In [1]:

```
import numpy as np
import pandas as pd
from collections import Counter

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

# Root Mean Square Error
from sklearn.metrics import mean_squared_error
from math import sqrt

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

plot_data_list = []
```

```
plot_RSME_list = []
```

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

Out[3]: (375, 35)

In [4]:
```
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 
family income                  375 non-null    object 
 22  Pressure of 
 working                  375 non-null    object 
 23  Frequent 
travel                     375 non-null    object 
 24  Monotonus 
 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]:
```
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]:

```python
# check null
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
Monotonus\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]:

```
# list of numerical variables
numerical_features = [feature for feature in df.columns if df[feature].dtypes != 'O']

print('Number of numerical variables: ', len(numerical_features))

numerical_variables_df = df[numerical_features]

# visualise the numerical variables
df[numerical_features].head()
```

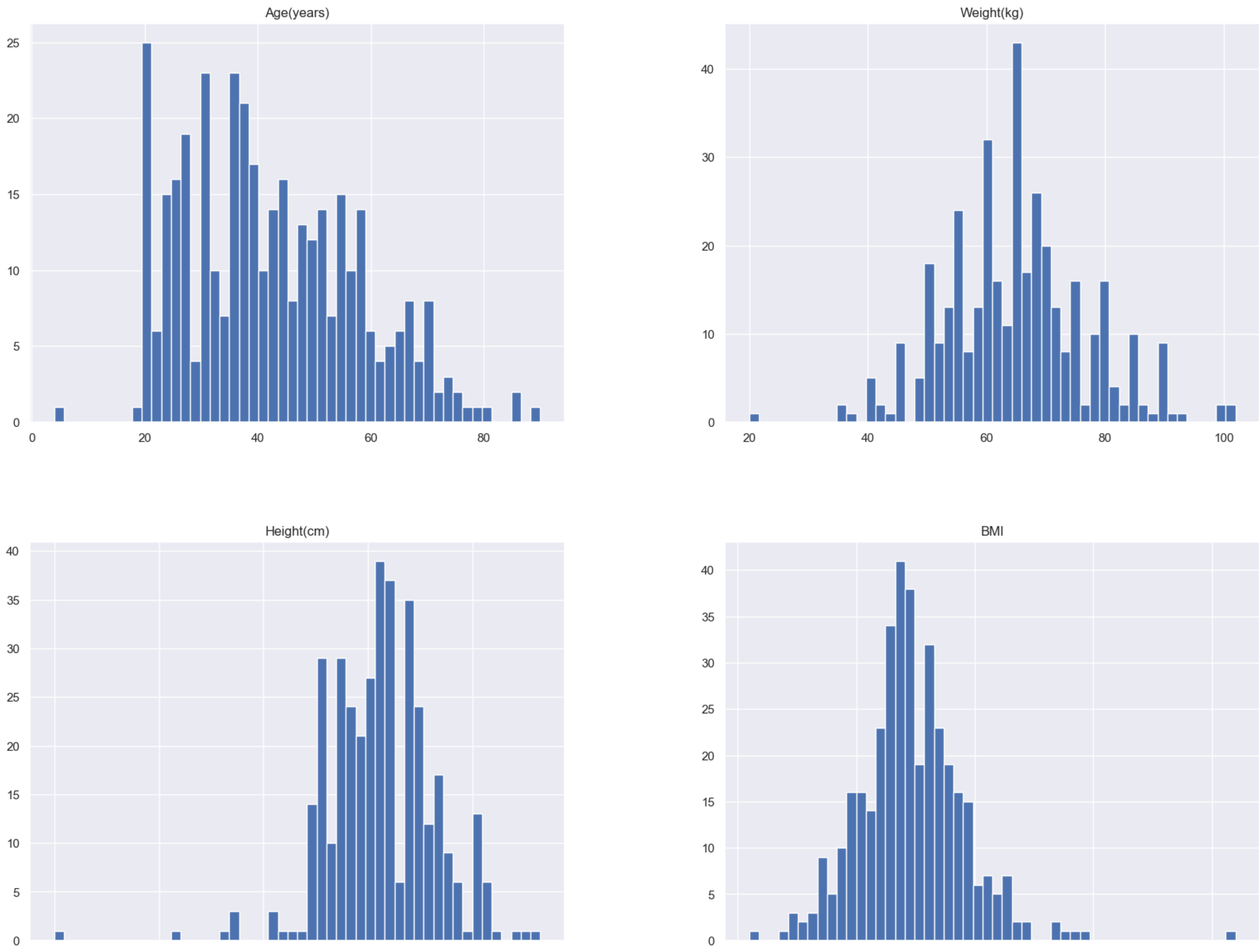
```
Number of numerical variables:  4
