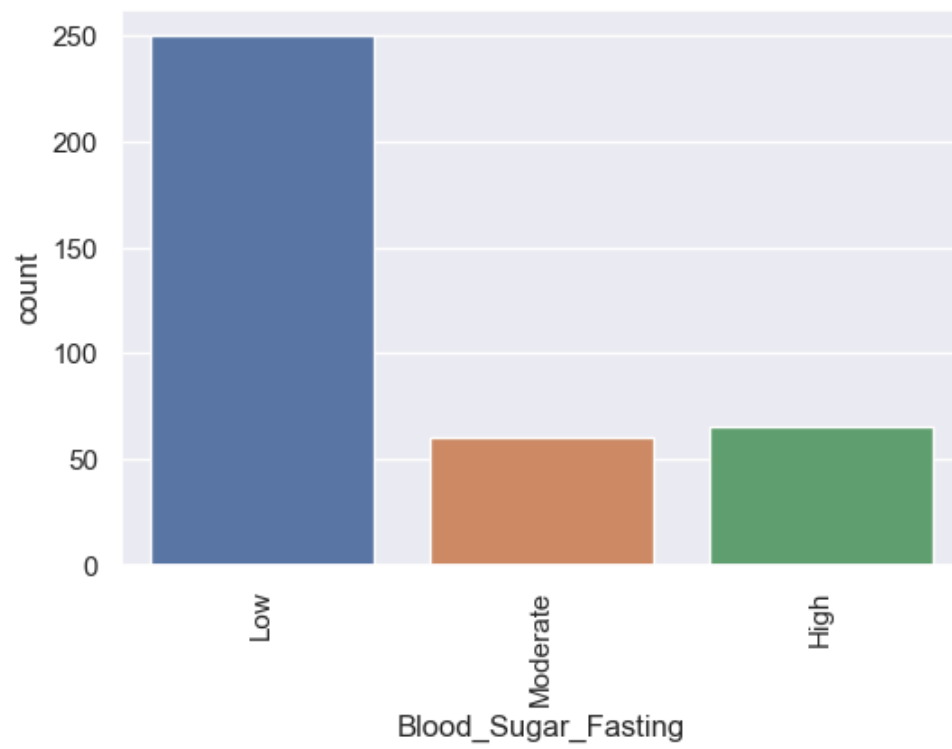


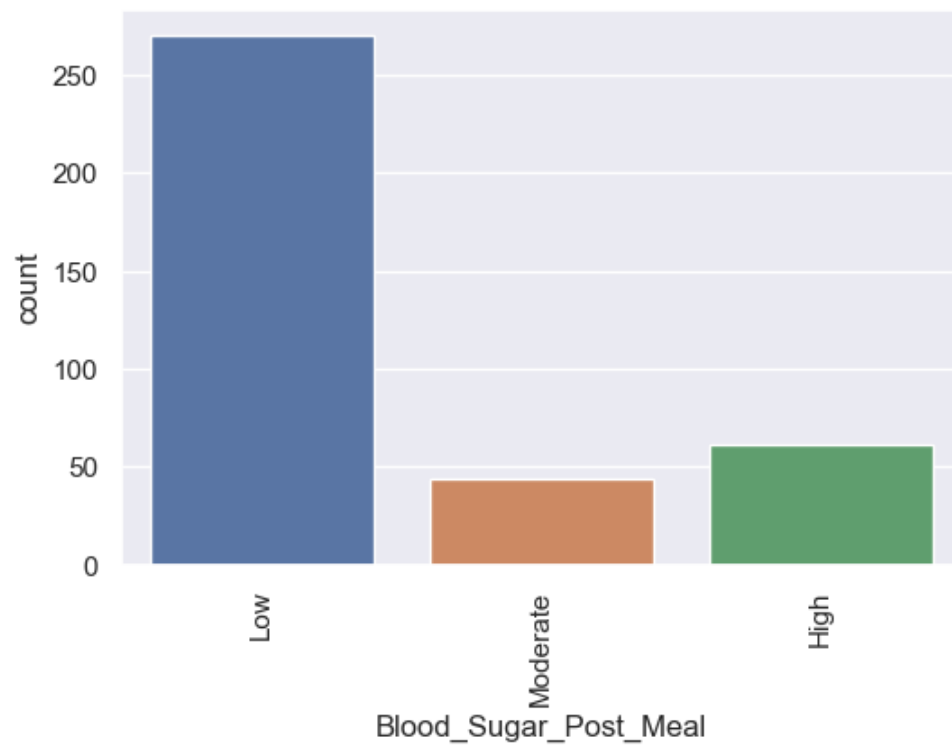


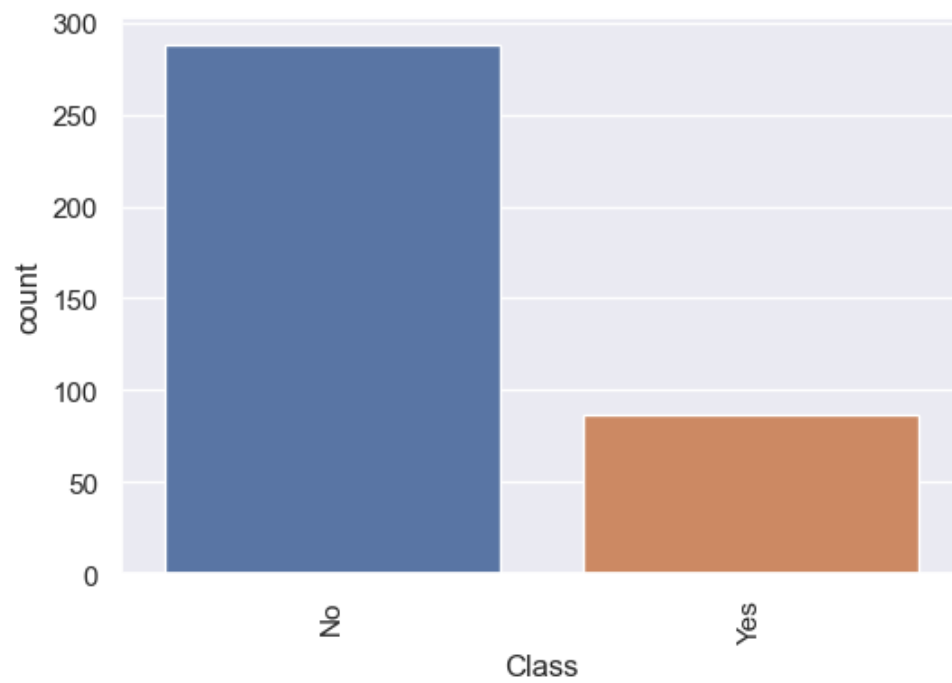
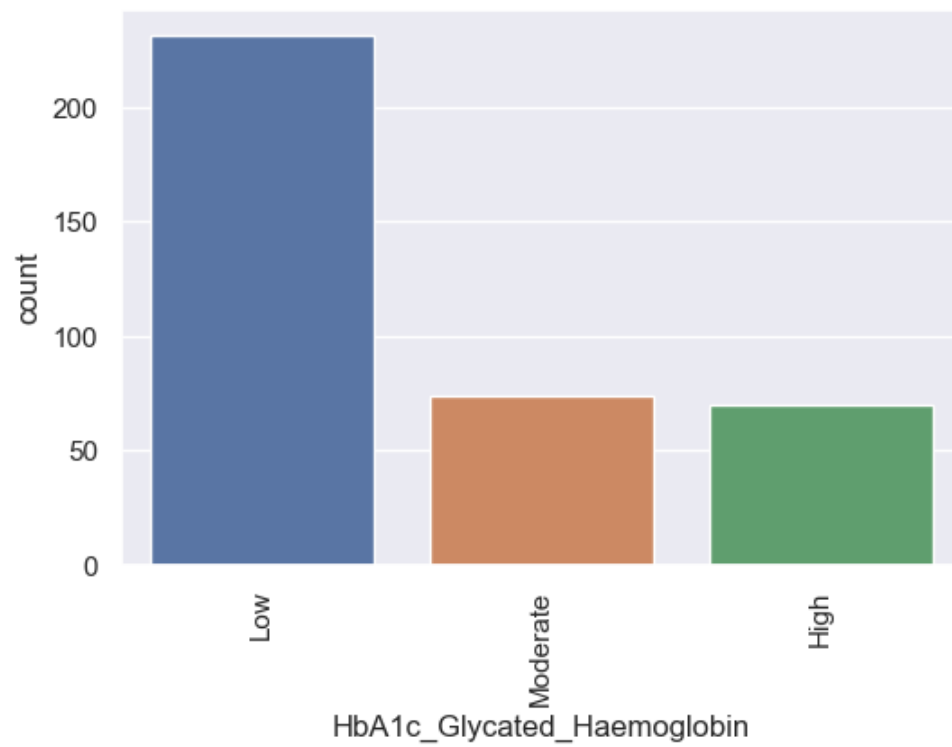




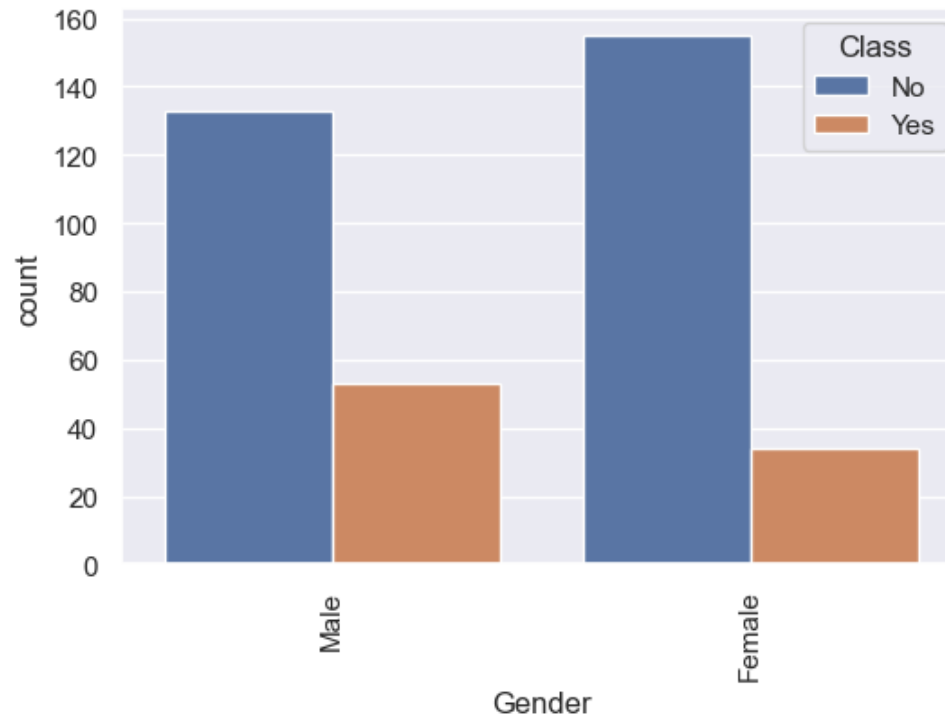


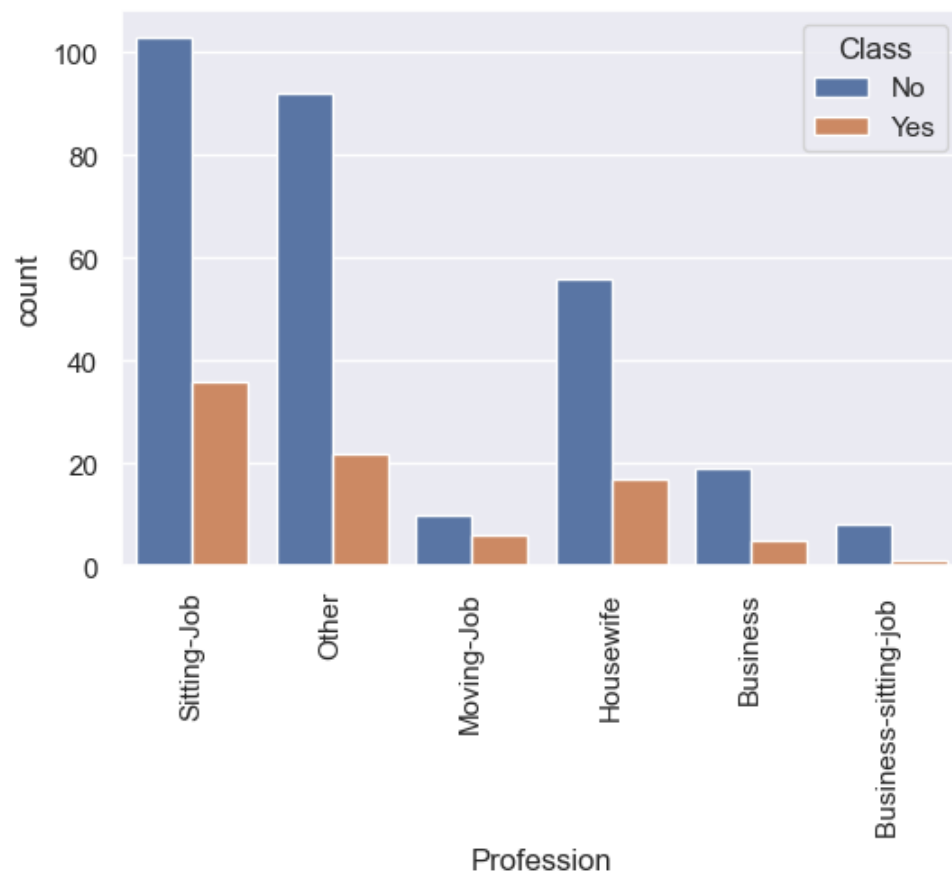


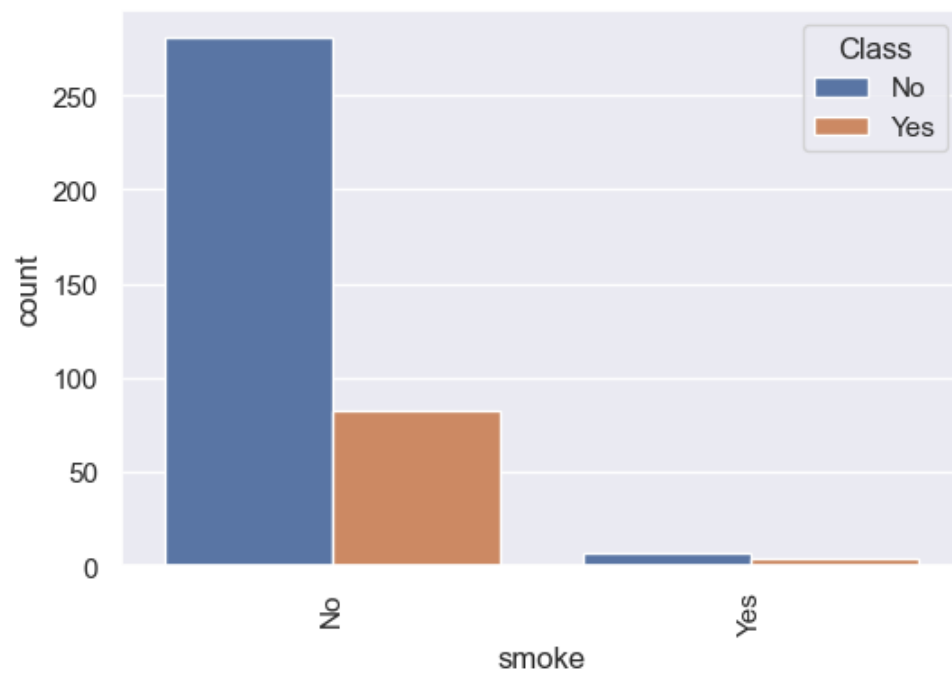


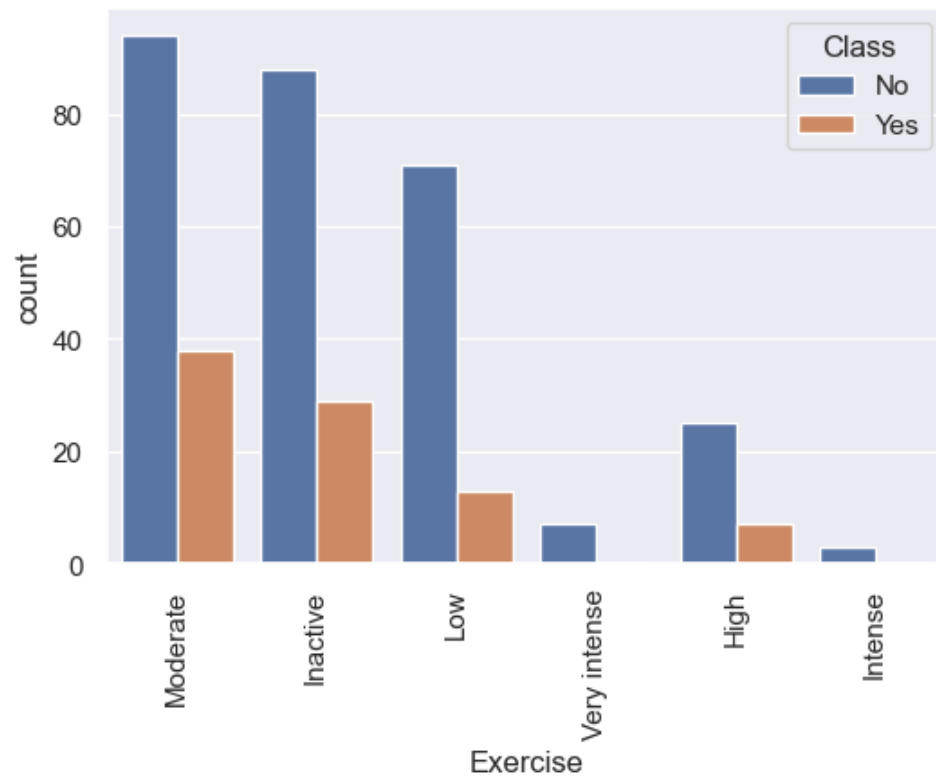


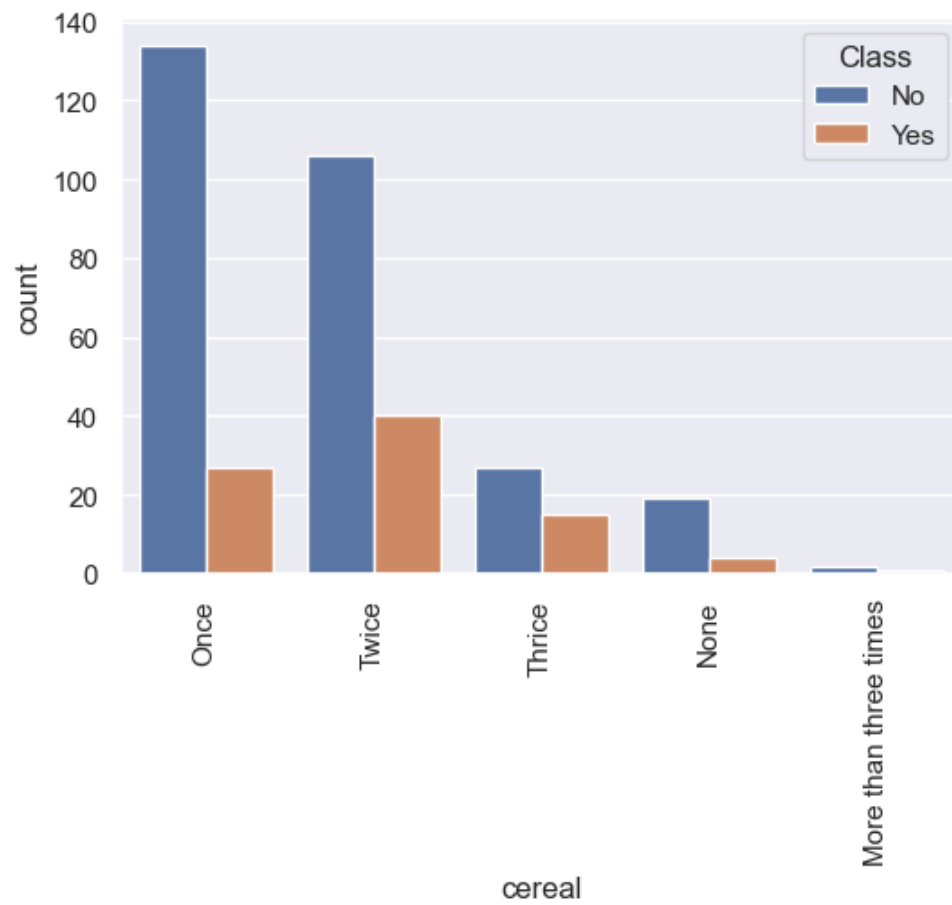

```
In [41]: 1 # Count Plot: Displays the count of occurrences for each category as bars.  
2 for col in categorical_variables_df.columns[:-1]:  
3     plt.xticks(rotation=90)  
4     sns.countplot(data=categorical_variables_df, x=col, hue="Class")  
5     plt.show()
```