```

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]:

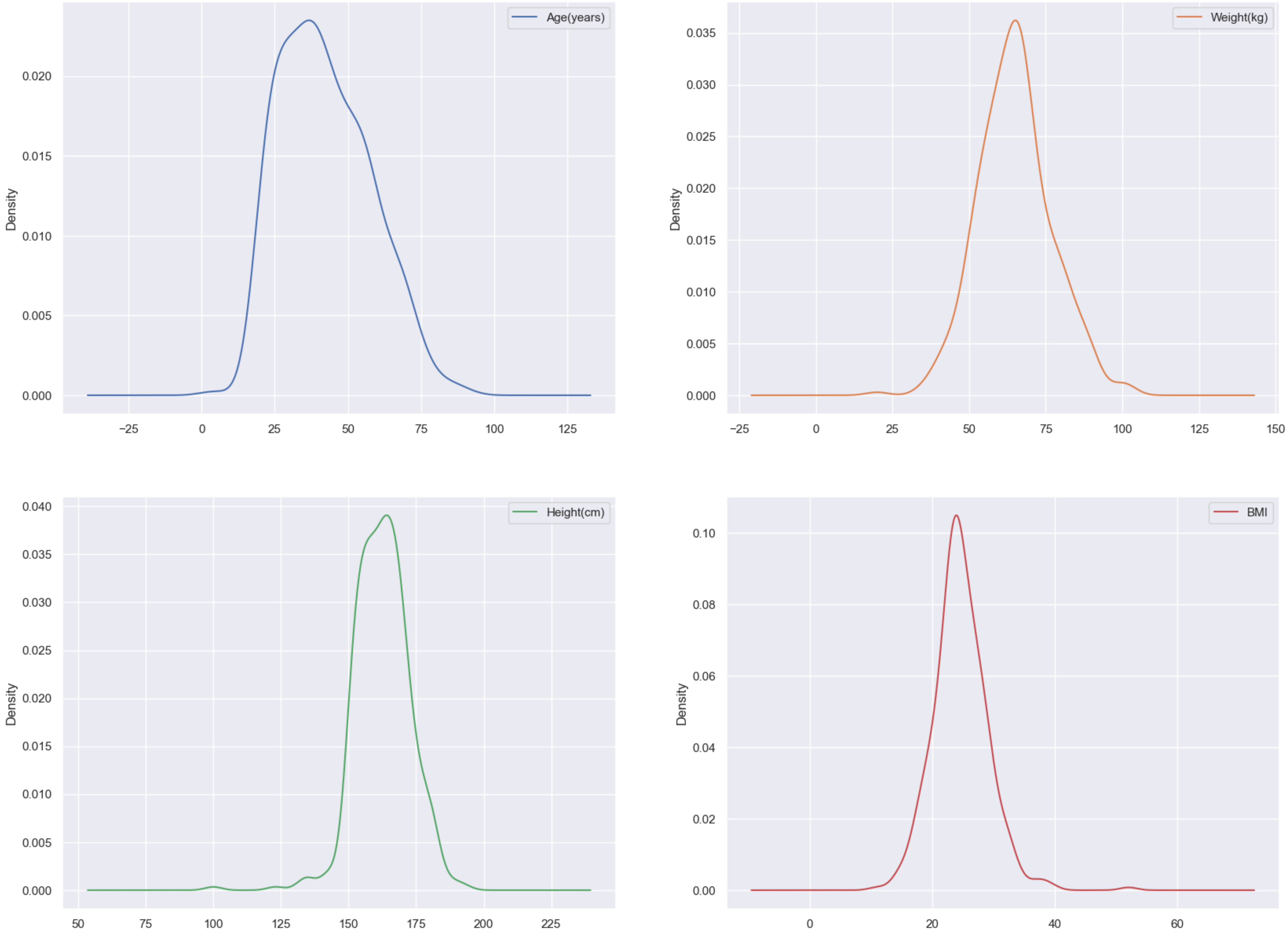
```
# Histogram:distribution of a single numerical variable
numerical_variables_df.hist(bins=50, figsize=(20, 15))
plt.show()
```


Age(years)
0
5
10
15
20
25
0
20
40
60
80
Weight(kg)
0
10
20
30
40
20
40
60
80
100
Height(cm)
0
5
10
15
20
25
30
35
40
BMI
0
5
10
15
20
25
30
35
40

100 120 140 160 180 10 20 30 40 50

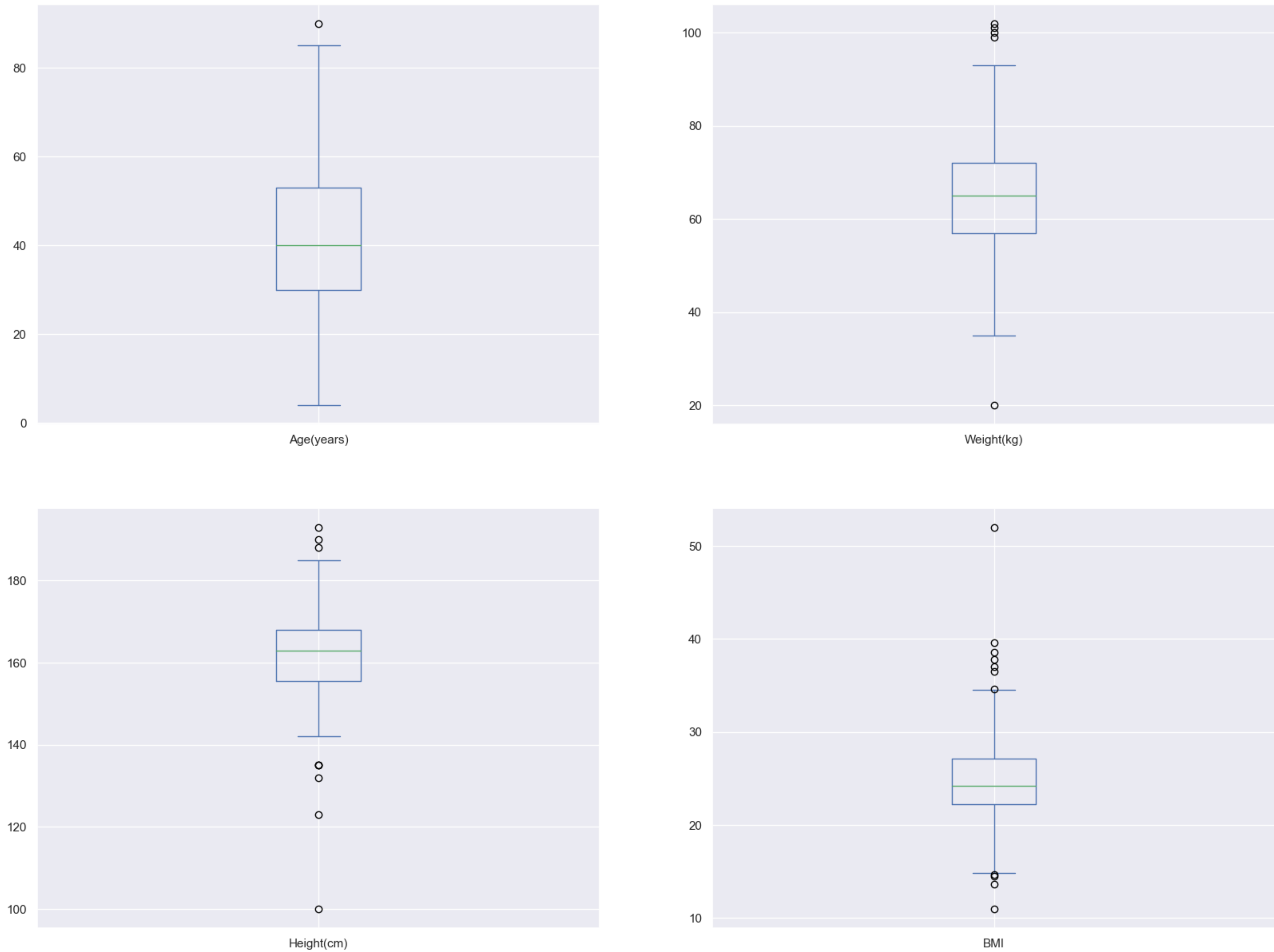
In [9]:

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


Age(years)
Weight(kg)
Height(cm)
BMI
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
0.00
0.02
0.04
0.06
0.08
0.10
−25
0
25
50
75
100
125
150
175
200
225
20
40
60

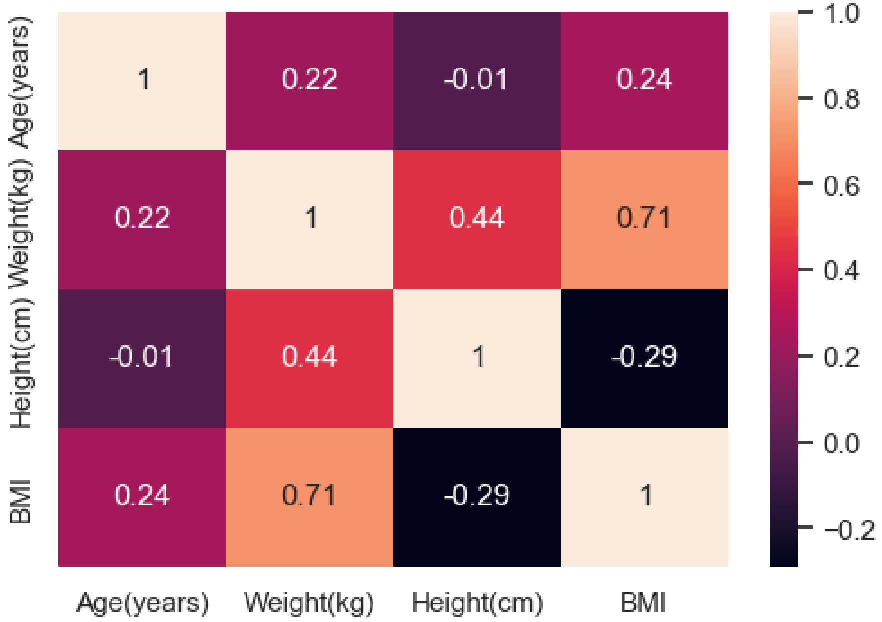
In [10]:

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


0
20
40
60
80
Age(years)
20
40
60
80
100
Weight(kg)
100
120
140
160
180
Height(cm)
10
20
30
40
50
BMI

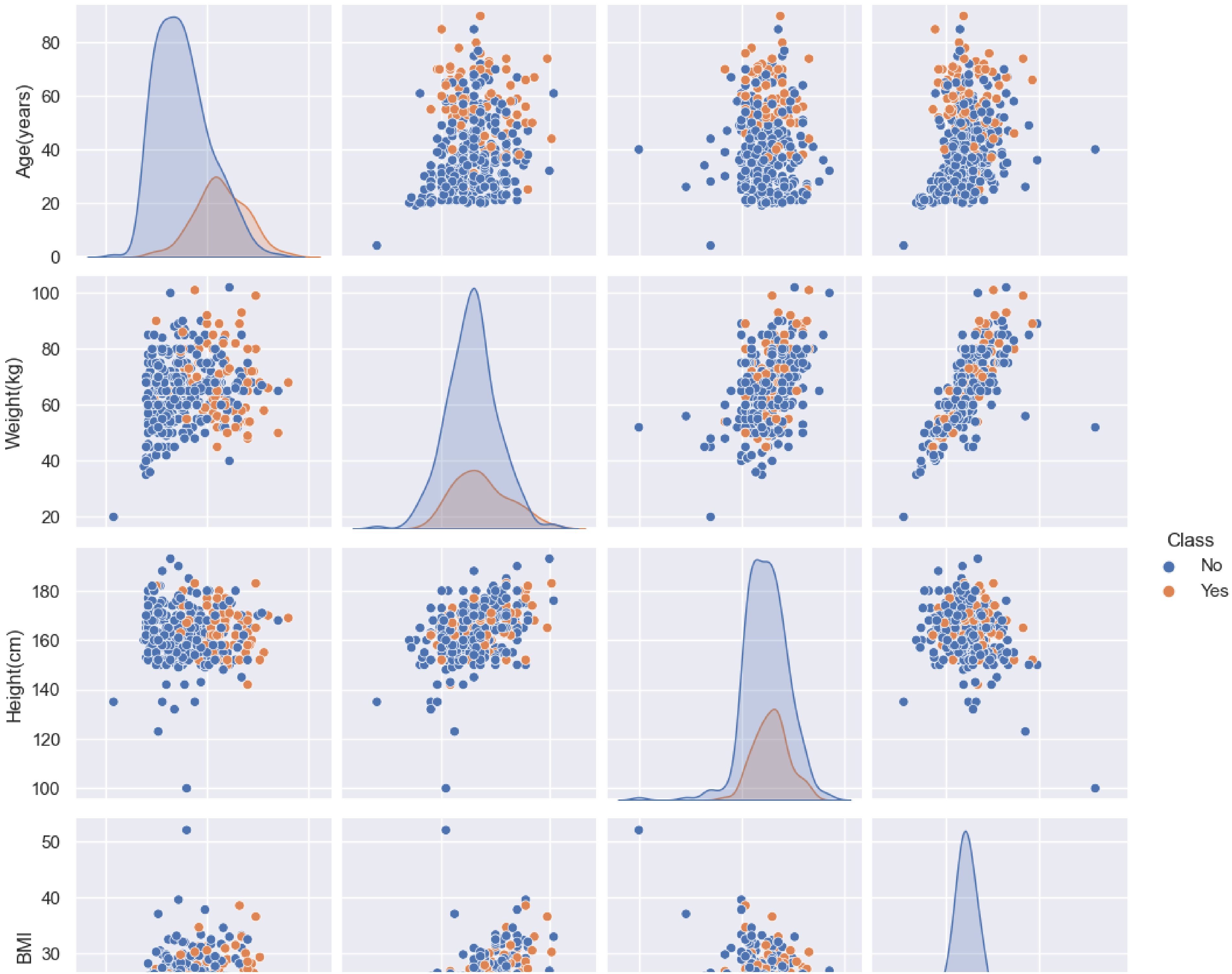
observation : outlier is detected in all features

In [11]:
```
# Correlation between the different characteristics. Closer to 1 better is the correlation.
sns.heatmap(round(numerical_variables_df.corr(method='pearson'),2), annot = True)
plt.show()
```



In [12]:

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


Age(years)
Weight(kg)
Height(cm)
BMI
Class
No
Yes

In [13]:

```python
# list of categorical variables
categorical_features = [feature for feature in df.columns if df[feature].dtypes == 'O']

print('Number of categorical variables: ', len(categorical_features))

categorical_variables_df = df[categorical_features]

# visualise the numerical variables
df[categorical_features].head()
```

```
Number of categorical variables:  31
```

Out[13]:

	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	H
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 [14]:

```
for col in categorical_variables_df.columns:
    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', 'other'],           Cou
nt : 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', 'Rar
ely', 'Rarely (Weekly)', 'Sometimes'],           Count : 9

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

Unique values in grains is : ['Brown-Rice', 'Brown-Rice-Millets', 'Brown-Rice-Millets-Maida', 'Brown-Rice-Wheat', 'Brown-Rice-Wheat-Millets', 'Mi
llets', 'Polished-Rice', 'Polished-Rice-Brown-Rice-Wheat', 'Polished-Rice-Brown-Rice-Wheat-Millets', 'Polished-Rice-Maida', 'Polished-Rice-Millet
s', 'Polished-Rice-Wheat', 'Polished-Rice-Wheat-Maida-Millets', 'Polished-Rice-Wheat-Millets', 'Rice-Wheat', 'Rice-Wheat-Brown-Rice', 'Rice-Wheat
-Brown-Rice-Maida', 'Rice-Wheat-Maida', 'Rice-Wheat-Millets', 'Rice-Wheat-Millets-Maida', 'Wheat', 'Wheat-Brown-Rice', 'Wheat-Brown-Rice-Millet
s', 'Wheat-Maida', 'Wheat-Millets', 'Wheat-Millets-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 Monotonus
 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 [15]:

```python
replacement_dict = {'other': 'Other'}
categorical_variables_df['Profession'].replace(replacement_dict, inplace=True)

replacement_dict = {'YES': 'Yes', 'yes': 'Yes'}
categorical_variables_df['smoke'].replace(replacement_dict, inplace=True)

replacement_dict = {'Thrice in a day': 'Thrice'}
categorical_variables_df['cereal'].replace(replacement_dict, inplace=True)

replacement_dict = {'None': 'Never', 'Rarely (Weekly)': 'Rarely', 'Sometimes': 'Rarely'}
categorical_variables_df['salad'].replace(replacement_dict, inplace=True)

replacement_dict = {'None': 'Never', 'Once in a week': 'Once', 'Sometimes': 'Rarely', 'Twice a day': 'Twice'}
categorical_variables_df['vegetables'].replace(replacement_dict, inplace=True)

replacement_dict = {' Low': 'Low', 'Very High': 'Very high'}
categorical_variables_df['Stress_Workload'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Poor \nfamily income'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Pressure of\n working'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Frequent \ntravel'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Monotonus\n work'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Poor and unplanned work'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Lack of career development'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Feeling of powerlessness'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Lack of financial security'].replace(replacement_dict, inplace=True)

replacement_dict = {'Very High': 'Very high'}
categorical_variables_df['Unable to satisfy all stakeholders'].replace(replacement_dict, inplace=True)

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

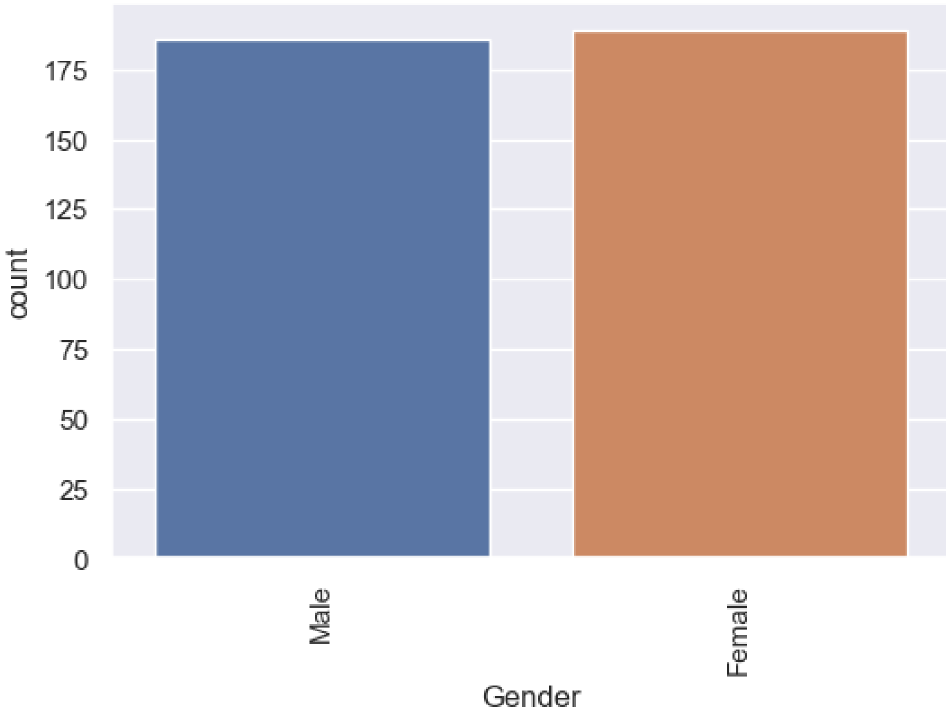
```
        Count : {len(sorted(categorical_variables_df[col].unique()))} \n")
```

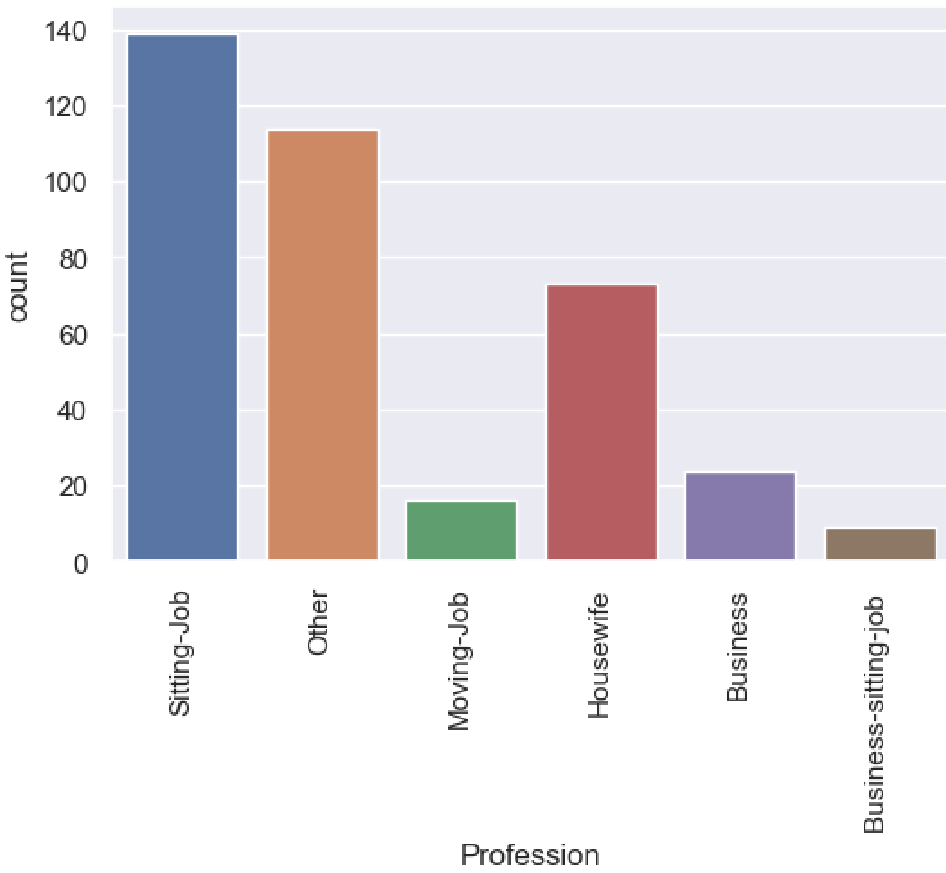
```
Unique values in Gender is : ['Female', 'Male'],           Count : 2