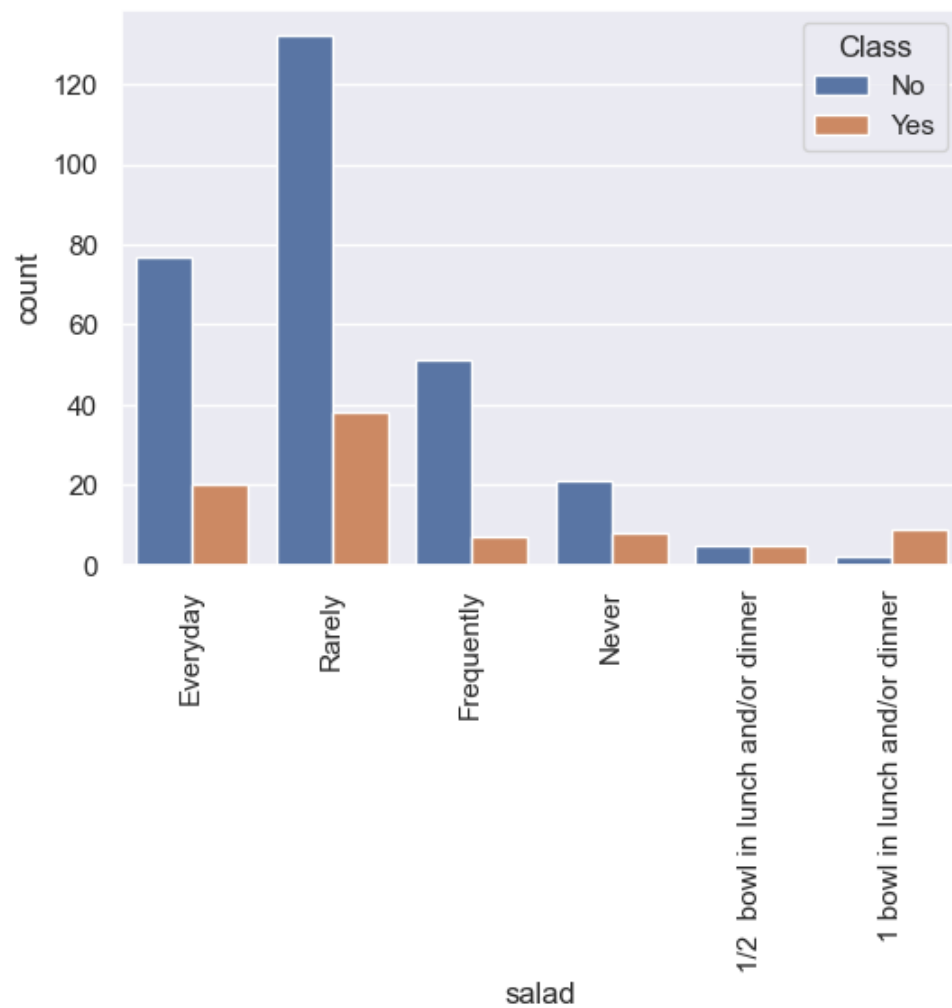


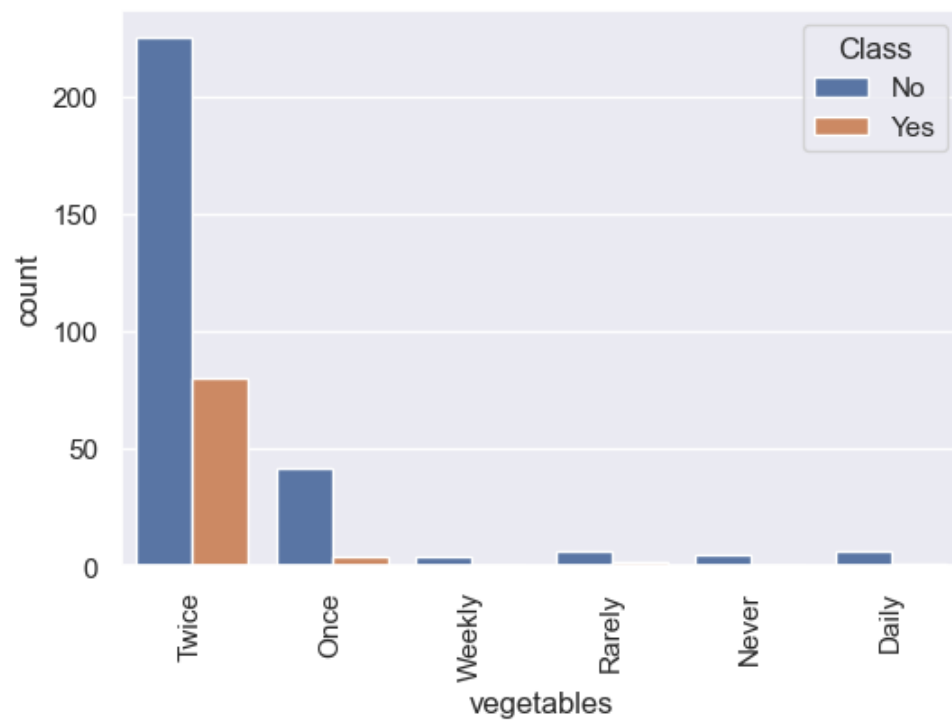


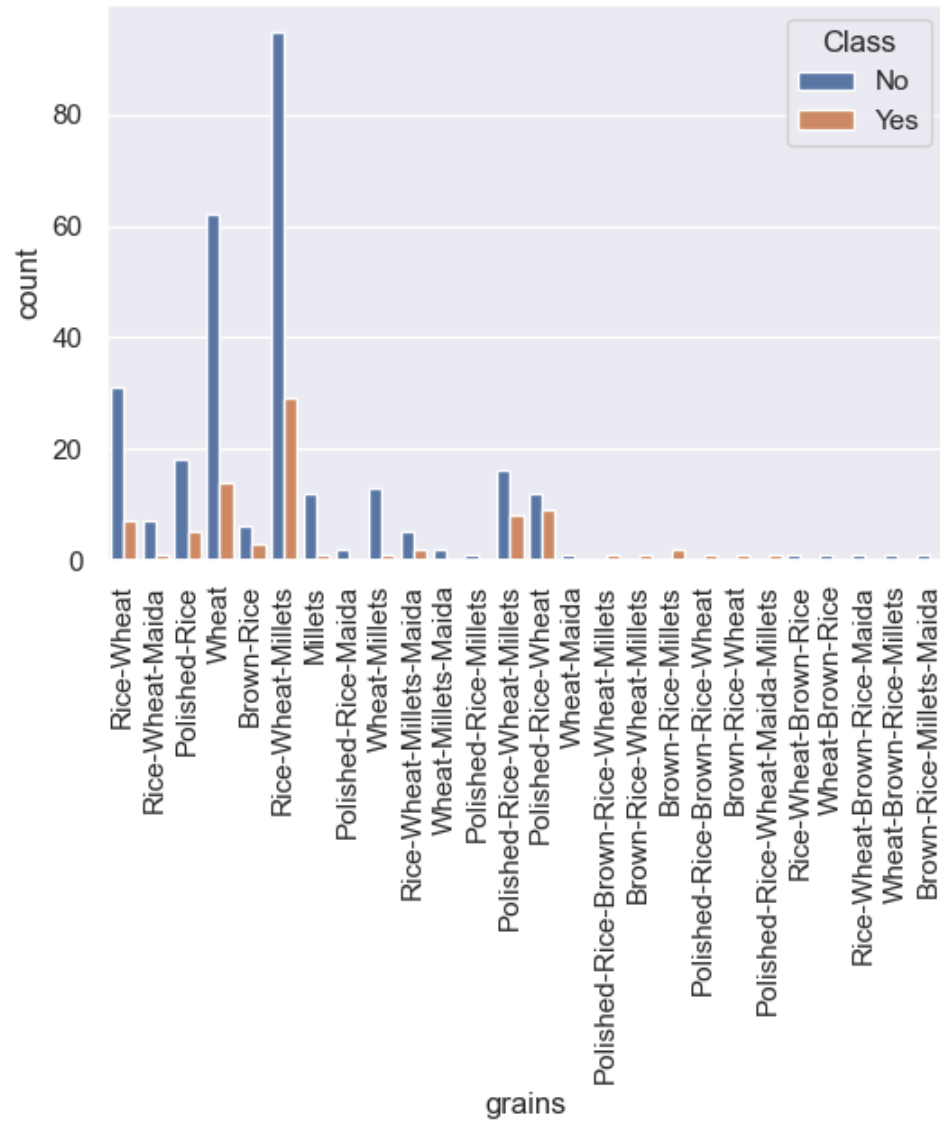


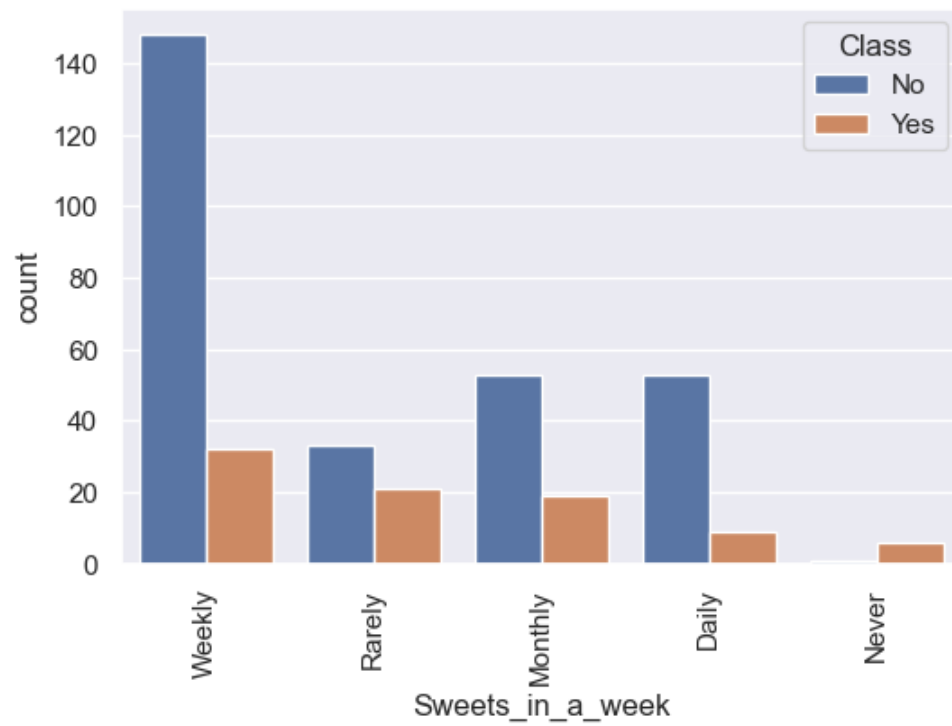
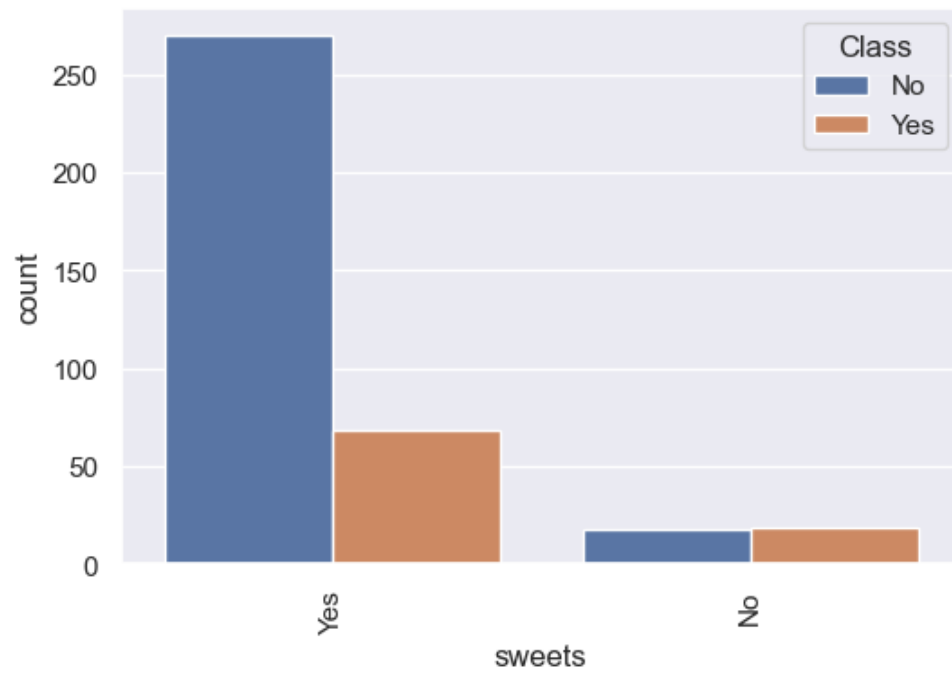


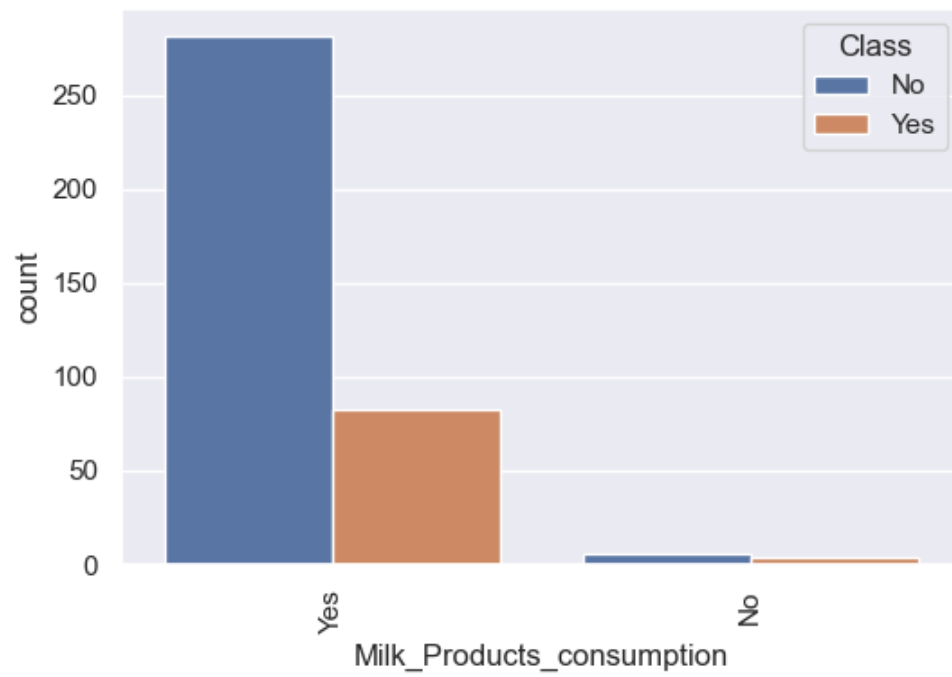
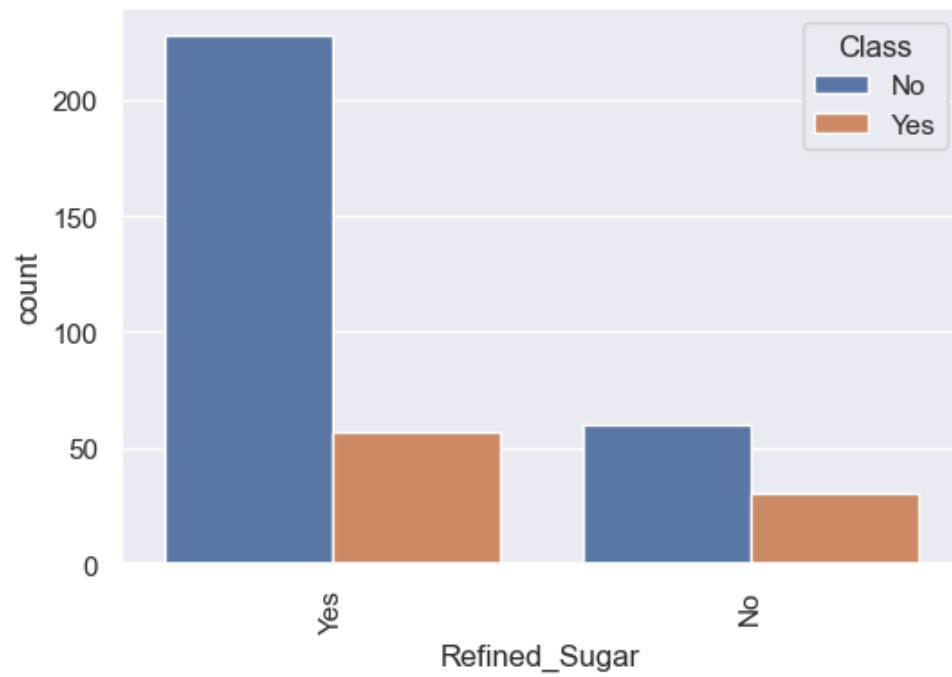


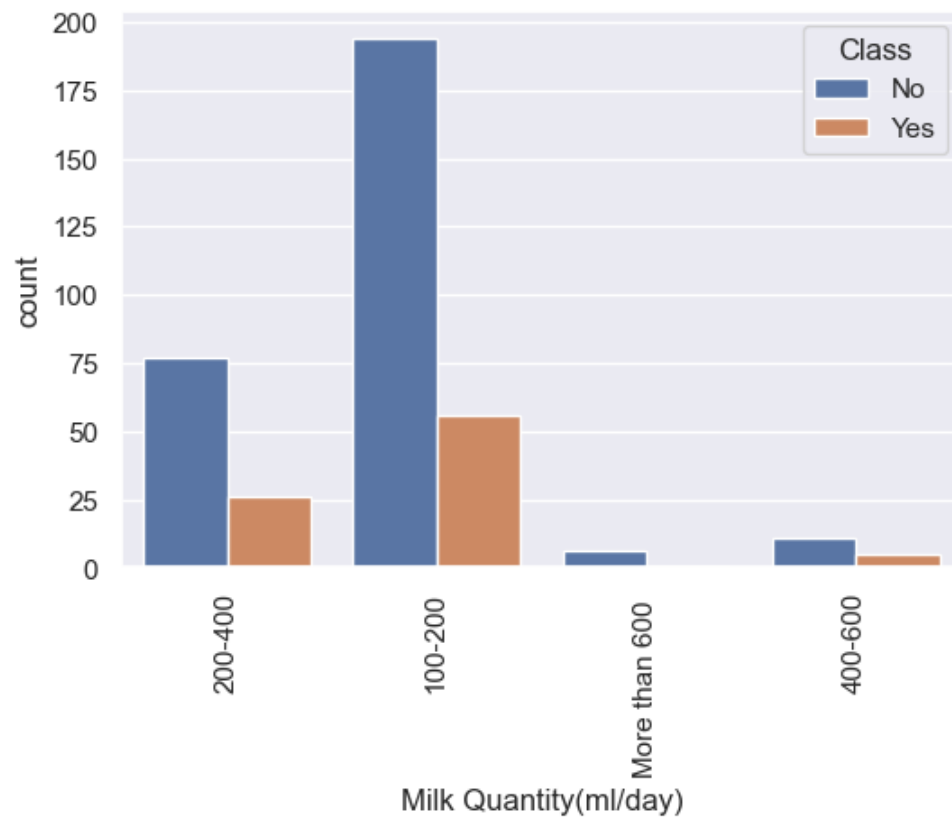


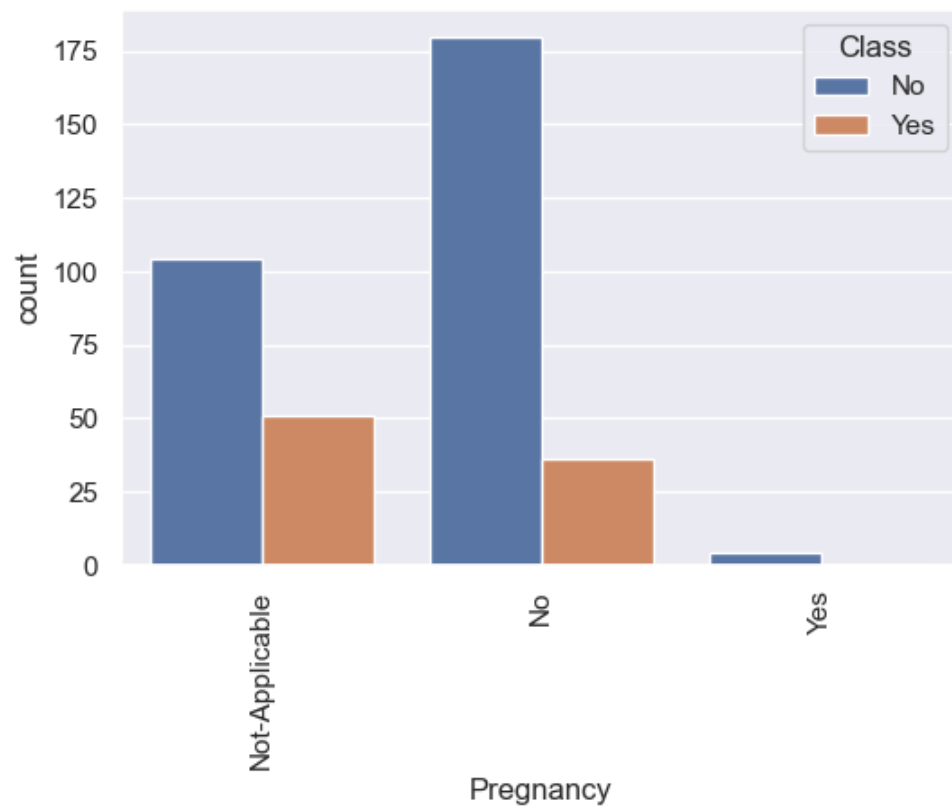


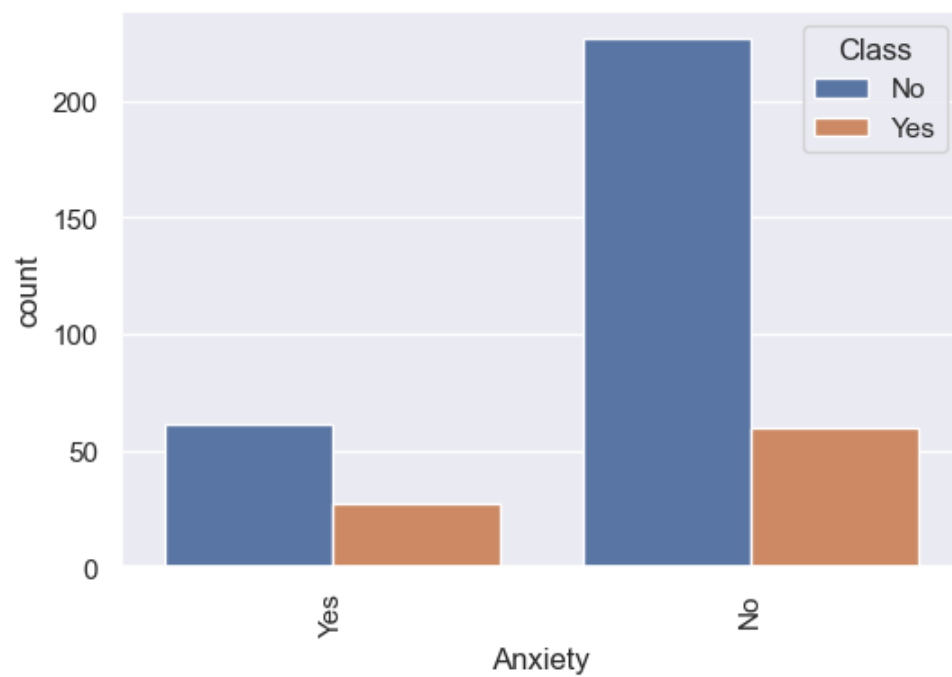
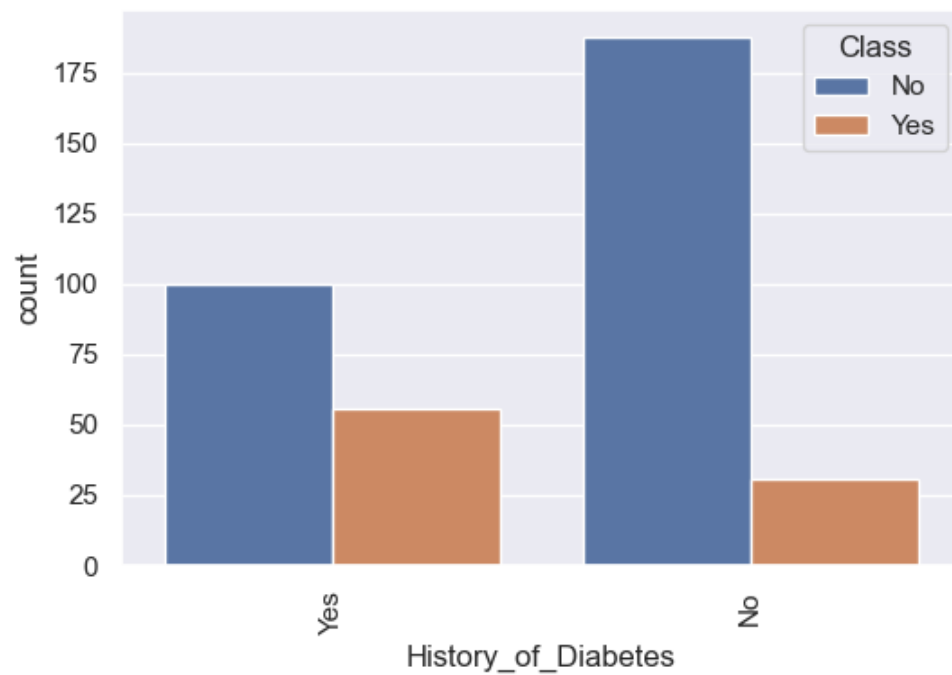