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-Millets', 'Brown-Rice-Millets-Maida', 'Brown-Rice-Wheat', 'Brown-Rice-Wheat-Millets', 'Mi
llets', 'Polished-Rice', 'Polished-Rice-Brown-Rice-Wheat', 'Polished-Rice-Brown-Rice-Wheat-Millets', 'Polished-Rice-Maida', 'Polished-Rice-Millet
s', 'Polished-Rice-Wheat', 'Polished-Rice-Wheat-Maida-Millets', 'Polished-Rice-Wheat-Millets', 'Rice-Wheat', 'Rice-Wheat-Brown-Rice', 'Rice-Wheat
-Brown-Rice-Maida', 'Rice-Wheat-Maida', 'Rice-Wheat-Millets', 'Rice-Wheat-Millets-Maida', 'Wheat', 'Wheat-Brown-Rice', 'Wheat-Brown-Rice-Millet
s', 'Wheat-Maida', 'Wheat-Millets', 'Wheat-Millets-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
```

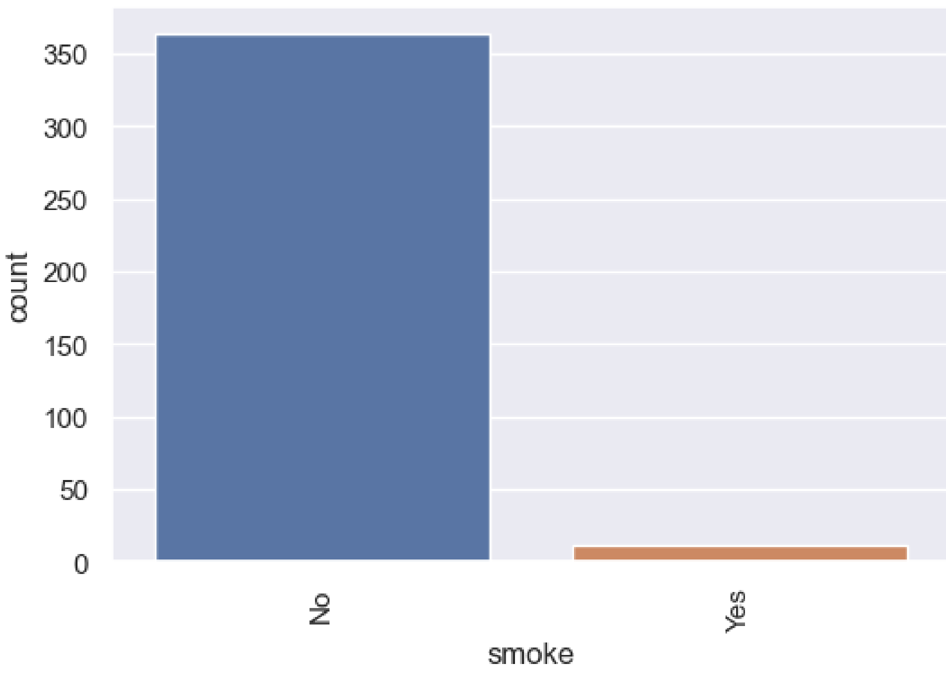
```
work is : ['Average', 'High', 'Low', 'Very high', 'Very low'],          Count : 5

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
```

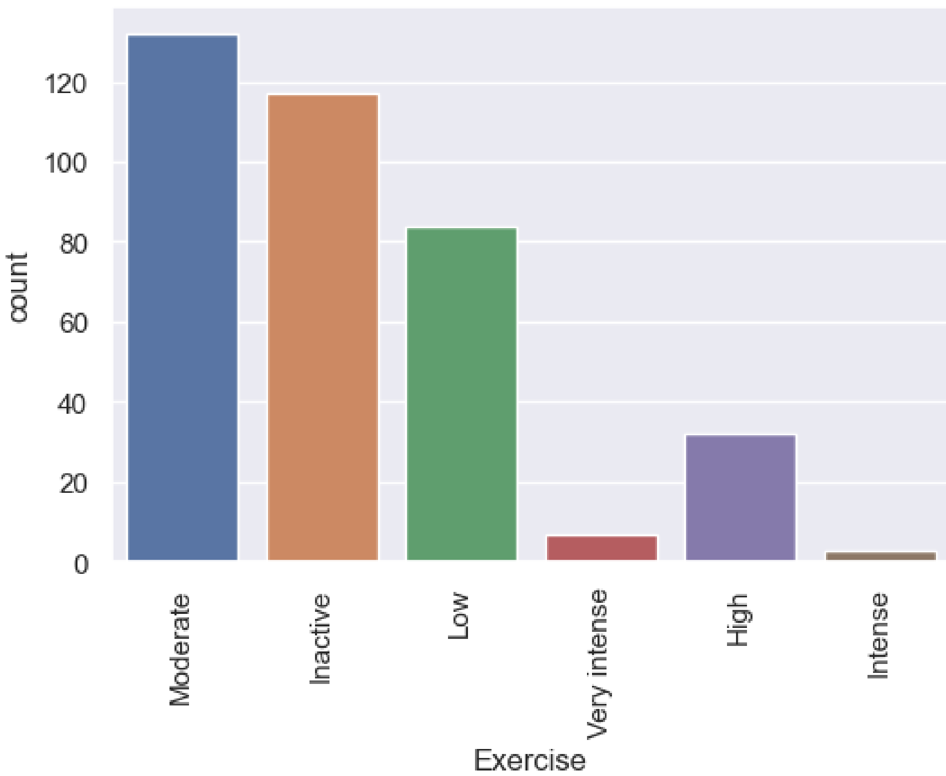
In [16]:

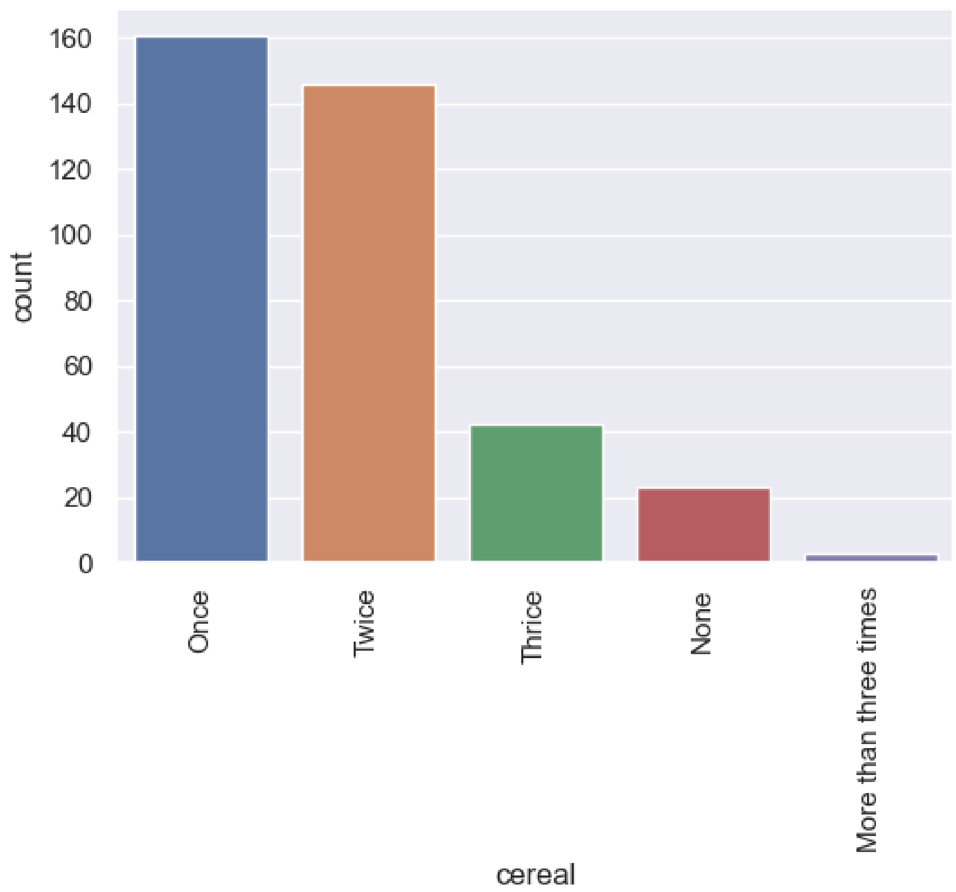
```
# Count Plot: Displays the count of occurrences for each category as bars.
for col in categorical_variables_df.columns:
    plt.xticks(rotation=90)
    sns.countplot(data=categorical_variables_df, x=col)
    plt.show()
```