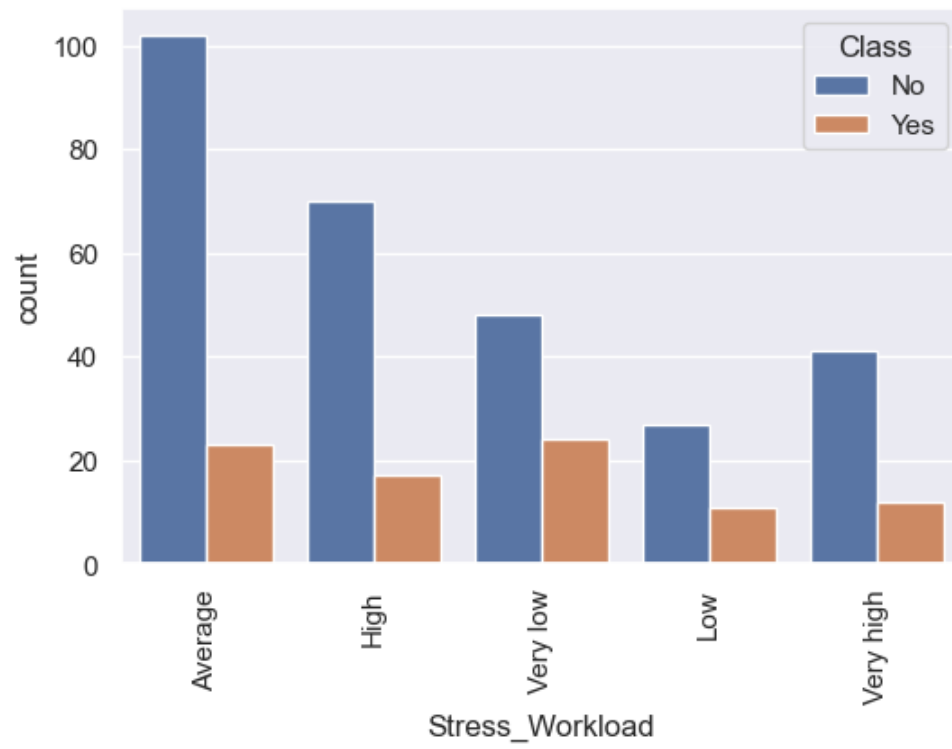


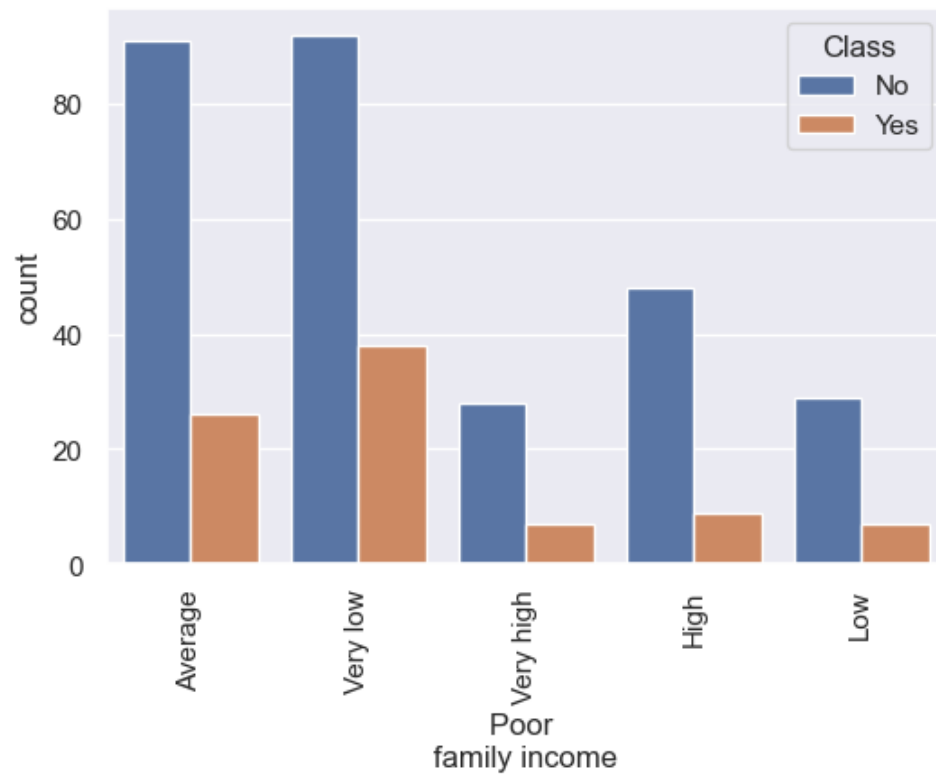


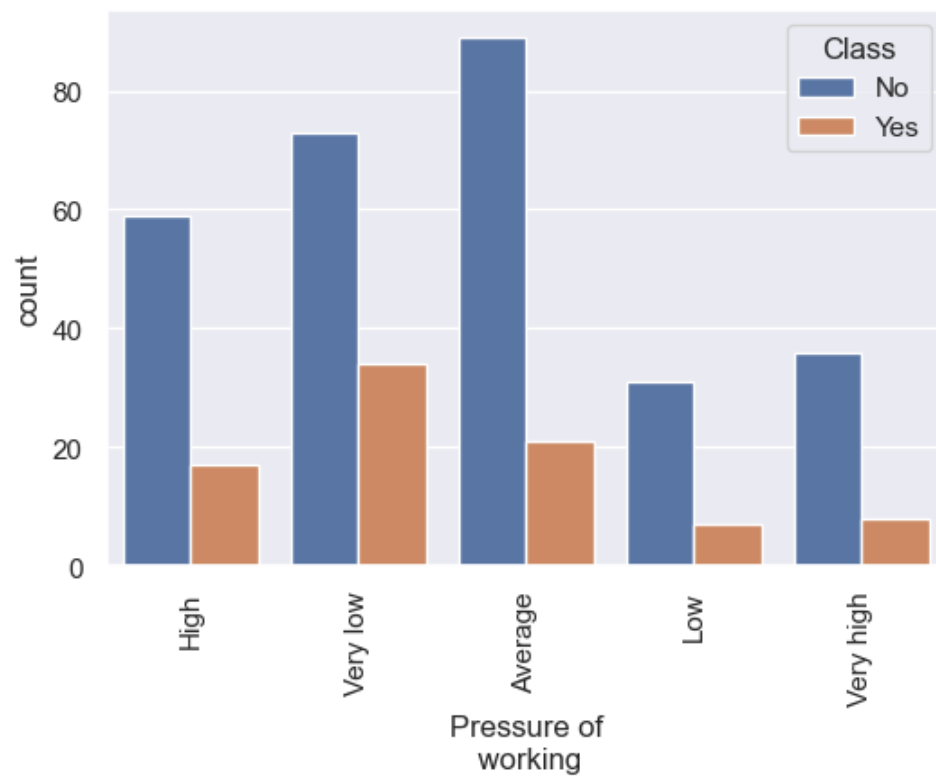


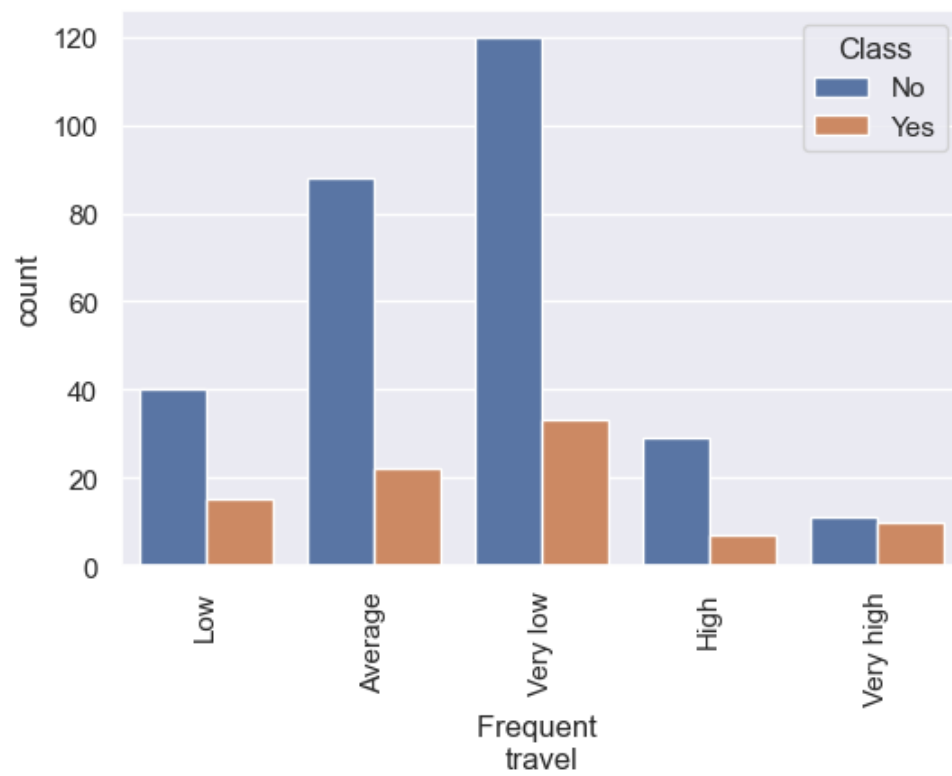


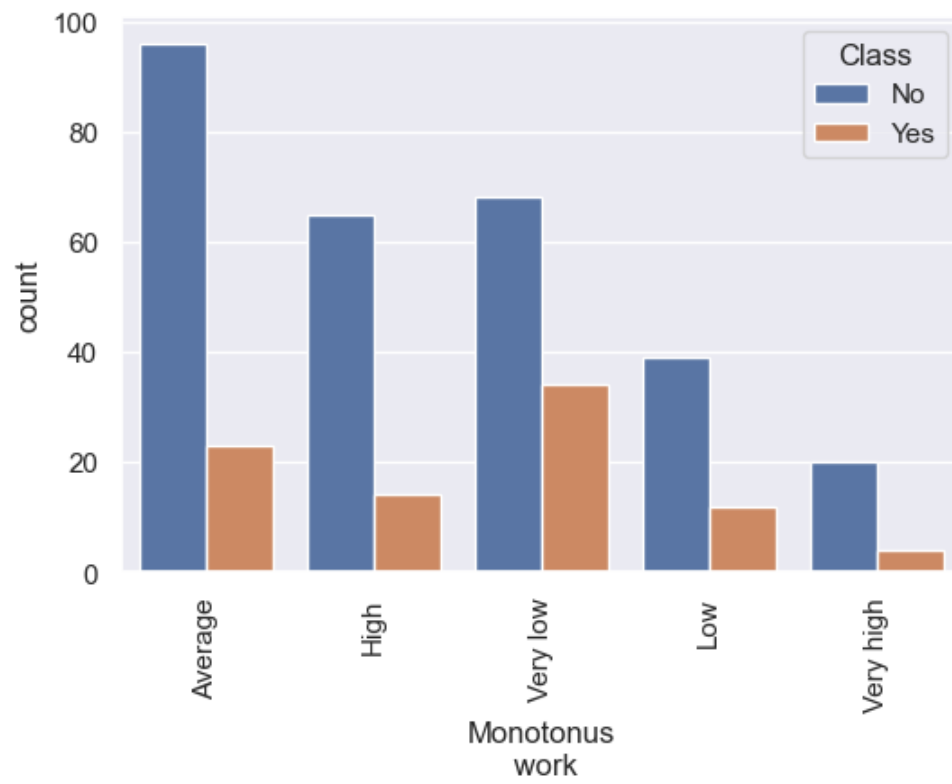


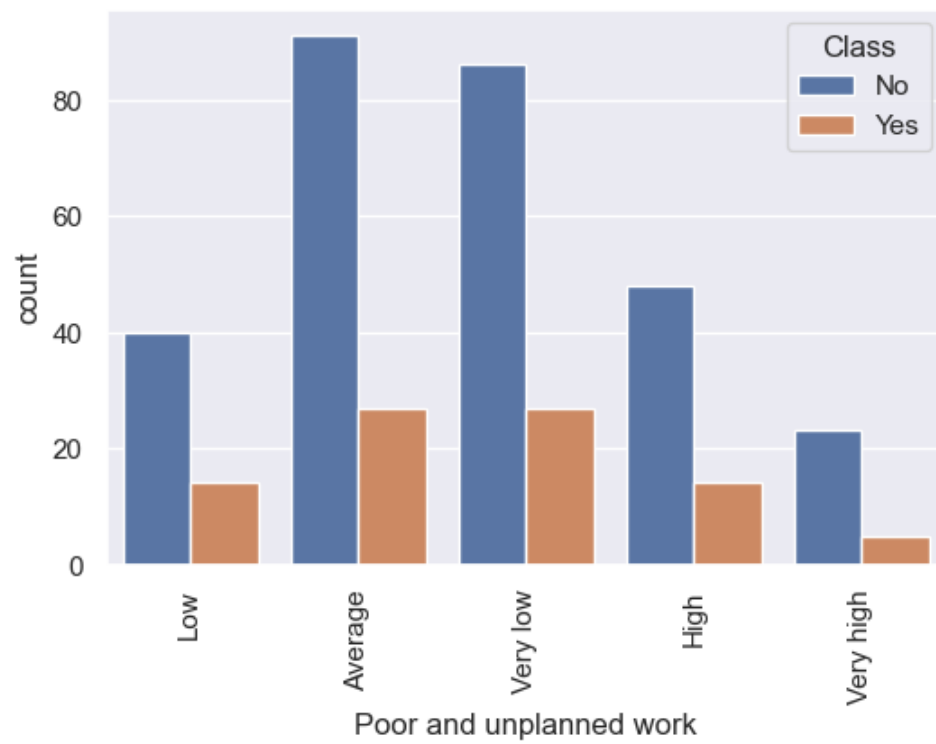


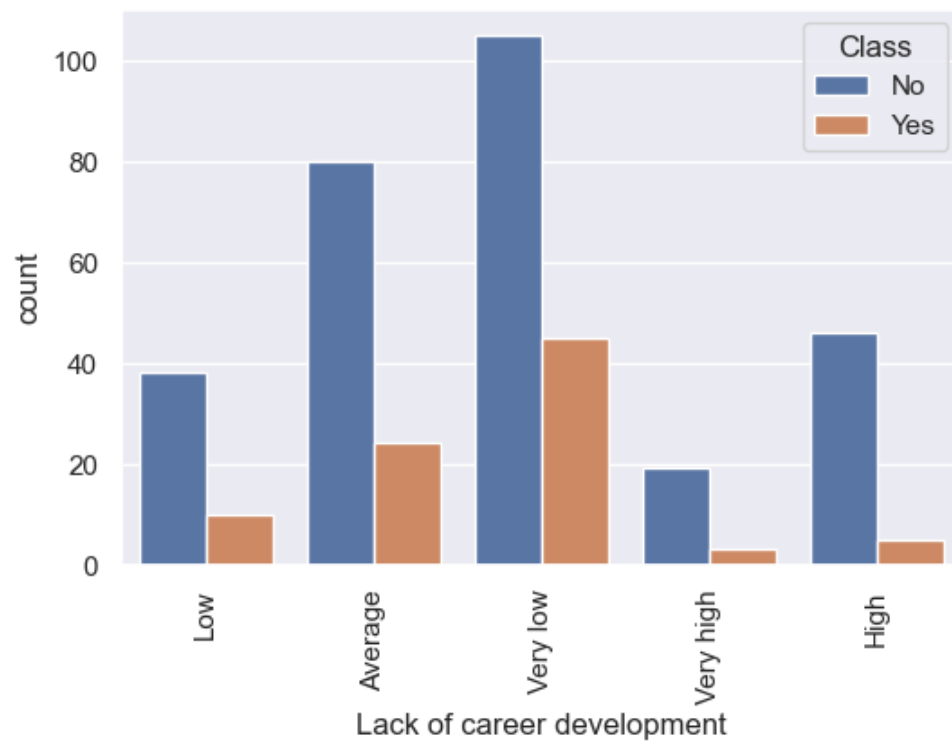


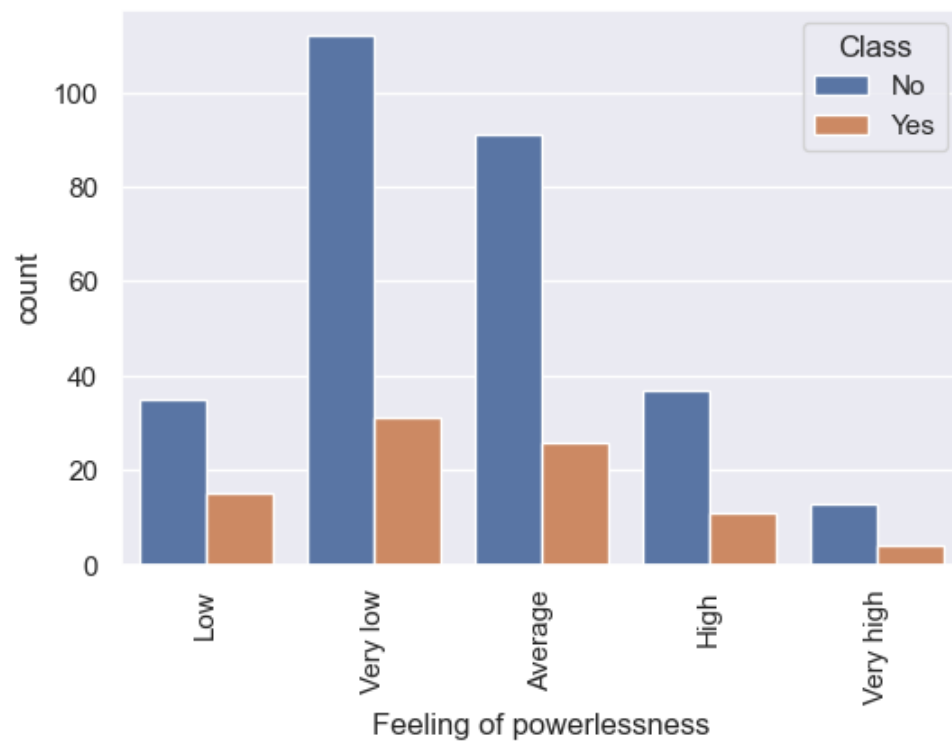


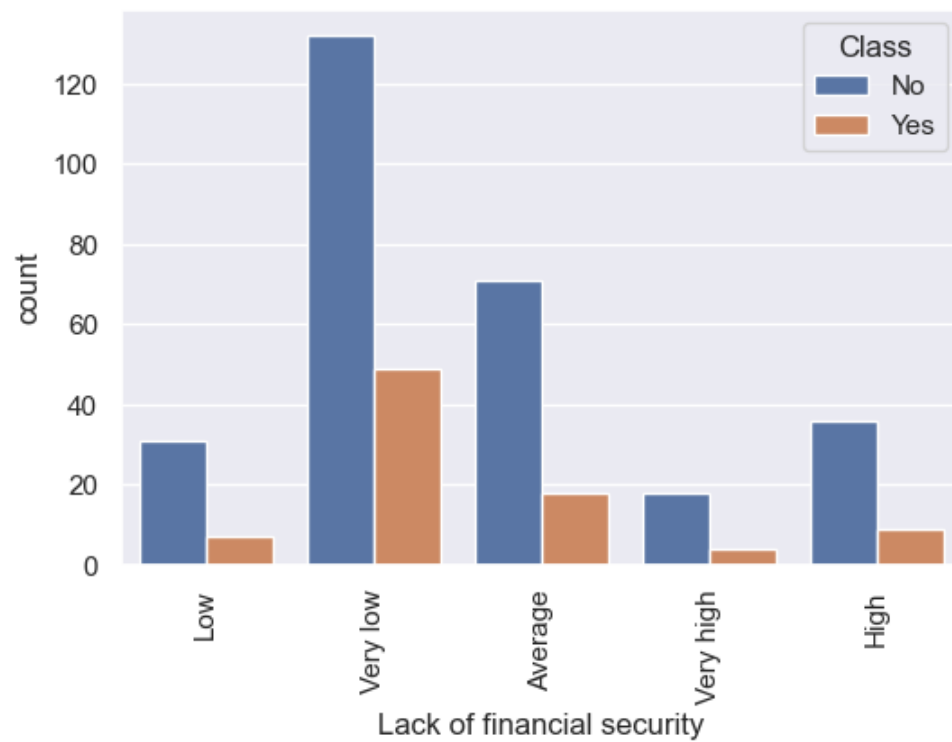


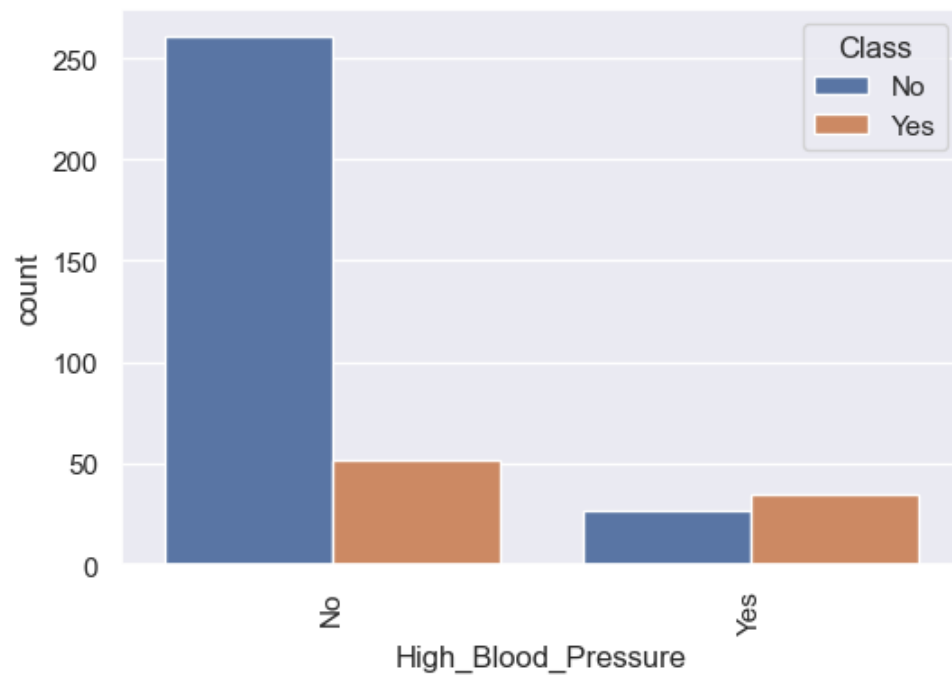
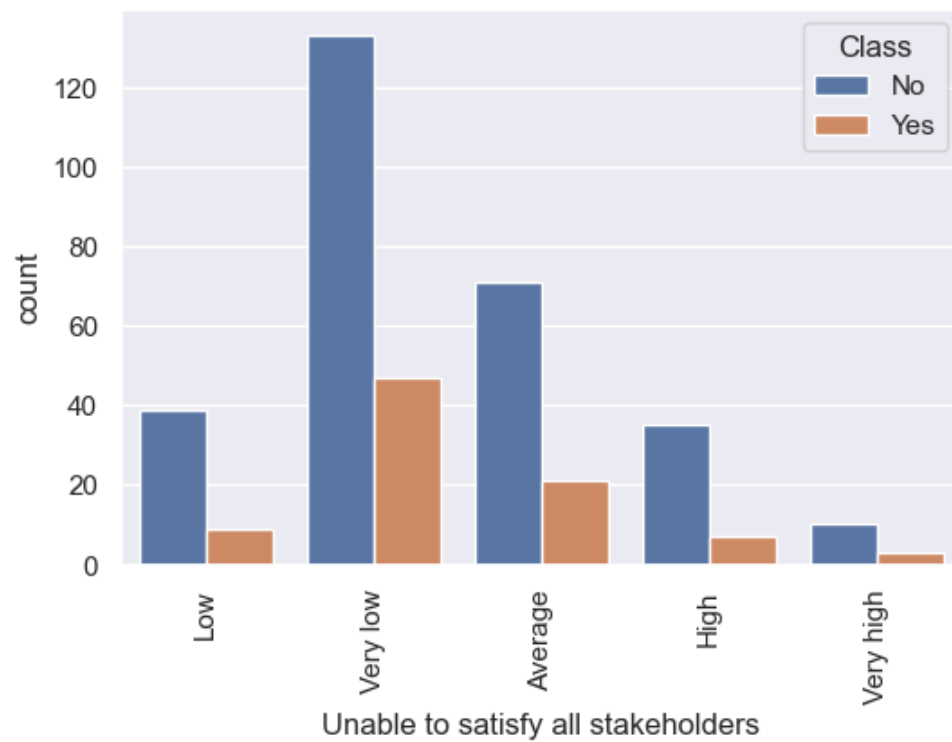


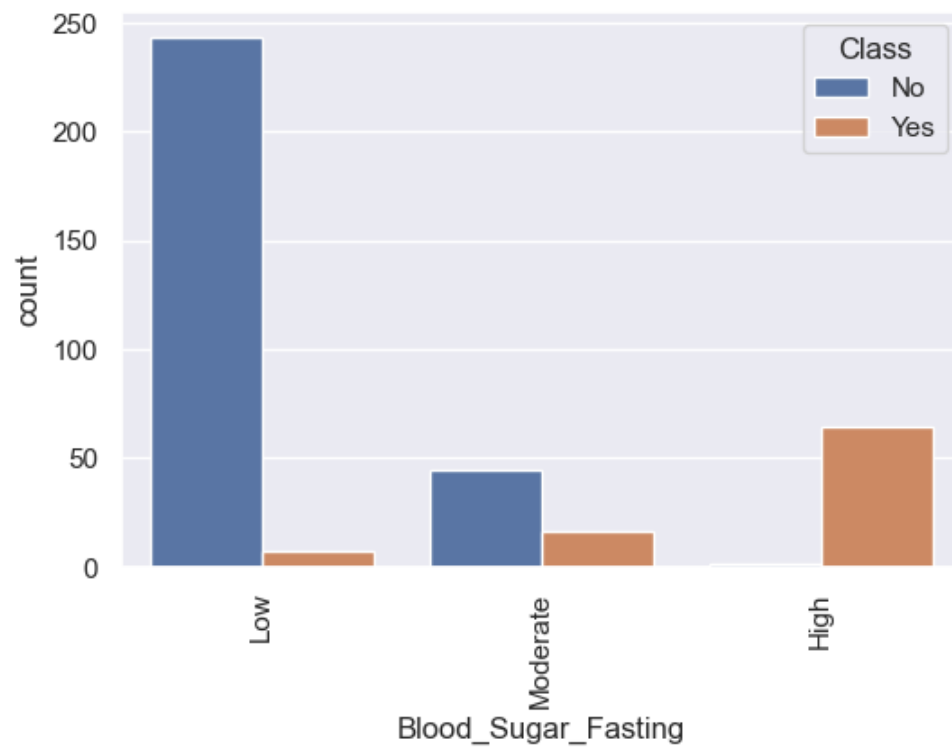


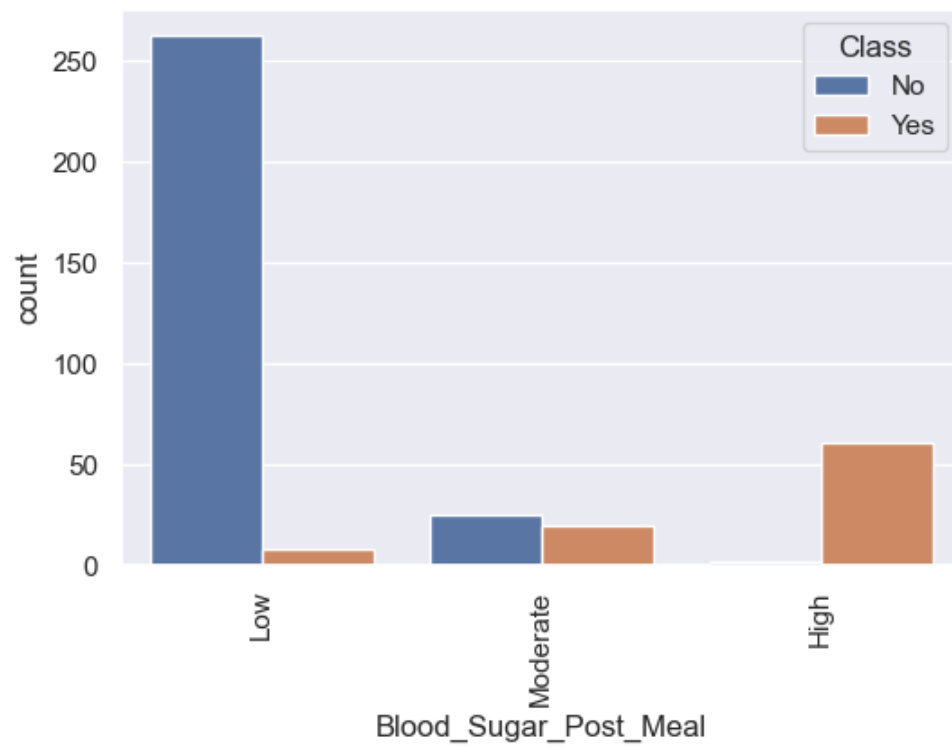


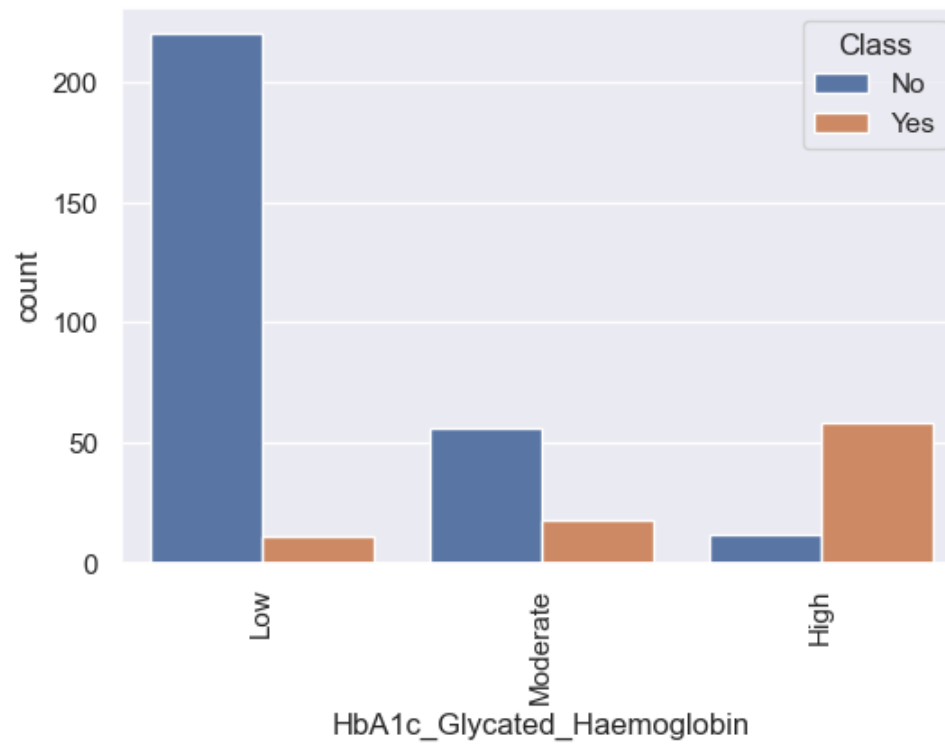












In [42]:

```
1 # LabelEncoder()
2 label_encoder = LabelEncoder()
3 for col in categorical_variables_df.columns:
4     # Encode labels in column
5     categorical_variables_df[col] = label_encoder.fit_transform(categorical_variables_df[col])
```



```
In [43]: 1 categorical_variables_df.head()
```

Out[43]:

	Gender	Profession	smoke	Exercise	cereal	salad	vegetables	grains	sweets	Sweets_in_a_week	...	Poor and unplanned work	Lack of career development	Feeling of powerlessness	Lack of financial security	Unable to satisfy all stakeholders	High_I
0	1	5	0	4	2	2	4	14	1	4	...	2	2	2	2	2	
1	0	4	0	1	4	5	4	17	1	4	...	0	0	2	2	2	
2	1	3	1	4	4	5	4	6	1	3	...	4	4	4	4	4	
3	1	4	0	3	3	5	4	20	1	4	...	0	0	4	0	0	
4	0	3	0	1	4	2	4	0	1	4	...	4	3	0	0	2	

5 rows × 31 columns

```
In [44]: 1 dataset = pd.concat([numerical_variables_df, categorical_variables_df], axis=1)
2 dataset.head()
```

Out[44]:

	Age(years)	Weight(kg)	Height(cm)	BMI	Gender	Profession	smoke	Exercise	cereal	salad	...	Poor and unplanned work	Lack of career development	Feeling of powerlessness	Lack of financial security	Unable to satisfy all stakeholders	High_Blood
0	42	61.0	165	22.41	1	5	0	4	2	2	...	2	2	2	2	2	
1	30	49.0	165	18.00	0	4	0	1	4	5	...	0	0	2	2	2	
2	52	60.0	159	23.73	1	3	1	4	4	5	...	4	4	4	4	4	
3	46	61.0	172	20.62	1	4	0	3	3	5	...	0	0	4	0	0	
4	45	65.0	155	27.06	0	3	0	1	4	2	...	4	3	0	0	2	

5 rows × 35 columns

```
In [45]: 1 X, Y = dataset.drop(['Class'], axis = 1), dataset['Class']
2 # train_test_split 80/20
3 X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, random_state = 42, stratify = Y)
```

In [46]:

```
1 # Model initialization
2 lr_Classifier = LogisticRegression()
3 knn_Classifier = KNeighborsClassifier()
4 gnb_Classifier = GaussianNB()
5 dt_Classifier = DecisionTreeClassifier()
6 rf_Classifier = RandomForestClassifier()
7 model_list = [lr_Classifier, knn_Classifier, gnb_Classifier, dt_Classifier, rf_Classifier]
8
9 # Scaler initialization
10 MinMax_scaler = MinMaxScaler()
11 Standard_scaler = StandardScaler()
12 MaxAbs_scaler = MaxAbsScaler()
13 Robust_scaler = RobustScaler()
14 Quantile_scaler = QuantileTransformer()
15 Power_scaler = PowerTransformer()
16 Normalizer_scaler = Normalizer()
17 scaler_list = [MinMax_scaler, Standard_scaler, MaxAbs_scaler, Robust_scaler,
18               Quantile_scaler, Power_scaler, Normalizer_scaler]
```



```
In [47]: 1 def run_pipeline(X_train, X_test, y_train, y_test, scaler, classifier):
2     # Model Information
3     print(f"Model name : {type(classifier).__name__}")
4     print(f"Scaler name : {type(scaler).__name__}")
5
6     # process 1 : fit and transform X_train data
7     scaled_X_train = scaler.fit_transform(X_train)
8
9     # process 2 : train model
10    classifier.fit(scaled_X_train, y_train)
11
12    # process 3 : transform X_test data
13    scaled_X_test = scaler.transform(X_test)
14
15    # process 4 : test model
16    y_pred = classifier.predict(scaled_X_test)
17    # print(y_pred, Le.inverse_transform(y_pred))
18
19    # process 5 : Perform k-fold cross-validation using cross_val_score
20    scores = cross_val_score(classifier, scaled_X_train, y_train, cv=10, scoring='accuracy')
21    print(f"10 K-Fold Accuracy_score : {np.round_(scores,4)}")
22    print(f"10 K-Fold Average Accuracy_score : {round(np.average(scores)*100,2)} %")
23
24    # process 6 : model evaluation
25    print("Accuracy_score:", round((accuracy_score(y_test, y_pred))*100,2), '%')
26    print("Loss:", round((1-accuracy_score(y_test, y_pred))*100,2), '%')
27    print("Cohen_kappa_score:", round((cohen_kappa_score(y_test, y_pred))*100,2), '%')
28    print("Classification_report:\n", metrics.classification_report(y_test, y_pred))
29    print("confusion_matrix:\n", confusion_matrix(y_test, y_pred))
30    # plot confusion_matrix
31    fig, ax = plt.subplots()
32    fig.set_size_inches(6,4) # WH
33    sns.heatmap(confusion_matrix(y_test, y_pred),
34                annot=True,
35                fmt=".1f",
36                linewidths = 2,
37                linecolor = "blue",
38                center=0)
39    plt.show()
40
41    # process 7 : save model in pkl file
42    filename = f'Moduls\\{str(type(classifier).__name__)}_{str(type(scaler).__name__)}_01_LS_Disease_Prediction.pkl'
43    pickle.dump(classifier, open(filename, 'wb'))
44
45    # collect data for bar plot
46    global plot_data_list
47    plot_data_list.append([str(type(classifier).__name__),
48                          str(type(scaler).__name__),
```

```
49         round((accuracy_score(y_test, y_pred))*100,2)])
50
51     # end
52     print("==="*30)
53     print("\n\n")
54     time.sleep(0.5)
```

```

In [48]: 1 for model in model_list:
          2     for scaler in scaler_list:
          3         run_pipeline(X_train, X_test, y_train, y_test, scaler, model)
          4
          5     # plot data
          6     plot_df = pd.DataFrame(plot_data_list, columns=['classifier', 'scaler', 'accuracy_score'])
          7     plot_df.to_csv(f"Dataset\\{str(type(model).__name__)}_accuracy_score_plot_data_01_LS_Disease_Prediction.csv", index=False)
          8     sns.set(rc={'figure.figsize':(18,6)})
          9     ax = sns.barplot(data=plot_df, x="classifier", y="accuracy_score", hue="scaler")
         10     plt.title('Accuracy Score Plot')
         11     plt.xlabel('Classifier')
         12     plt.ylabel('Accuracy Score')
         13     for i in ax.containers:
         14         ax.bar_label(i,)
         15     plt.show()
         16
         17     # empty list
         18     plot_data_list = []
         19     print("\n\n")
         20
         21 print("Done...")

```

Modele name : LogisticRegression

Scaler name : MinMaxScaler

10 K-Fold Accuracy_score : [0.9667 0.9 0.9667 0.8667 0.8667 0.8667 0.8667 0.9 0.8667 0.8667]

10 K-Fold Average Accuracy_score : 89.33 %

Accuracy_score: 88.0 %

Loss: 12.0 %

Cohen_kappa_score: 61.8 %

Classification_report:

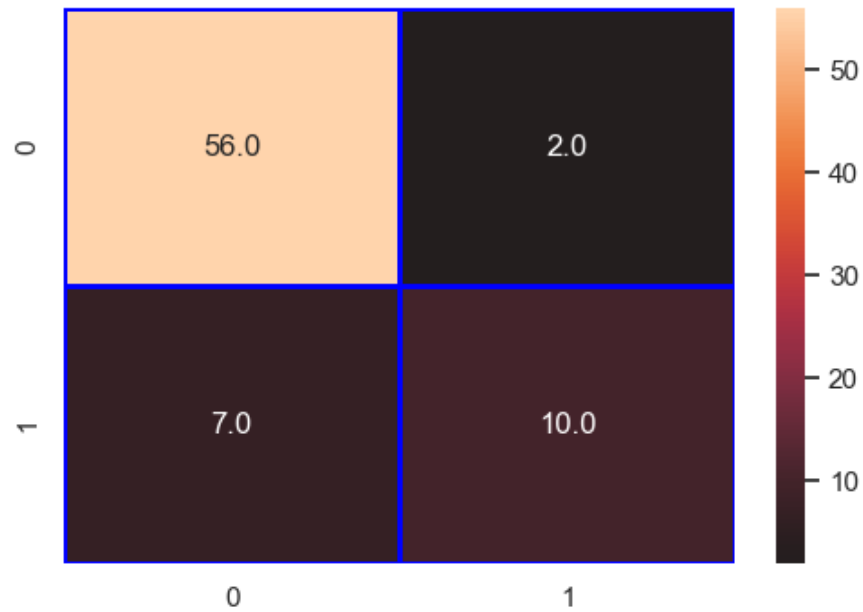
	precision	recall	f1-score	support
0	0.89	0.97	0.93	58
1	0.83	0.59	0.69	17
accuracy			0.88	75
macro avg	0.86	0.78	0.81	75
weighted avg	0.88	0.88	0.87	75

confusion_matrix:

```

[[56  2]
 [ 7 10]]

```



```

=====

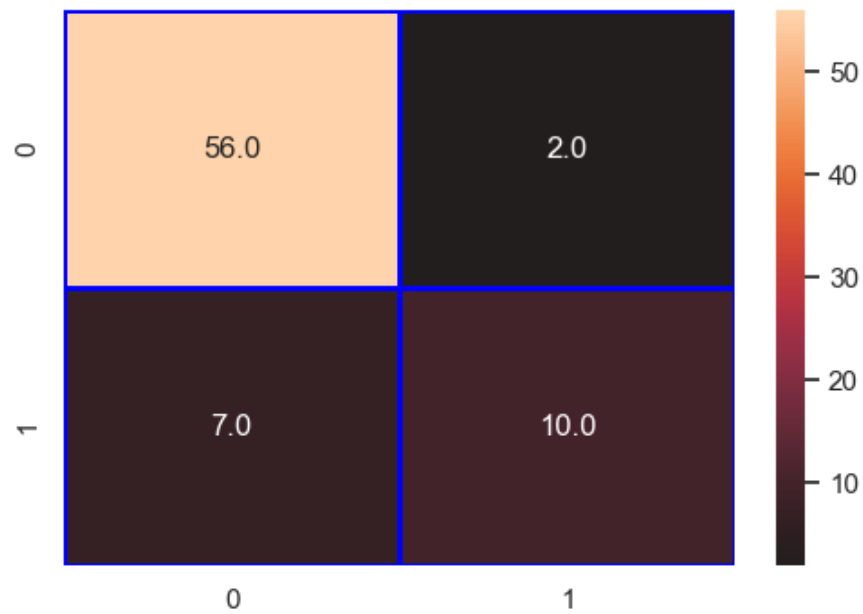
Modele name : LogisticRegression
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.9667 0.9667 0.8667 0.8667 0.8667 0.9    0.8333 0.9    0.7667 0.8667]
10 K-Fold Average Accuracy_score : 88.0 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 61.8 %
Classification_report:
      precision    recall  f1-score   support

     0       0.89      0.97      0.93        58
     1       0.83      0.59      0.69        17

   accuracy          0.88        75
  macro avg       0.86      0.78      0.81        75
 weighted avg       0.88      0.88      0.87        75

confusion_matrix:
[[56  2]
 [ 7 10]]