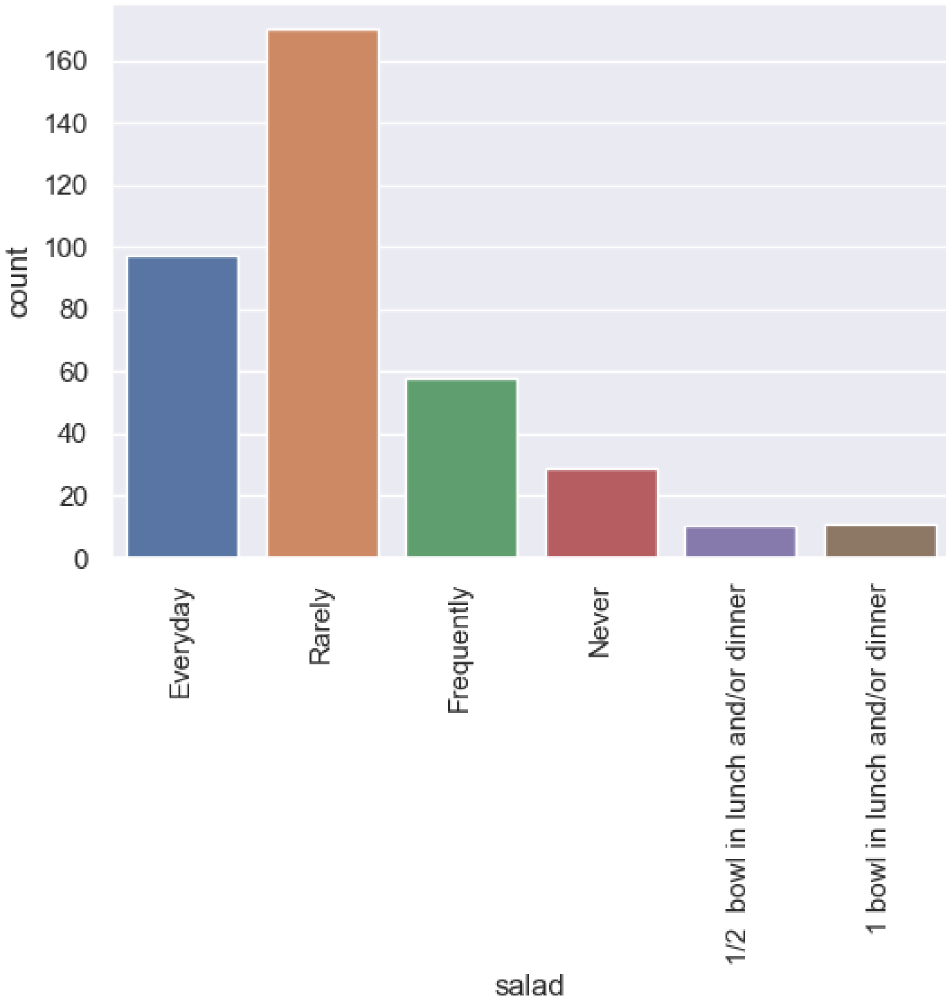


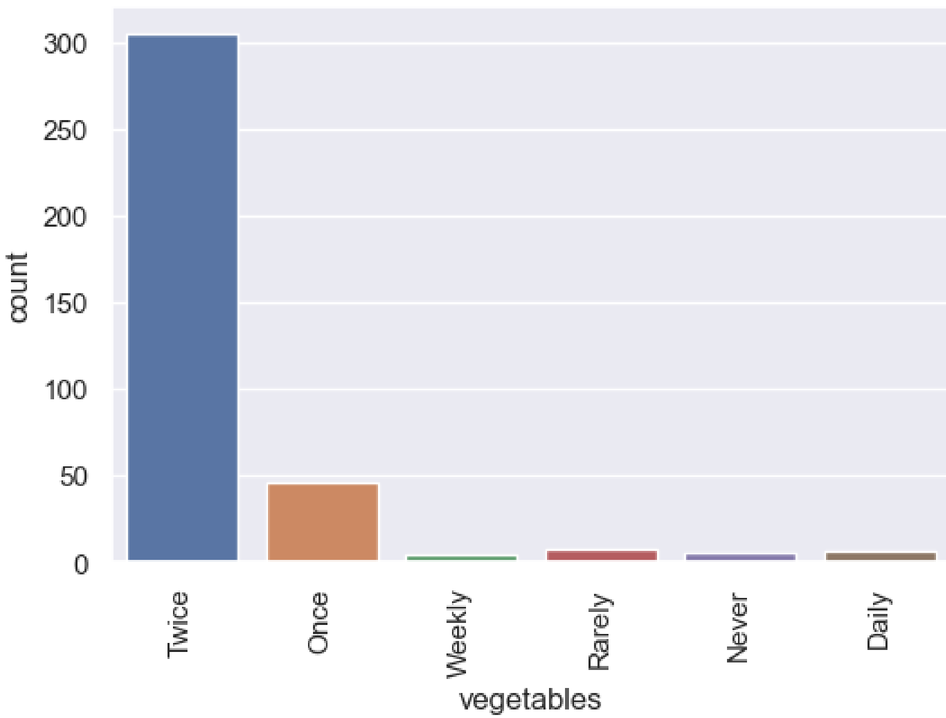

140
120
100
80
60
40
20
0
count
Sitting-Job
Other
Moving-Job
Housewife
Business
Business-sitting-job
Profession

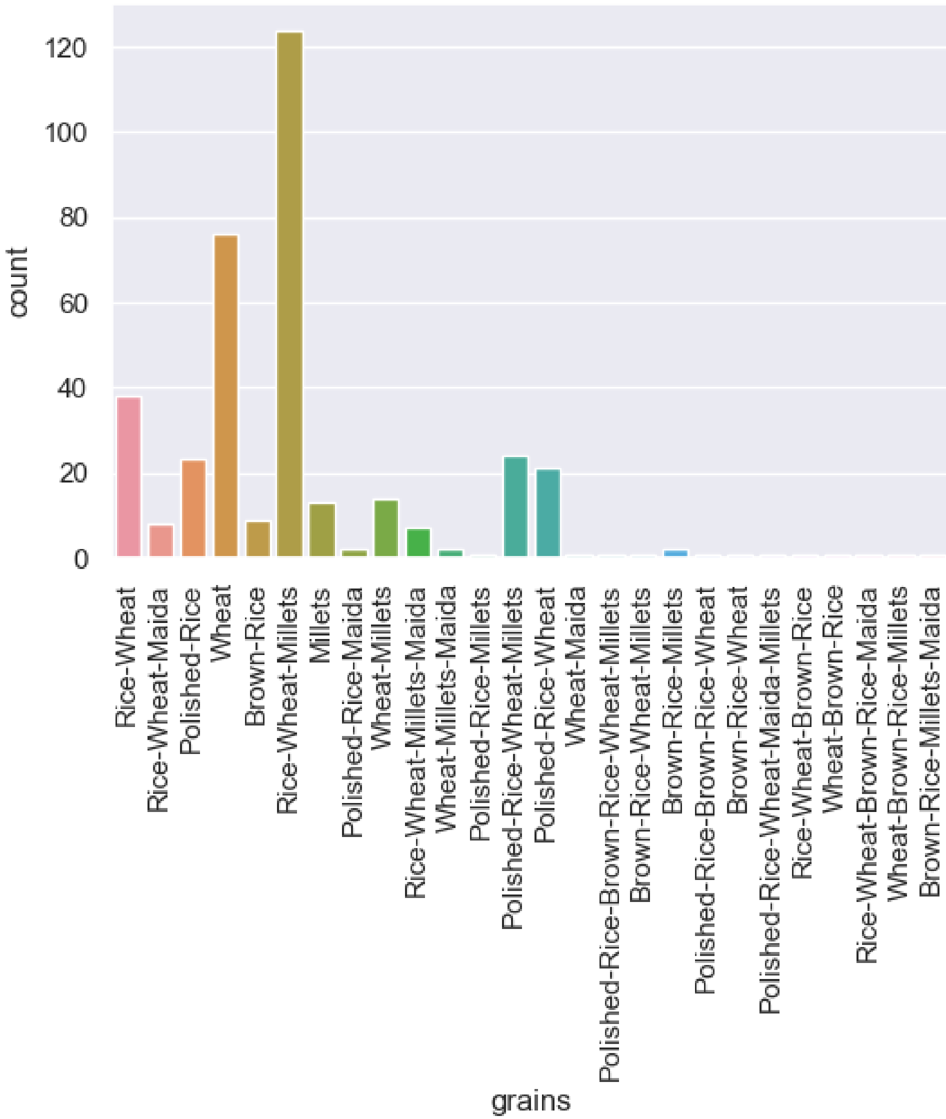

count
350
300
250
200
150
100
50
0
No
Yes
smoke

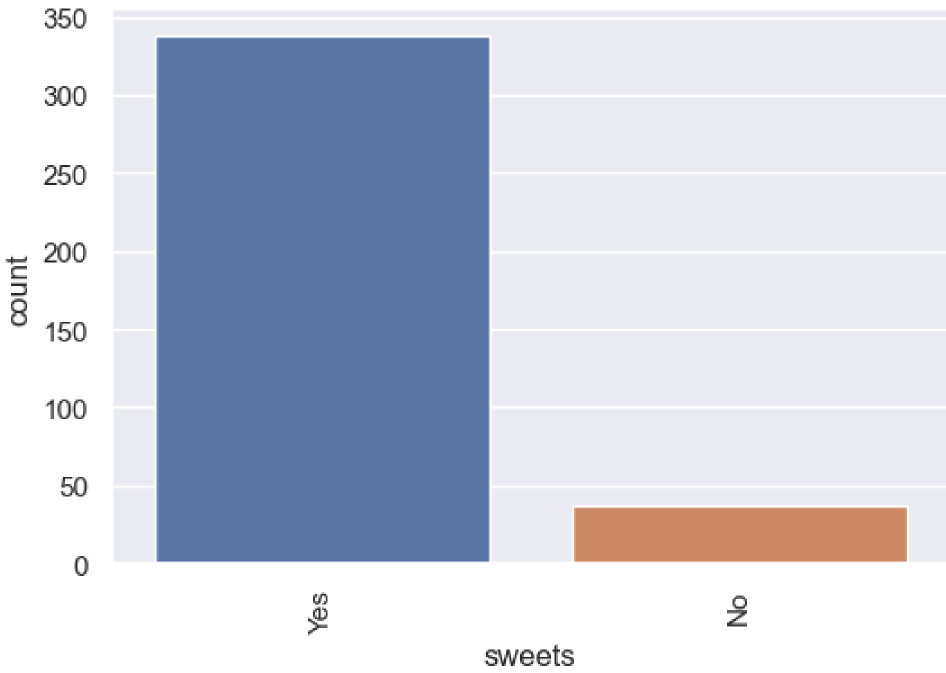

count
0
20
40
60
80
100
120
Moderate
Inactive
Low
Very intense
High
Intense
Exercise

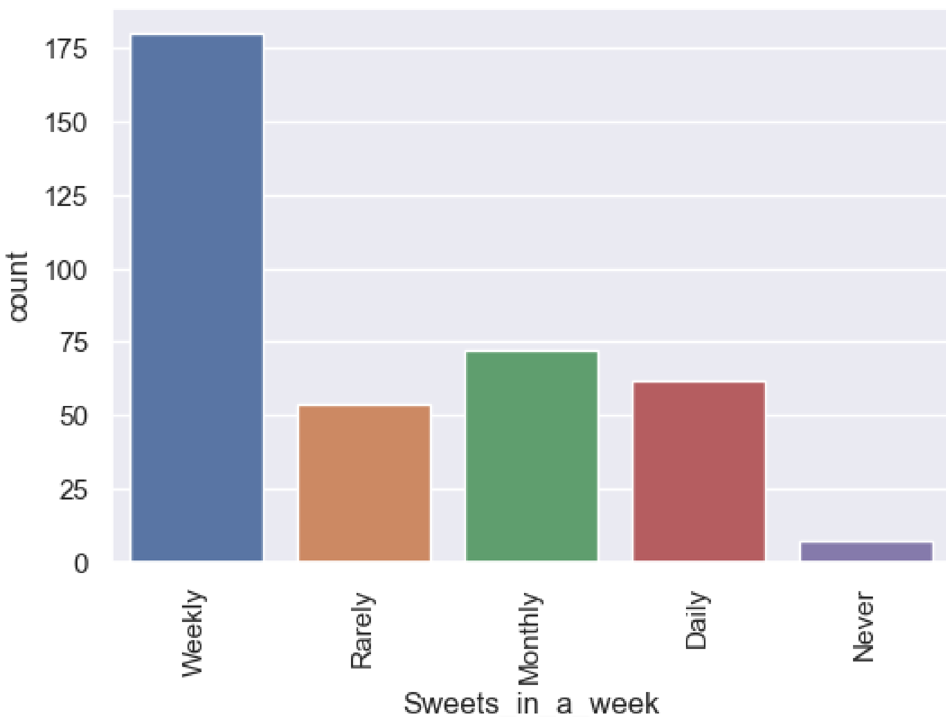

160
140
120
100
80
60
40
20
0
count
Once
Twice
Thrice
None
More than three times
cereal

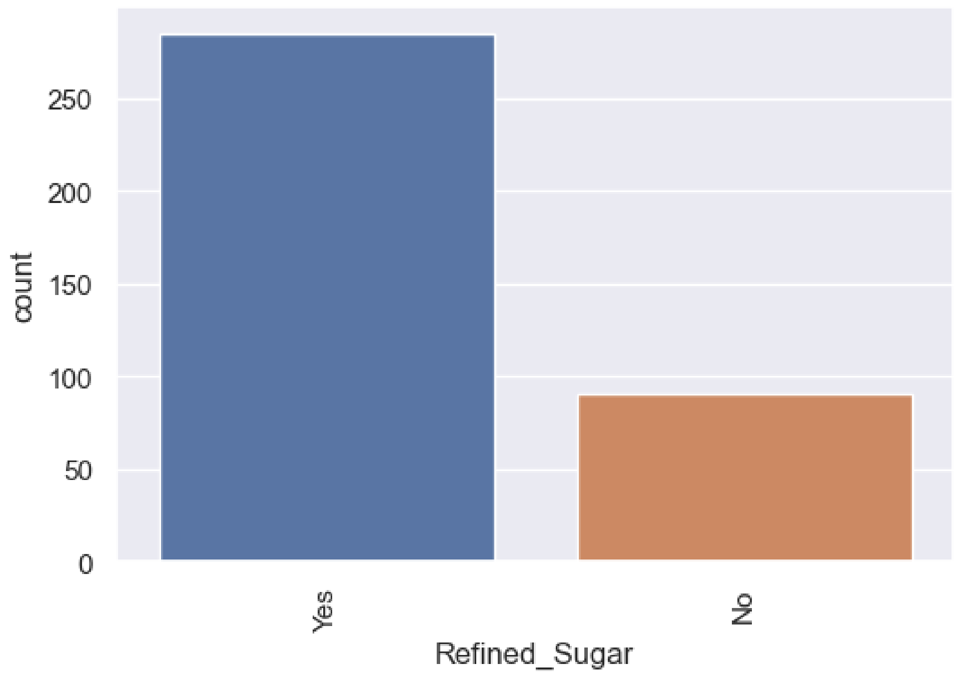

count
0
20
40
60
80
100
120
140
160
Everyday
Rarely
Frequently
Never
1/2 bowl in lunch and/or dinner
1 bowl in lunch and/or dinner
salad

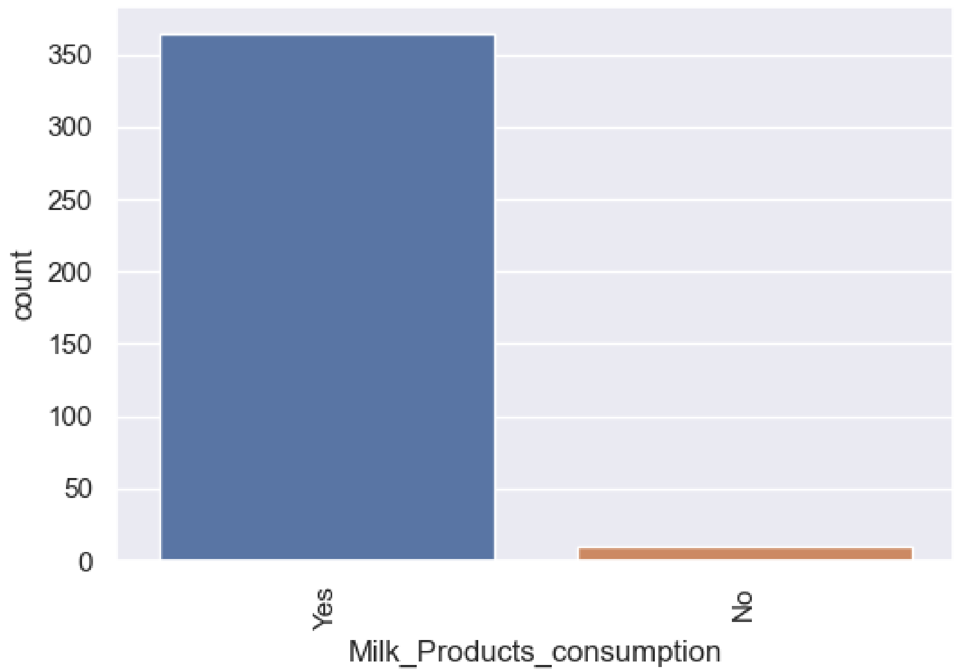

count
300
250
200
150
100
50
0
Twice