```



```

=====

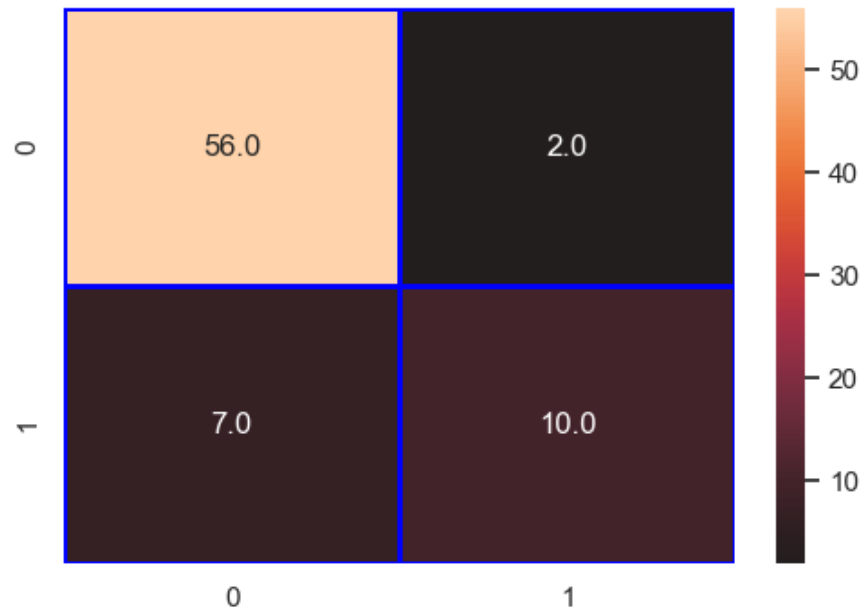
Modele name : LogisticRegression
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.9333 0.9    0.9    0.8667 0.8667 0.8667 0.8667 0.9    0.8667 0.8333]
10 K-Fold Average Accuracy_score : 88.0 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 61.8 %
Classification_report:
      precision    recall  f1-score   support

     0       0.89      0.97      0.93        58
     1       0.83      0.59      0.69        17

   accuracy          0.88        75
  macro avg       0.86      0.78      0.81        75
 weighted avg       0.88      0.88      0.87        75

confusion_matrix:
[[56  2]
 [ 7 10]]

```

```

=====

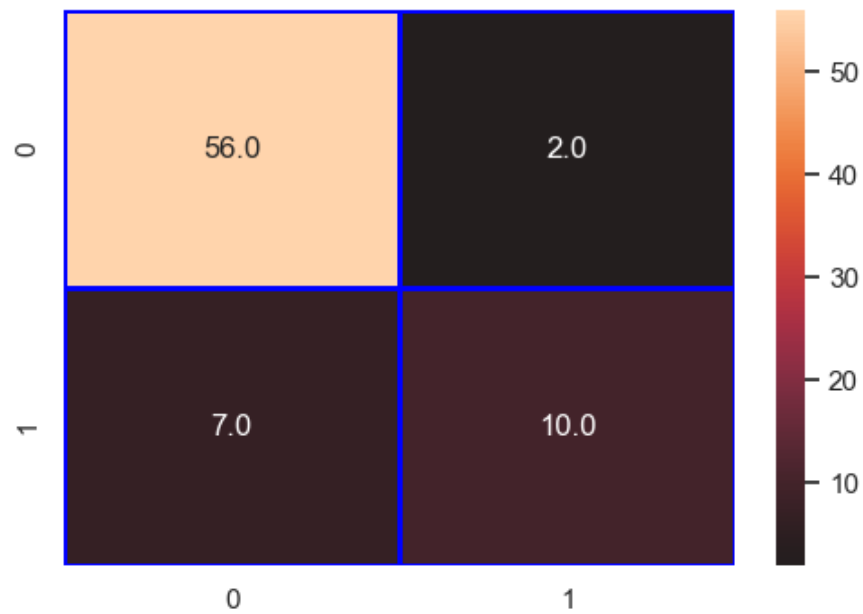
Modele name : LogisticRegression
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.9667 0.9333 0.9    0.9    0.8667 0.9    0.8333 0.9    0.8    0.8667]
10 K-Fold Average Accuracy_score : 88.67 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 61.8 %
Classification_report:
      precision    recall  f1-score   support

     0       0.89      0.97      0.93        58
     1       0.83      0.59      0.69        17

   accuracy          0.88        75
  macro avg       0.86      0.78      0.81        75
 weighted avg       0.88      0.88      0.87        75

confusion_matrix:
[[56  2]
 [ 7 10]]

```



```

=====

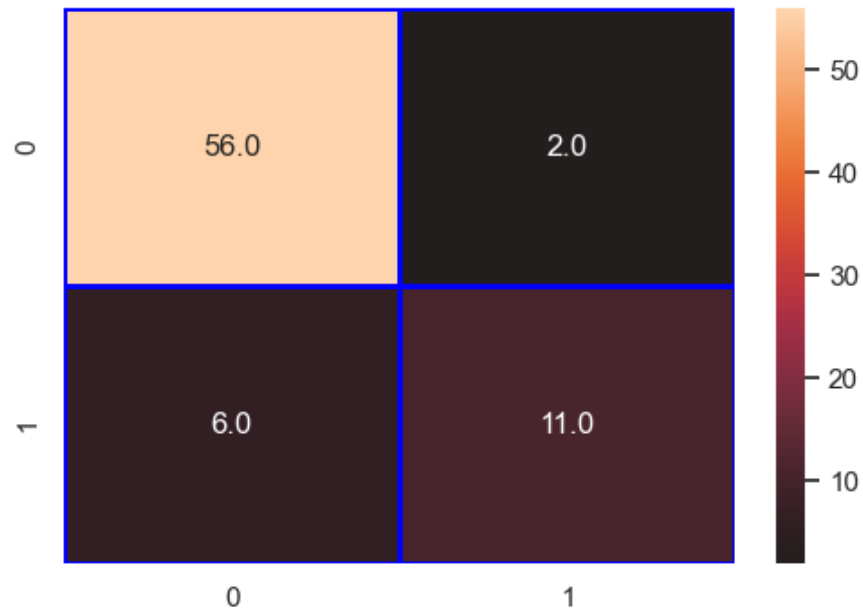
Modele name : LogisticRegression
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.9    0.9    0.9    0.9    0.8667 0.8667 0.8667 0.9    0.8667 0.8333]
10 K-Fold Average Accuracy_score : 88.0 %
Accuracy_score: 89.33 %
Loss: 10.67 %
Cohen_kappa_score: 66.81 %
Classification_report:
      precision    recall  f1-score   support

     0       0.90      0.97      0.93        58
     1       0.85      0.65      0.73        17

   accuracy          0.89        75
  macro avg       0.87      0.81      0.83        75
 weighted avg       0.89      0.89      0.89        75

confusion_matrix:
[[56  2]
 [ 6 11]]

```



```

=====

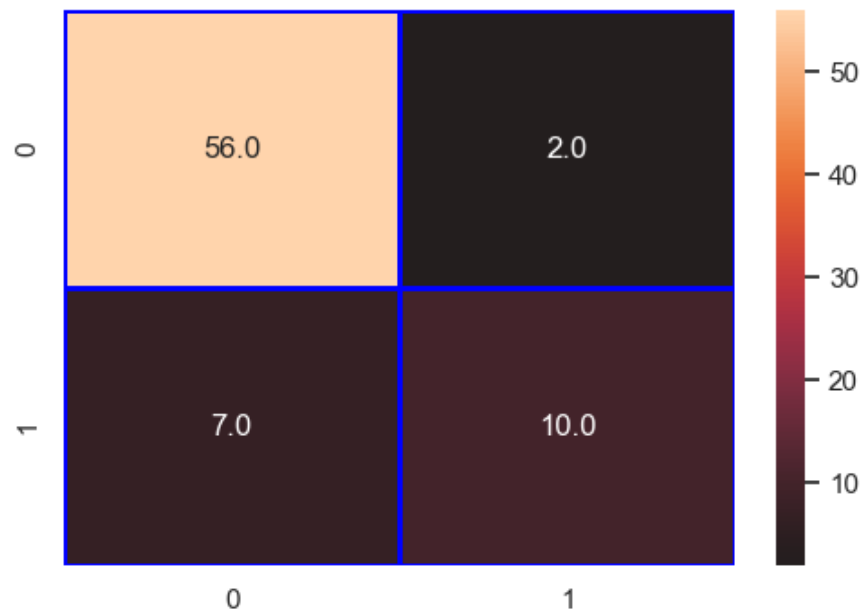
Modele name : LogisticRegression
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.9667 0.9667 0.9    0.8667 0.8333 0.9    0.8667 0.9    0.8667 0.8667]
10 K-Fold Average Accuracy_score : 89.33 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 61.8 %
Classification_report:
      precision    recall  f1-score   support

     0       0.89      0.97      0.93        58
     1       0.83      0.59      0.69        17

   accuracy          0.88        75
  macro avg       0.86      0.78      0.81        75
 weighted avg       0.88      0.88      0.87        75

confusion_matrix:
[[56  2]
 [ 7 10]]

```



```

=====

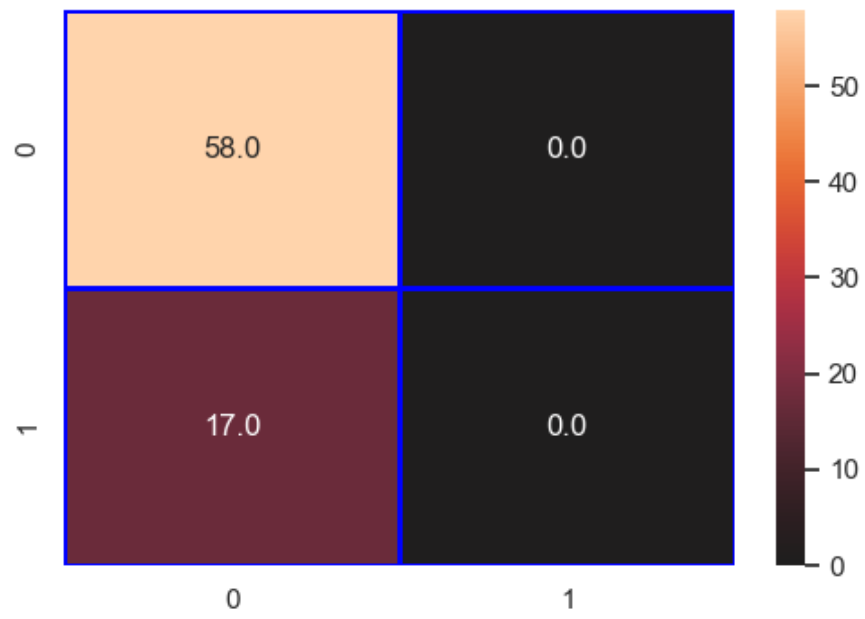
Modele name : LogisticRegression
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.7667 0.7667 0.7667 0.7667 0.7667 0.7667 0.7667 0.7667 0.7667 0.7667]
10 K-Fold Average Accuracy_score : 76.67 %
Accuracy_score: 77.33 %
Loss: 22.67 %
Cohen_kappa_score: 0.0 %
Classification_report:
      precision    recall  f1-score   support

     0       0.77      1.00      0.87        58
     1       0.00      0.00      0.00        17

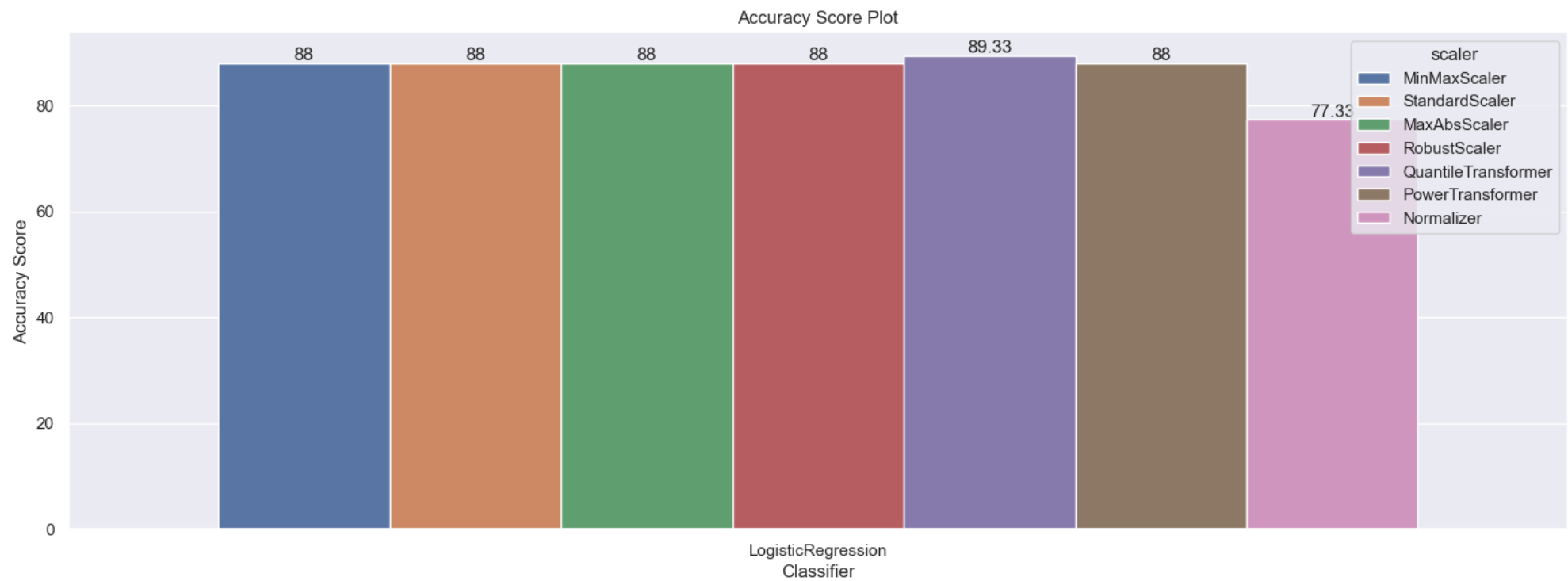
   accuracy          0.77        75
  macro avg       0.39      0.50      0.44        75
 weighted avg       0.60      0.77      0.67        75

confusion_matrix:
[[58  0]
 [17  0]]

```



=====



Modele name : KNeighborsClassifier

Scaler name : MinMaxScaler

10 K-Fold Accuracy_score : [0.8667 0.9 0.8667 0.8333 0.8667 0.8333 0.8333 0.8667 0.8667 0.8333]

10 K-Fold Average Accuracy_score : 85.67 %

Accuracy_score: 88.0 %

Loss: 12.0 %

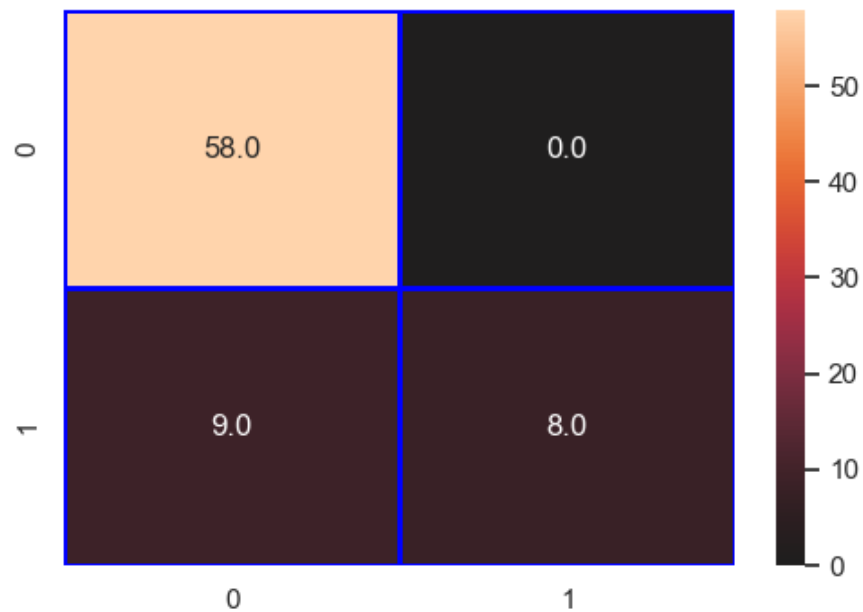
Cohen_kappa_score: 57.89 %

Classification_report:

	precision	recall	f1-score	support
0	0.87	1.00	0.93	58
1	1.00	0.47	0.64	17
accuracy			0.88	75
macro avg	0.93	0.74	0.78	75
weighted avg	0.90	0.88	0.86	75

confusion_matrix:

```
[[58  0]
 [ 9  8]]
```



```

=====

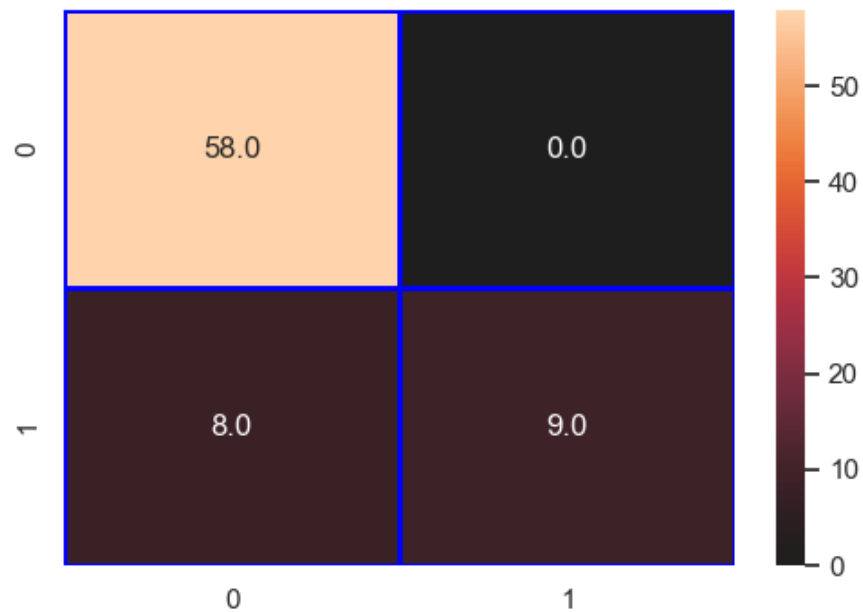
Modele name : KNeighborsClassifier
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.8333 0.9333 0.9    0.8333 0.8667 0.8667 0.9    0.9    0.9333 0.8667]
10 K-Fold Average Accuracy_score : 88.33 %
Accuracy_score: 89.33 %
Loss: 10.67 %
Cohen_kappa_score: 63.5 %
Classification_report:
      precision    recall  f1-score   support

     0       0.88      1.00      0.94         58
     1       1.00      0.53      0.69         17

   accuracy          0.89         75
  macro avg       0.94      0.76      0.81         75
 weighted avg       0.91      0.89      0.88         75

confusion_matrix:
[[58  0]
 [ 8  9]]

```



```

=====

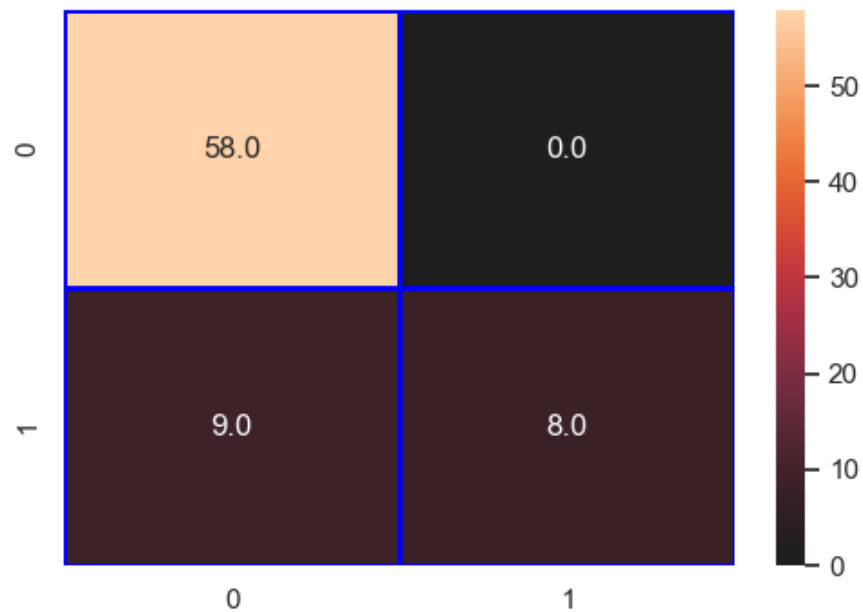
Modele name : KNeighborsClassifier
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.8667 0.9    0.8667 0.8333 0.8667 0.8333 0.8333 0.8667 0.8667 0.8333]
10 K-Fold Average Accuracy_score : 85.67 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 57.89 %
Classification_report:
      precision    recall  f1-score   support

     0       0.87      1.00      0.93        58
     1       1.00      0.47      0.64        17

   accuracy          0.88          75
  macro avg       0.93      0.74      0.78          75
 weighted avg       0.90      0.88      0.86          75

confusion_matrix:
[[58  0]
 [ 9  8]]

```

```

=====

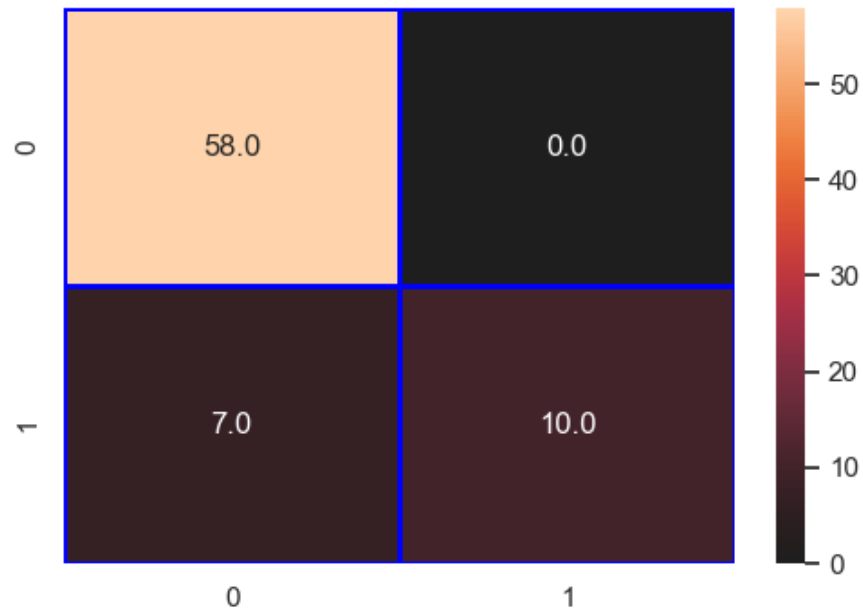
Modele name : KNeighborsClassifier
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.8    0.8667 0.9333 0.9    0.8667 0.8667 0.9    0.8667 0.9333 0.8667]
10 K-Fold Average Accuracy_score : 88.0 %
Accuracy_score: 90.67 %
Loss: 9.33 %
Cohen_kappa_score: 68.84 %
Classification_report:
      precision    recall  f1-score   support

     0       0.89      1.00      0.94         58
     1       1.00      0.59      0.74         17

   accuracy          0.91         75
  macro avg       0.95      0.79      0.84         75
 weighted avg       0.92      0.91      0.90         75

confusion_matrix:
[[58  0]
 [ 7 10]]

```



```

=====

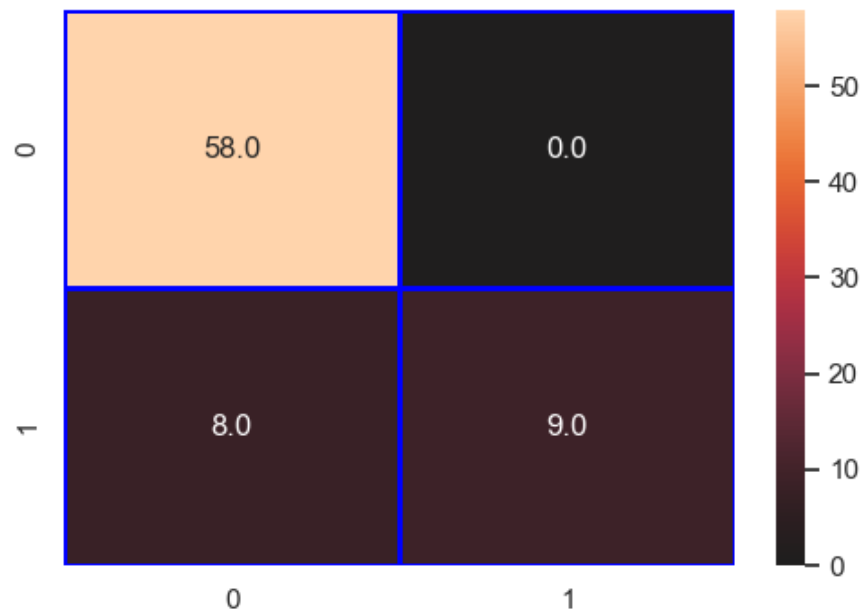
Modele name : KNeighborsClassifier
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.8    0.9    0.9    0.8333 0.8333 0.8333 0.7333 0.8333 0.8333 0.8667]
10 K-Fold Average Accuracy_score : 83.67 %
Accuracy_score: 89.33 %
Loss: 10.67 %
Cohen_kappa_score: 63.5 %
Classification_report:
      precision    recall  f1-score   support

     0       0.88      1.00      0.94        58
     1       1.00      0.53      0.69        17

   accuracy          0.89        75
  macro avg       0.94      0.76      0.81        75
 weighted avg       0.91      0.89      0.88        75

confusion_matrix:
[[58  0]
 [ 8  9]]

```



```

=====

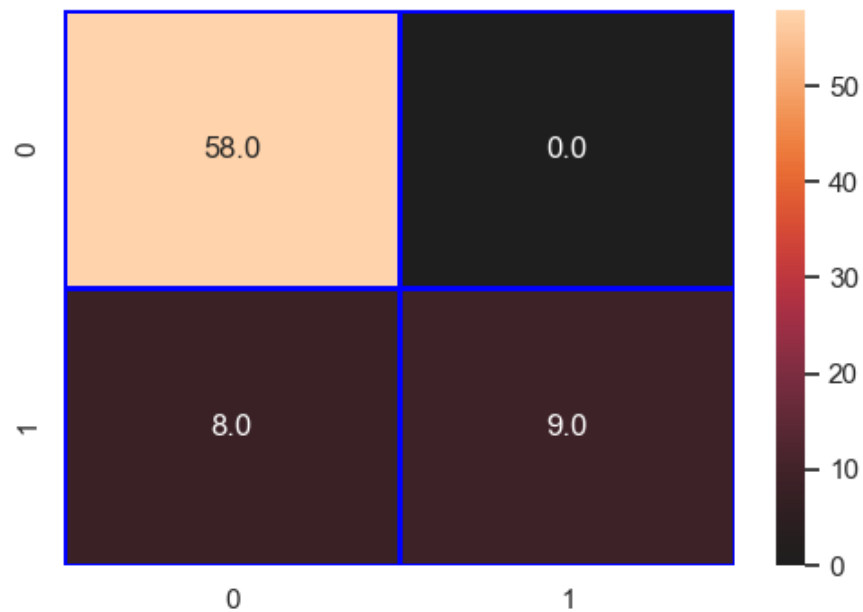
Modele name : KNeighborsClassifier
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.8333 0.9333 0.9333 0.8667 0.8667 0.8667 0.9    0.9    0.9    0.8667]
10 K-Fold Average Accuracy_score : 88.67 %
Accuracy_score: 89.33 %
Loss: 10.67 %
Cohen_kappa_score: 63.5 %
Classification_report:
      precision    recall  f1-score   support

     0       0.88      1.00      0.94         58
     1       1.00      0.53      0.69         17

   accuracy          0.89         75
  macro avg       0.94      0.76      0.81         75
 weighted avg       0.91      0.89      0.88         75

confusion_matrix:
[[58  0]
 [ 8  9]]

```



```

=====

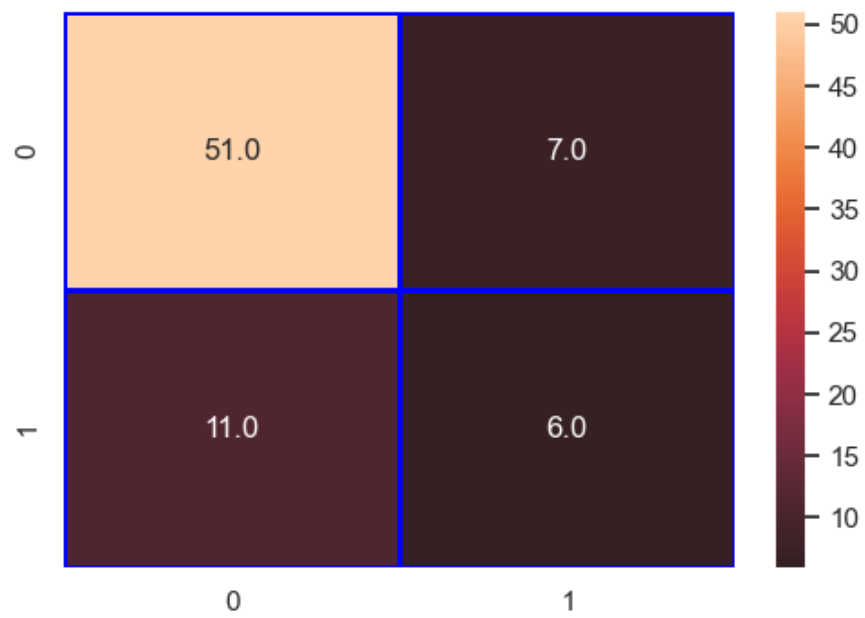
Modele name : KNeighborsClassifier
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.8    0.8333 0.8333 0.8    0.7333 0.7667 0.7667 0.8    0.7333 0.7    ]
10 K-Fold Average Accuracy_score : 77.67 %
Accuracy_score: 76.0 %
Loss: 24.0 %
Cohen_kappa_score: 25.33 %
Classification_report:
      precision    recall  f1-score   support

     0       0.82      0.88      0.85        58
     1       0.46      0.35      0.40        17

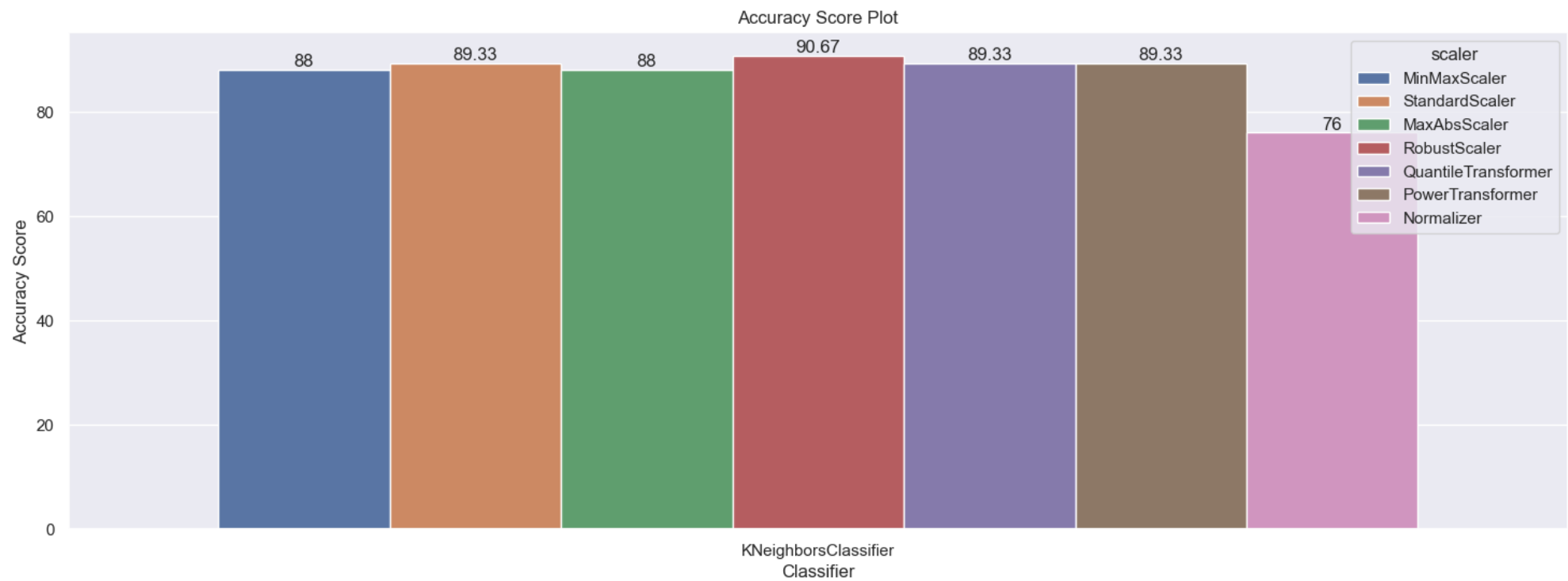
   accuracy          0.76        75
  macro avg       0.64      0.62      0.63        75
 weighted avg       0.74      0.76      0.75        75

confusion_matrix:
[[51  7]
 [11  6]]

```



=====



Modele name : GaussianNB

Scaler name : MinMaxScaler

10 K-Fold Accuracy_score : [0.9667 0.9 0.8667 0.8 0.8667 0.8667 0.9667 0.9333 0.8667 0.9]

10 K-Fold Average Accuracy_score : 89.33 %

Accuracy_score: 84.0 %

Loss: 16.0 %

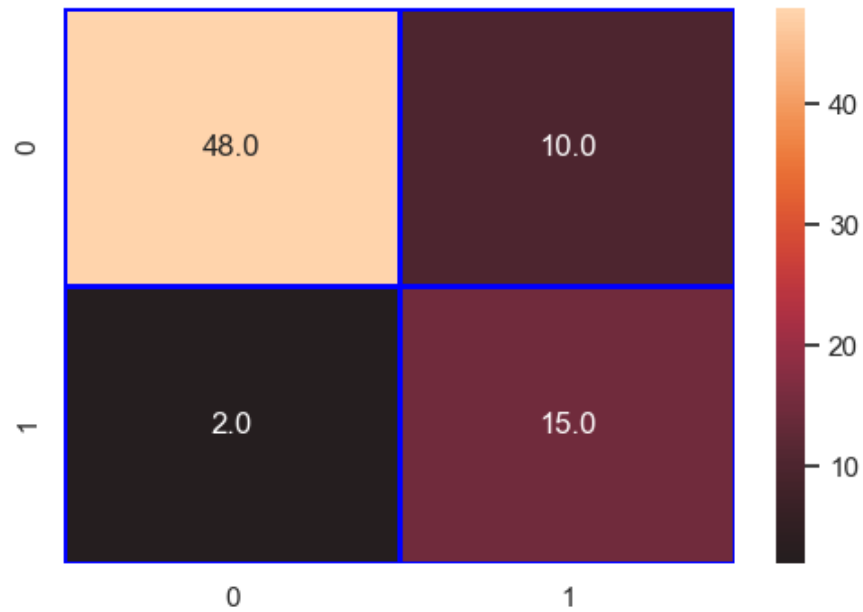
Cohen_kappa_score: 60.87 %

Classification_report:

	precision	recall	f1-score	support
0	0.96	0.83	0.89	58
1	0.60	0.88	0.71	17
accuracy			0.84	75
macro avg	0.78	0.85	0.80	75
weighted avg	0.88	0.84	0.85	75

confusion_matrix:

```
[[48 10]
 [ 2 15]]
```



```

=====

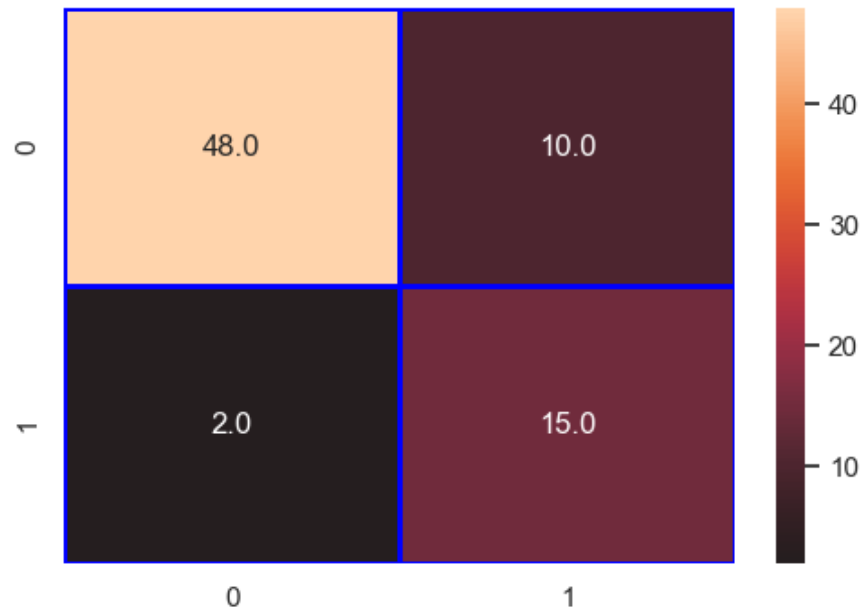
Modele name : GaussianNB
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.9667 0.9    0.8667 0.8    0.8667 0.8667 0.9667 0.9333 0.8667 0.9    ]
10 K-Fold Average Accuracy_score : 89.33 %
Accuracy_score: 84.0 %
Loss: 16.0 %
Cohen_kappa_score: 60.87 %
Classification_report:
      precision    recall  f1-score   support

     0       0.96      0.83      0.89        58
     1       0.60      0.88      0.71        17

   accuracy          0.84        75
  macro avg       0.78      0.85      0.80        75
 weighted avg       0.88      0.84      0.85        75

confusion_matrix:
[[48 10]
 [ 2 15]]

```



```

=====

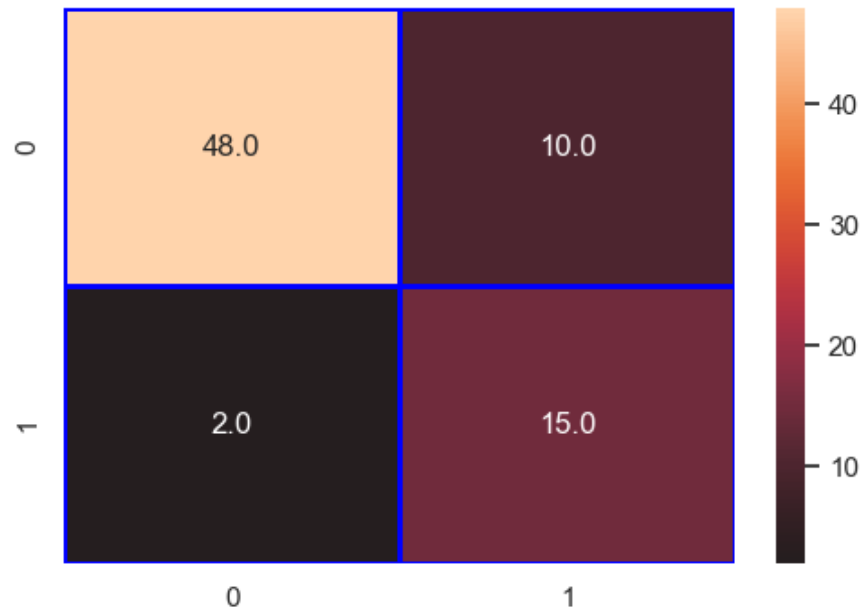
Modele name : GaussianNB
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.9667 0.9    0.8667 0.8    0.8667 0.8667 0.9667 0.9333 0.8667 0.9    ]
10 K-Fold Average Accuracy_score : 89.33 %
Accuracy_score: 84.0 %
Loss: 16.0 %
Cohen_kappa_score: 60.87 %
Classification_report:
      precision    recall  f1-score   support

     0       0.96      0.83      0.89        58
     1       0.60      0.88      0.71        17

   accuracy          0.84        75
  macro avg       0.78      0.85      0.80        75
 weighted avg       0.88      0.84      0.85        75

confusion_matrix:
[[48 10]
 [ 2 15]]

```

```

=====

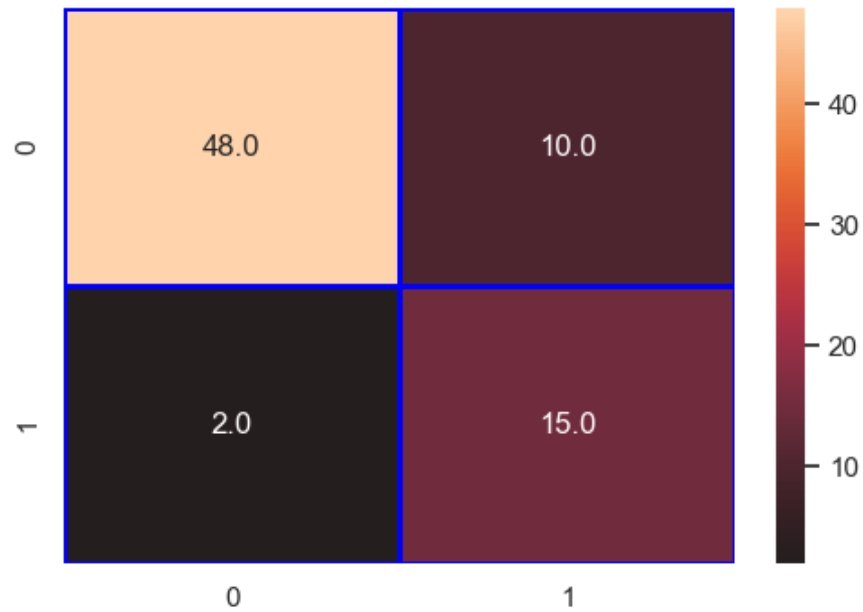
Modele name : GaussianNB
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.9667 0.9    0.8667 0.8    0.8667 0.8667 0.9667 0.9333 0.8667 0.9    ]
10 K-Fold Average Accuracy_score : 89.33 %
Accuracy_score: 84.0 %
Loss: 16.0 %
Cohen_kappa_score: 60.87 %
Classification_report:
      precision    recall  f1-score   support

     0       0.96      0.83      0.89         58
     1       0.60      0.88      0.71         17

   accuracy          0.84         75
  macro avg       0.78      0.85      0.80         75
 weighted avg       0.88      0.84      0.85         75

confusion_matrix:
[[48 10]
 [ 2 15]]

```



```

=====

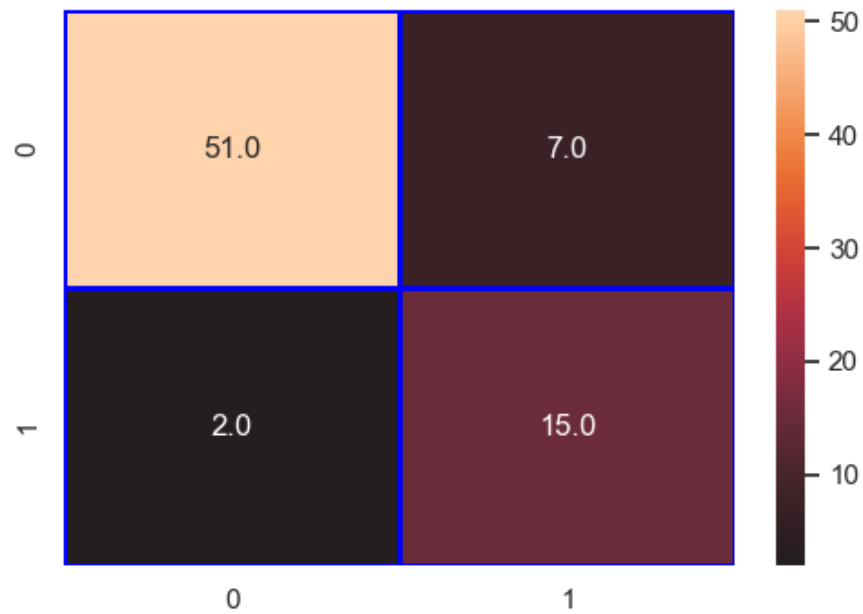
Modele name : GaussianNB
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.9667 0.9    0.8667 0.8    0.8667 0.8333 0.9667 0.9    0.9333 0.8333]
10 K-Fold Average Accuracy_score : 88.67 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 68.99 %
Classification_report:
      precision    recall  f1-score   support

     0       0.96      0.88      0.92        58
     1       0.68      0.88      0.77        17

   accuracy          0.88        75
  macro avg       0.82      0.88      0.84        75
 weighted avg       0.90      0.88      0.88        75

confusion_matrix:
[[51  7]
 [ 2 15]]

```



```

=====

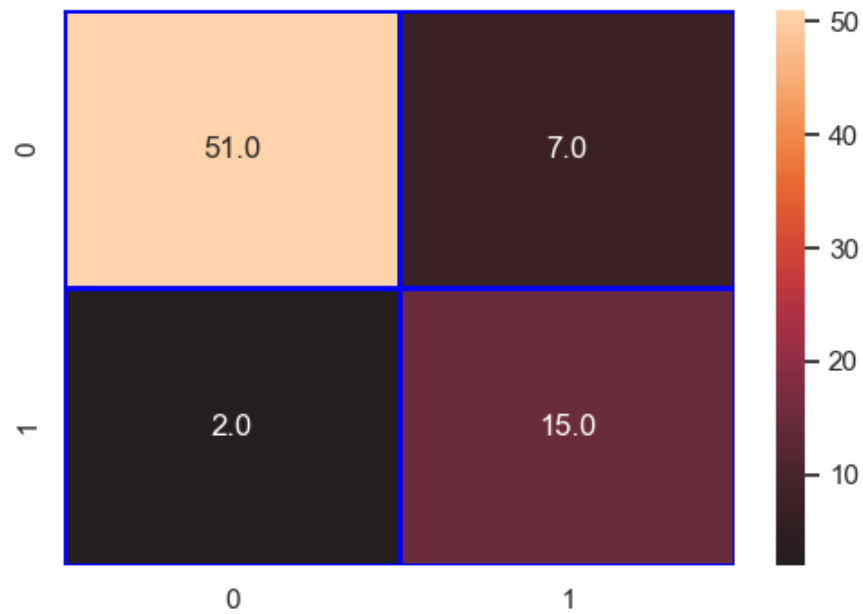
Modele name : GaussianNB
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.9667 0.9    0.8667 0.8333 0.8333 0.8667 0.9667 0.9    0.9333 0.8333]
10 K-Fold Average Accuracy_score : 89.0 %
Accuracy_score: 88.0 %
Loss: 12.0 %
Cohen_kappa_score: 68.99 %
Classification_report:
      precision    recall  f1-score   support

     0       0.96      0.88      0.92        58
     1       0.68      0.88      0.77        17

   accuracy          0.88        75
  macro avg       0.82      0.88      0.84        75
 weighted avg       0.90      0.88      0.88        75

confusion_matrix:
[[51  7]
 [ 2 15]]

```



```

=====

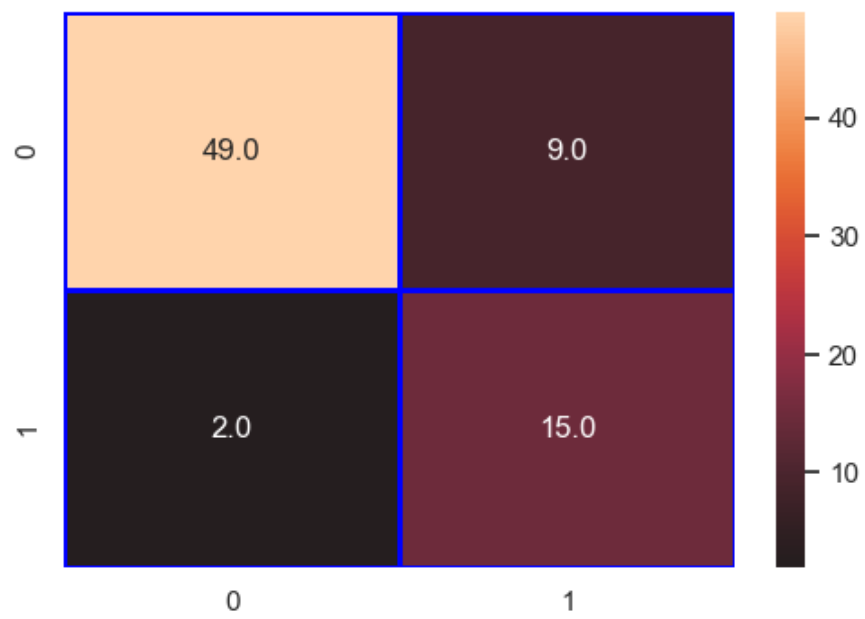
Modele name : GaussianNB
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.9667 0.9    0.8667 0.8    0.8667 0.8667 0.9    0.9    0.8667 0.8333]
10 K-Fold Average Accuracy_score : 87.67 %
Accuracy_score: 85.33 %
Loss: 14.67 %
Cohen_kappa_score: 63.48 %
Classification_report:
      precision    recall  f1-score   support

     0       0.96      0.84      0.90        58
     1       0.62      0.88      0.73        17

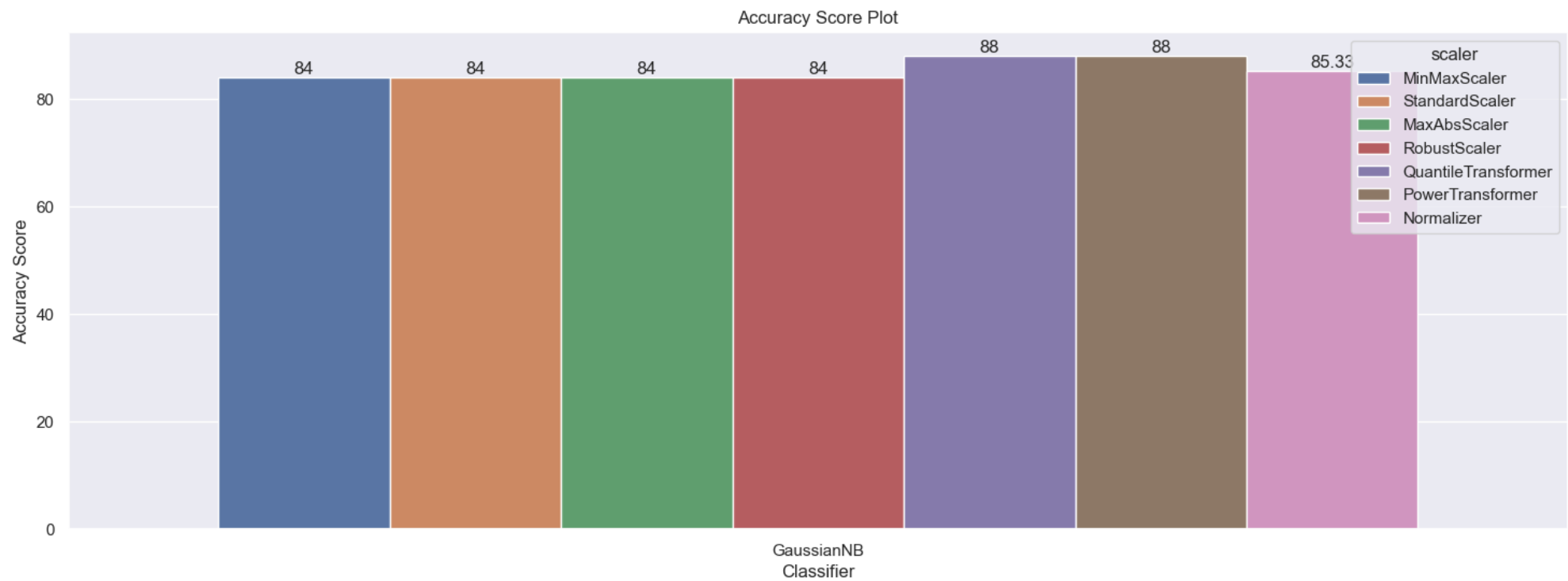
   accuracy          0.85        75
  macro avg       0.79      0.86      0.82        75
 weighted avg       0.88      0.85      0.86        75

confusion_matrix:
[[49  9]
 [ 2 15]]

```



=====



Modele name : DecisionTreeClassifier

Scaler name : MinMaxScaler

10 K-Fold Accuracy_score : [0.9667 0.9 0.9667 0.9 0.8667 1. 0.9 0.9 0.9333 0.9]

10 K-Fold Average Accuracy_score : 92.33 %

Accuracy_score: 92.0 %

Loss: 8.0 %

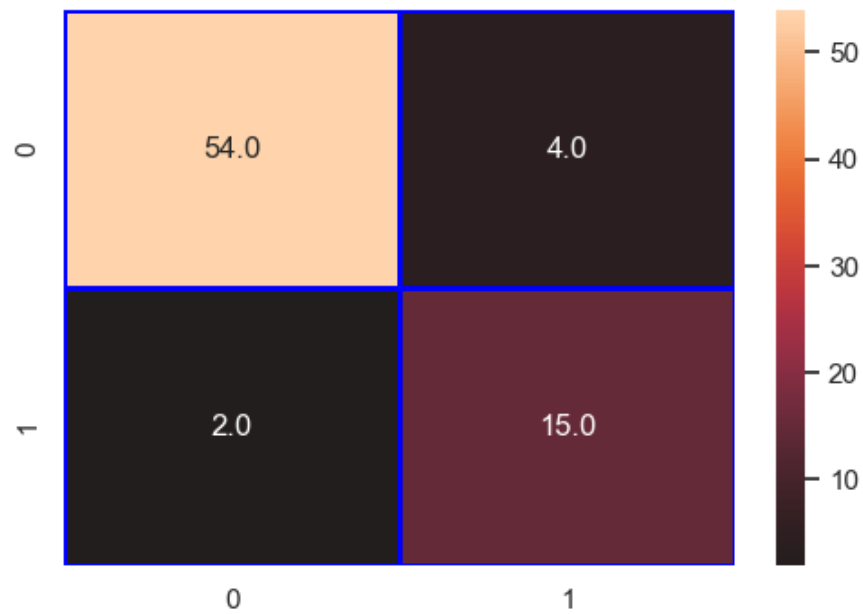
Cohen_kappa_score: 78.09 %

Classification_report:

	precision	recall	f1-score	support
0	0.96	0.93	0.95	58
1	0.79	0.88	0.83	17
accuracy			0.92	75
macro avg	0.88	0.91	0.89	75
weighted avg	0.92	0.92	0.92	75

confusion_matrix:

```
[[54  4]
 [ 2 15]]
```



```

=====

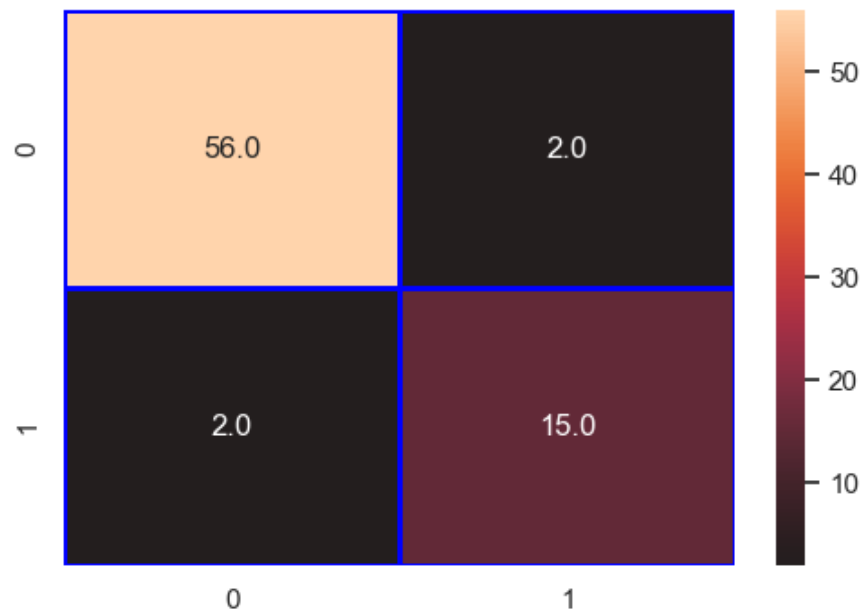
Modele name : DecisionTreeClassifier
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [0.9333 0.9    0.9667 0.9    0.8667 0.9667 0.9    0.9    0.9667 0.9    ]
10 K-Fold Average Accuracy_score : 92.0 %
Accuracy_score: 94.67 %
Loss: 5.33 %
Cohen_kappa_score: 84.79 %
Classification_report:
      precision    recall  f1-score   support

     0       0.97      0.97      0.97        58
     1       0.88      0.88      0.88        17

   accuracy          0.95          75
  macro avg       0.92      0.92      0.92          75
 weighted avg       0.95      0.95      0.95          75

confusion_matrix:
[[56  2]
 [ 2 15]]

```



```

=====

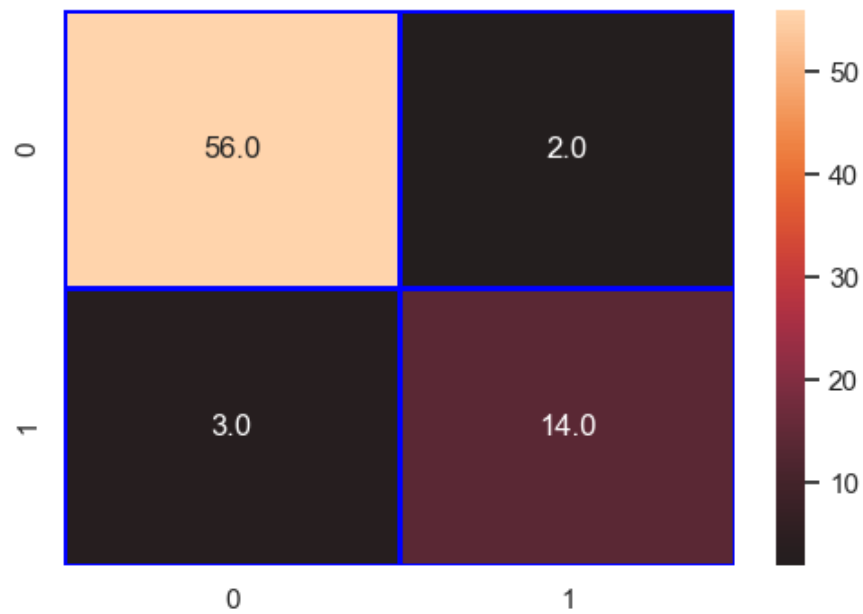
Modele name : DecisionTreeClassifier
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [0.9333 0.9    0.9667 0.9    0.9    0.9333 0.9    0.8667 0.9333 0.9    ]
10 K-Fold Average Accuracy_score : 91.33 %
Accuracy_score: 93.33 %
Loss: 6.67 %
Cohen_kappa_score: 80.58 %
Classification_report:
      precision    recall  f1-score   support

     0       0.95      0.97      0.96         58
     1       0.88      0.82      0.85         17

   accuracy          0.93         75
  macro avg       0.91      0.89      0.90         75
 weighted avg       0.93      0.93      0.93         75

confusion_matrix:
[[56  2]
 [ 3 14]]

```

```

=====

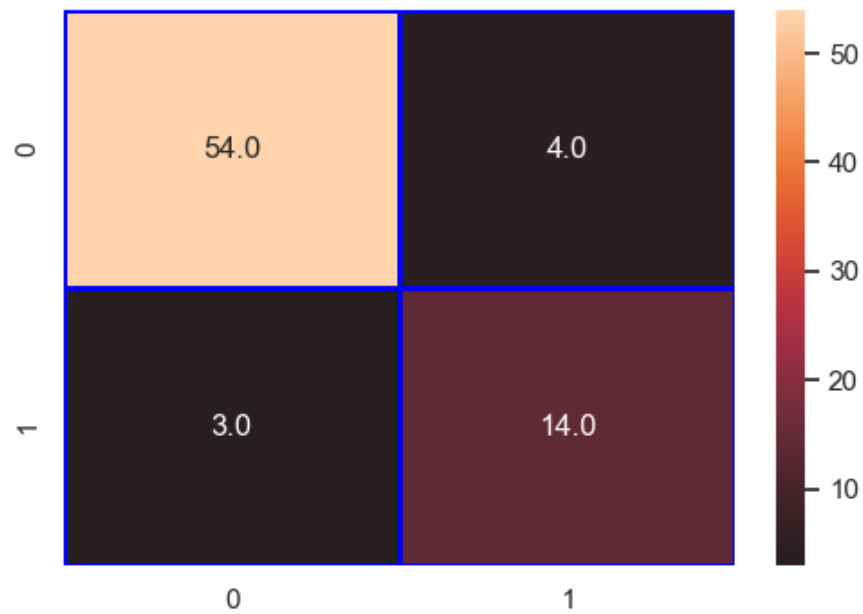
Modele name : DecisionTreeClassifier
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.9333 0.9    0.9667 0.9    0.9    0.9333 0.9    0.9    0.9    0.9    ]
10 K-Fold Average Accuracy_score : 91.33 %
Accuracy_score: 90.67 %
Loss: 9.33 %
Cohen_kappa_score: 73.92 %
Classification_report:
      precision    recall  f1-score   support

     0       0.95      0.93      0.94         58
     1       0.78      0.82      0.80         17

   accuracy          0.91         75
  macro avg       0.86      0.88      0.87         75
 weighted avg       0.91      0.91      0.91         75

confusion_matrix:
[[54  4]
 [ 3 14]]

```



```

=====

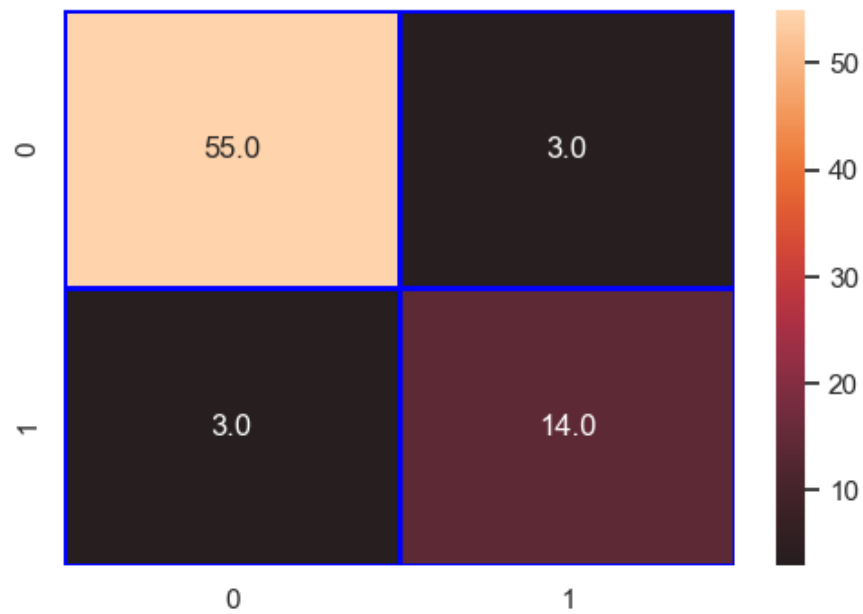
Modele name : DecisionTreeClassifier
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.9667 0.8667 0.9667 0.9    0.8333 1.    0.9    0.9    0.9667 0.9    ]
10 K-Fold Average Accuracy_score : 92.0 %
Accuracy_score: 92.0 %
Loss: 8.0 %
Cohen_kappa_score: 77.18 %
Classification_report:
      precision    recall  f1-score   support

     0       0.95      0.95      0.95        58
     1       0.82      0.82      0.82        17

   accuracy          0.92        75
  macro avg       0.89      0.89      0.89        75
 weighted avg       0.92      0.92      0.92        75

confusion_matrix:
[[55  3]
 [ 3 14]]

```



```

=====

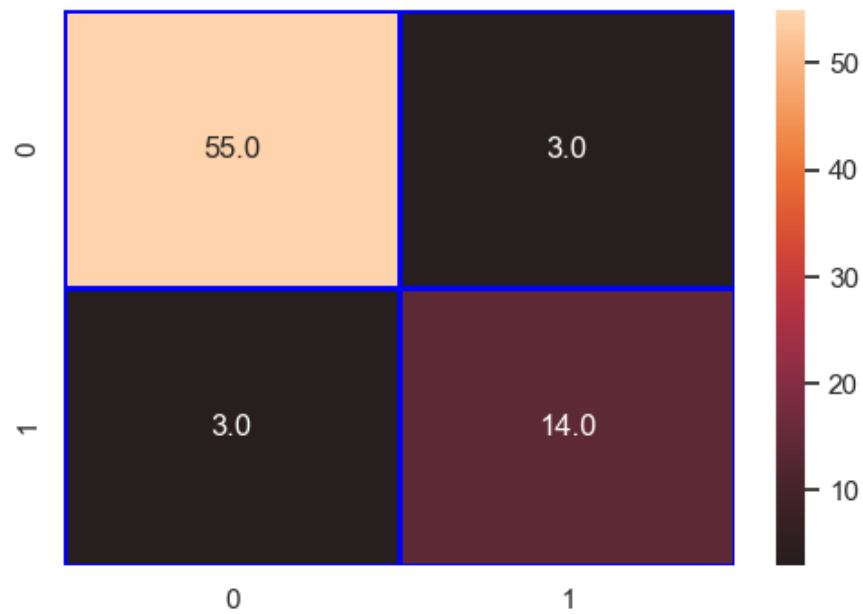
Modele name : DecisionTreeClassifier
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [0.9667 0.9333 0.9667 0.9    0.9    0.9333 0.9    0.9    0.9333 0.9    ]
10 K-Fold Average Accuracy_score : 92.33 %
Accuracy_score: 92.0 %
Loss: 8.0 %
Cohen_kappa_score: 77.18 %
Classification_report:
      precision    recall  f1-score   support

     0       0.95      0.95      0.95        58
     1       0.82      0.82      0.82        17

   accuracy          0.92        75
  macro avg       0.89      0.89      0.89        75
 weighted avg       0.92      0.92      0.92        75

confusion_matrix:
[[55  3]
 [ 3 14]]

```



```

=====

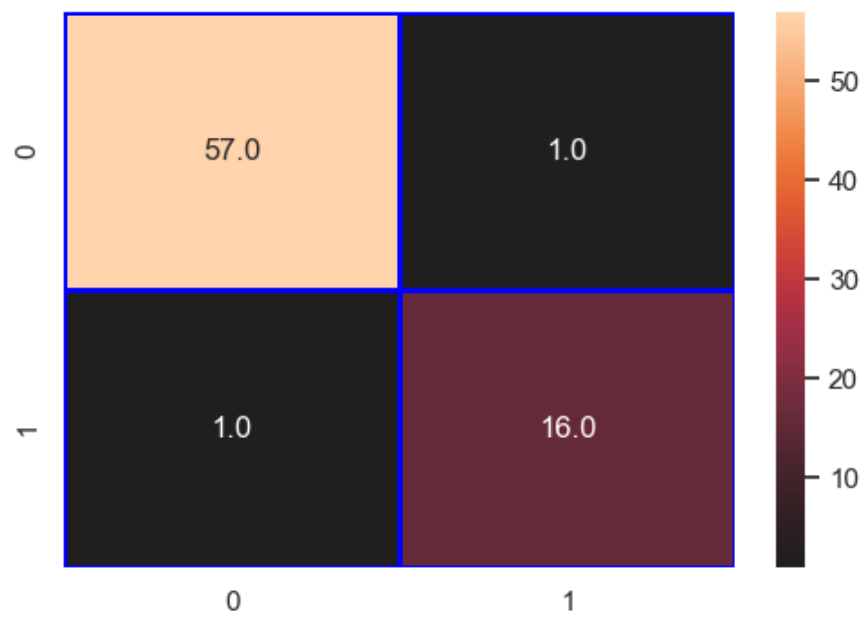
Modele name : DecisionTreeClassifier
Scaler name : Normalizer
10 K-Fold Accuracy_score : [1.      0.9333 0.9667 0.8333 0.9    0.9333 0.9333 0.9333 0.9667 0.9667]
10 K-Fold Average Accuracy_score : 93.67 %
Accuracy_score: 97.33 %
Loss: 2.67 %
Cohen_kappa_score: 92.39 %
Classification_report:
      precision    recall  f1-score   support

     0       0.98      0.98      0.98        58
     1       0.94      0.94      0.94        17

   accuracy          0.97        75
  macro avg       0.96      0.96      0.96        75
 weighted avg       0.97      0.97      0.97        75

confusion_matrix:
[[57  1]
 [ 1 16]]

```



=====



Modele name : RandomForestClassifier

Scaler name : MinMaxScaler

10 K-Fold Accuracy_score : [1. 0.9 0.9667 0.8667 0.9 0.9667 0.9333 1. 0.9667 0.9333]

10 K-Fold Average Accuracy_score : 94.33 %

Accuracy_score: 93.33 %

Loss: 6.67 %

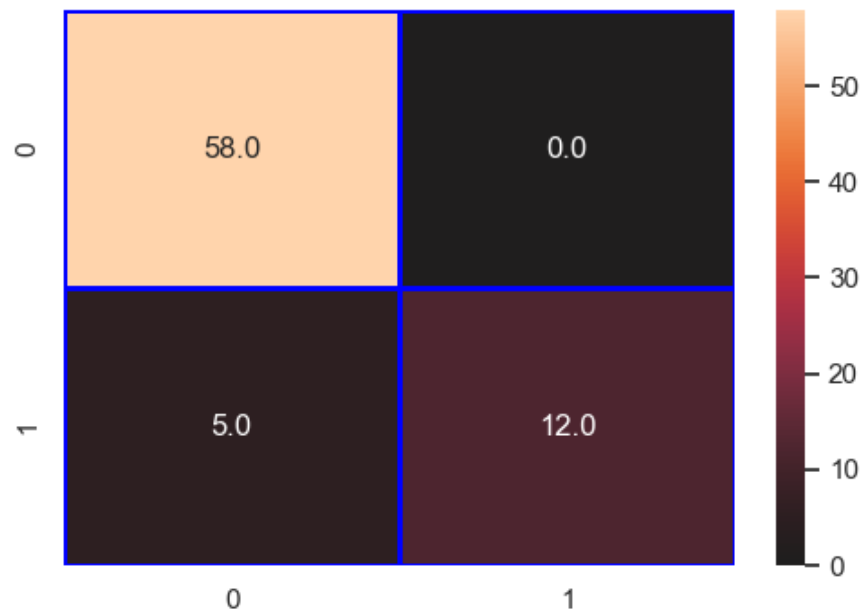
Cohen_kappa_score: 78.78 %

Classification_report:

	precision	recall	f1-score	support
0	0.92	1.00	0.96	58
1	1.00	0.71	0.83	17
accuracy			0.93	75
macro avg	0.96	0.85	0.89	75
weighted avg	0.94	0.93	0.93	75

confusion_matrix:

```
[[58 0]
 [ 5 12]]
```



```

=====

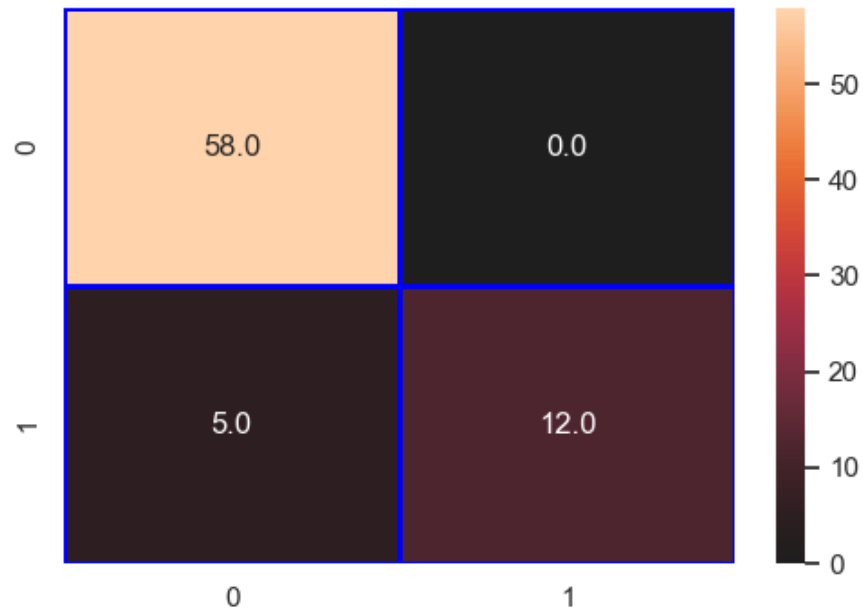
Modele name : RandomForestClassifier
Scaler name : StandardScaler
10 K-Fold Accuracy_score : [1.      0.9333 0.9667 0.8667 0.9    0.9667 0.9    0.9667 1.      0.9333]
10 K-Fold Average Accuracy_score : 94.33 %
Accuracy_score: 93.33 %
Loss: 6.67 %
Cohen_kappa_score: 78.78 %
Classification_report:
      precision    recall  f1-score   support

     0       0.92      1.00      0.96         58
     1       1.00      0.71      0.83         17

   accuracy          0.93         75
  macro avg       0.96      0.85      0.89         75
 weighted avg       0.94      0.93      0.93         75

confusion_matrix:
[[58  0]
 [ 5 12]]

```



```

=====

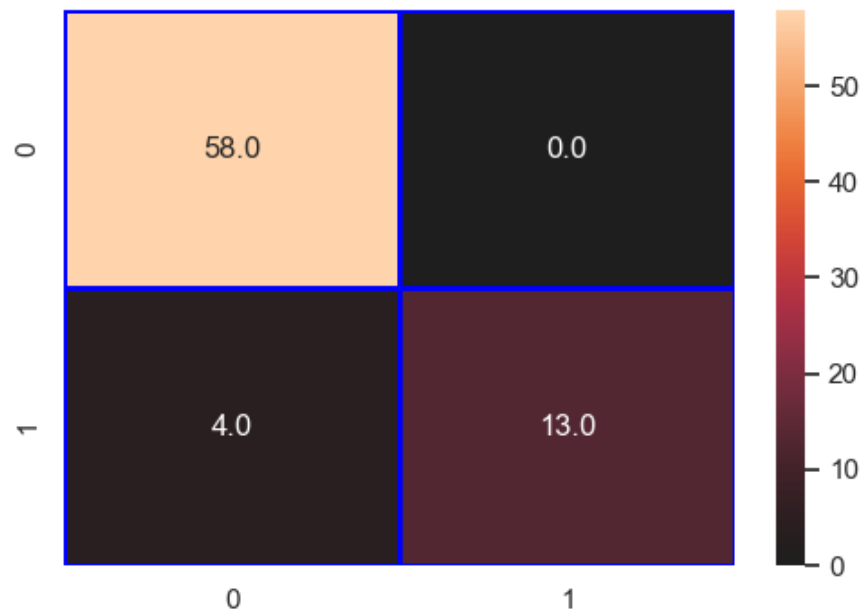
Modele name : RandomForestClassifier
Scaler name : MaxAbsScaler
10 K-Fold Accuracy_score : [1.      0.9333 0.9667 0.8667 0.9      0.9667 0.9      0.9667 1.      0.9333]
10 K-Fold Average Accuracy_score : 94.33 %
Accuracy_score: 94.67 %
Loss: 5.33 %
Cohen_kappa_score: 83.41 %
Classification_report:
      precision    recall  f1-score   support

     0       0.94      1.00      0.97        58
     1       1.00      0.76      0.87        17

   accuracy          0.95          75
  macro avg       0.97      0.88      0.92          75
 weighted avg       0.95      0.95      0.94          75

confusion_matrix:
[[58  0]
 [ 4 13]]

```

```

=====

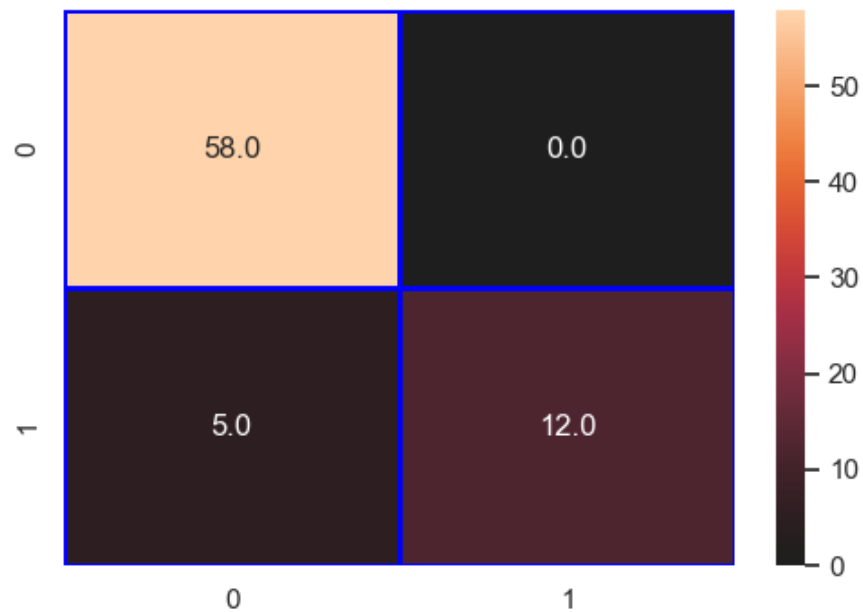
Modele name : RandomForestClassifier
Scaler name : RobustScaler
10 K-Fold Accuracy_score : [0.9667 0.9667 0.9667 0.8667 0.9    0.9667 0.9    0.9667 1.    0.9333]
10 K-Fold Average Accuracy_score : 94.33 %
Accuracy_score: 93.33 %
Loss: 6.67 %
Cohen_kappa_score: 78.78 %
Classification_report:
      precision    recall  f1-score   support

     0       0.92      1.00      0.96         58
     1       1.00      0.71      0.83         17

   accuracy          0.93         75
  macro avg       0.96      0.85      0.89         75
 weighted avg       0.94      0.93      0.93         75

confusion_matrix:
[[58  0]
 [ 5 12]]

```



```

=====

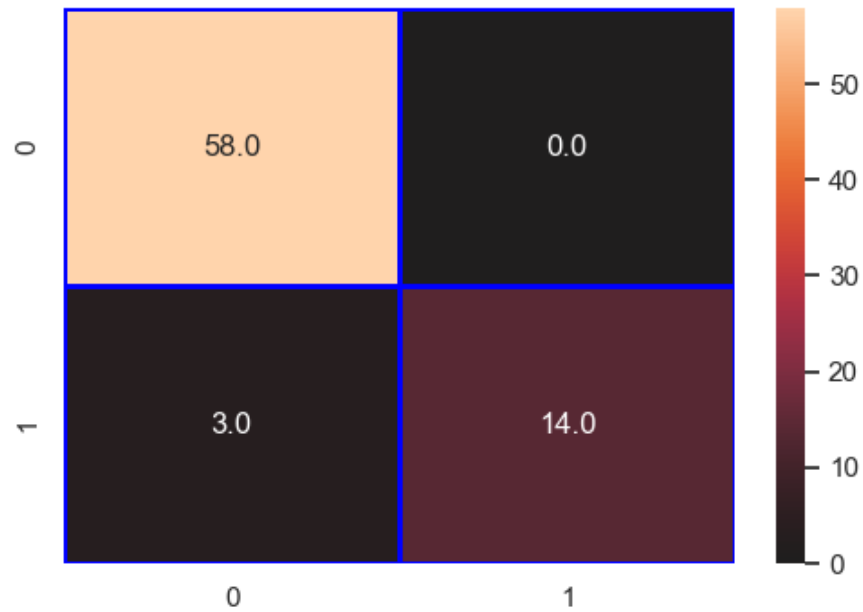
Modele name : RandomForestClassifier
Scaler name : QuantileTransformer
10 K-Fold Accuracy_score : [0.9667 0.9333 0.9667 0.8667 0.9    0.9667 0.9    0.9667 1.    0.9333]
10 K-Fold Average Accuracy_score : 94.0 %
Accuracy_score: 96.0 %
Loss: 4.0 %
Cohen_kappa_score: 87.83 %
Classification_report:
      precision    recall  f1-score   support

     0       0.95      1.00      0.97        58
     1       1.00      0.82      0.90        17

   accuracy          0.96          75
  macro avg       0.98      0.91      0.94          75
 weighted avg       0.96      0.96      0.96          75

confusion_matrix:
[[58  0]
 [ 3 14]]

```



```

=====

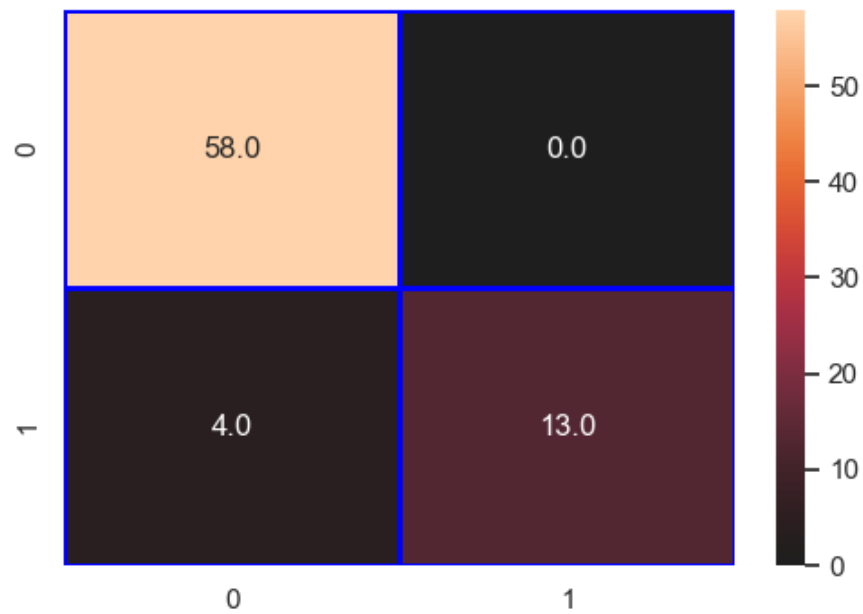
Modele name : RandomForestClassifier
Scaler name : PowerTransformer
10 K-Fold Accuracy_score : [1.      0.9333 0.9667 0.8667 0.9      0.9667 0.9      0.9333 1.      0.9333]
10 K-Fold Average Accuracy_score : 94.0 %
Accuracy_score: 94.67 %
Loss: 5.33 %
Cohen_kappa_score: 83.41 %
Classification_report:
      precision    recall  f1-score   support

     0       0.94      1.00      0.97        58
     1       1.00      0.76      0.87        17

   accuracy       0.95      0.95      0.94        75
  macro avg       0.97      0.88      0.92        75
 weighted avg       0.95      0.95      0.94        75

confusion_matrix:
[[58  0]
 [ 4 13]]

```



```

=====

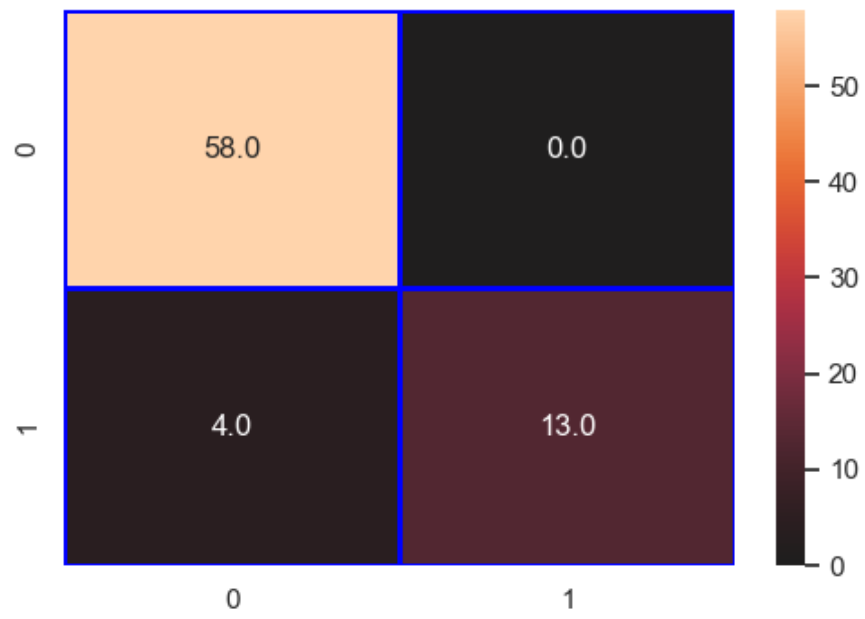
Modele name : RandomForestClassifier
Scaler name : Normalizer
10 K-Fold Accuracy_score : [0.9667 0.9333 0.9667 0.8667 0.9333 0.9667 0.9   0.9333 1.    0.9333]
10 K-Fold Average Accuracy_score : 94.0 %
Accuracy_score: 94.67 %
Loss: 5.33 %
Cohen_kappa_score: 83.41 %
Classification_report:
      precision    recall  f1-score   support

     0       0.94      1.00      0.97        58
     1       1.00      0.76      0.87        17

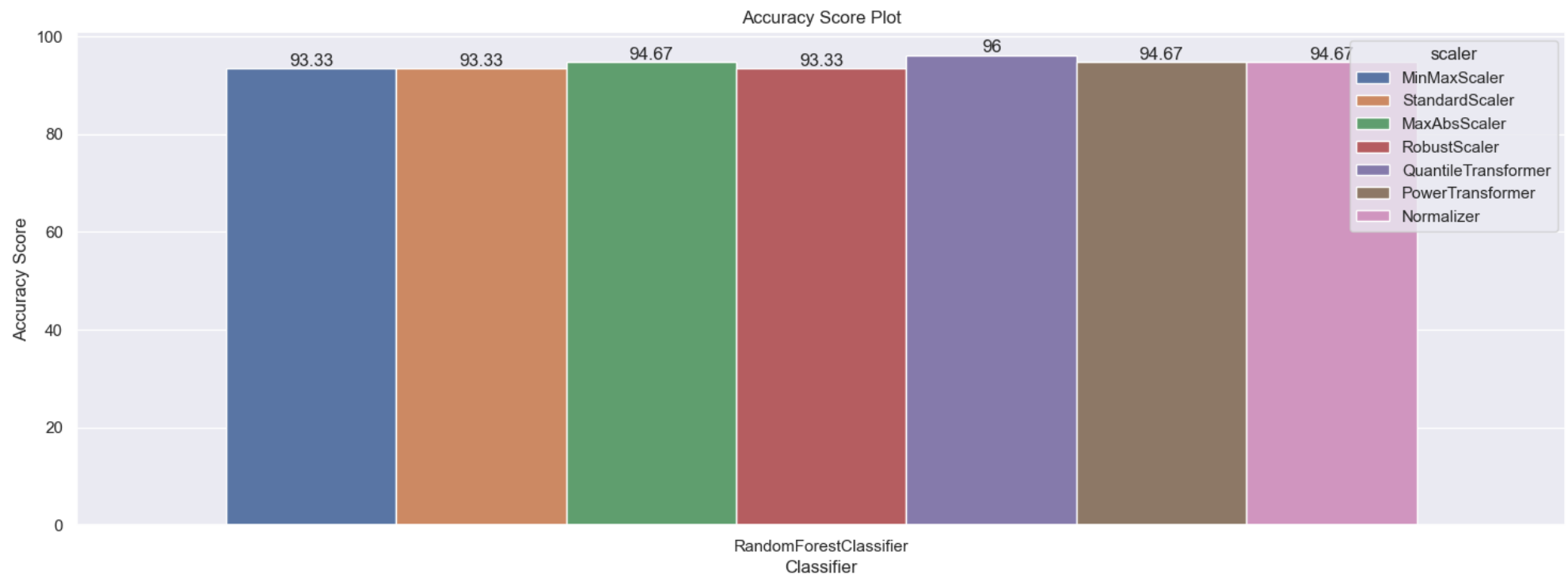
   accuracy          0.95          75
  macro avg       0.97      0.88      0.92          75
 weighted avg       0.95      0.95      0.94          75

confusion_matrix:
[[58  0]
 [ 4 13]]

```



=====



Done...

In []:

1