Once
Weekly
Rarely
Never
Daily
vegetables

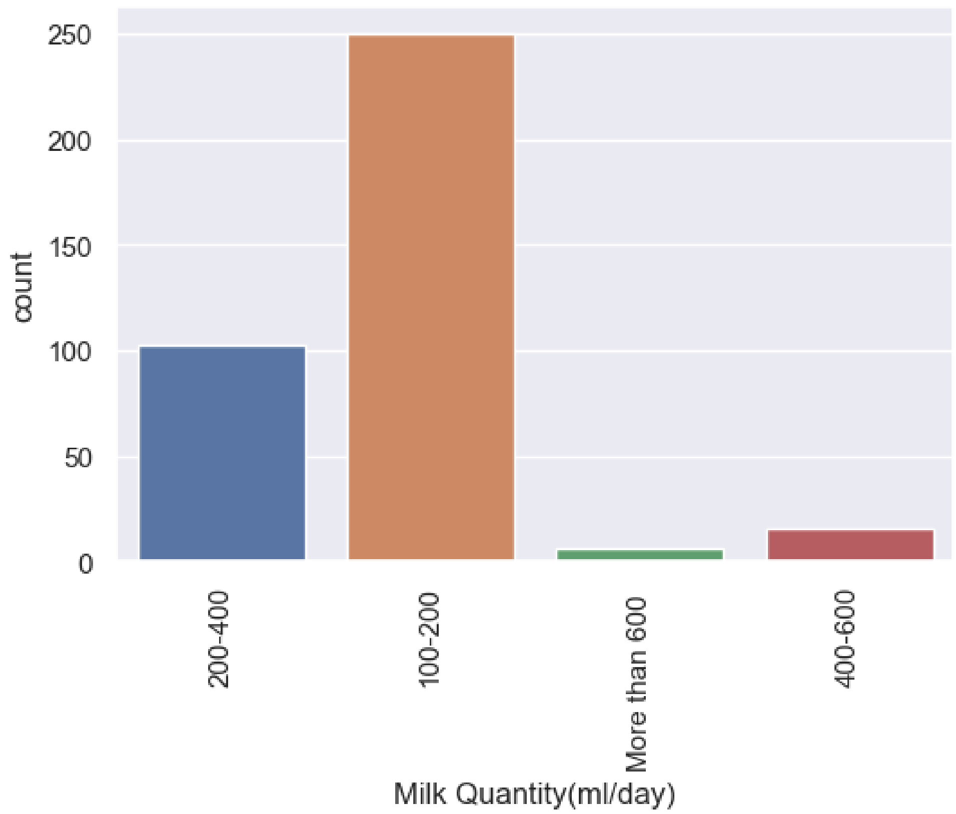

count
0
20
40
60
80
100
120
Rice-Wheat
Rice-Wheat-Maida
Polished-Rice
Wheat
Brown-Rice
Rice-Wheat-Millets
Millets
Polished-Rice-Maida
Wheat-Millets
Rice-Wheat-Millets-Maida
Wheat-Millets-Maida
Polished-Rice-Millets
Polished-Rice-Wheat-Millets
Polished-Rice-Wheat
Wheat-Maida
Polished-Rice-Brown-Rice-Wheat-Millets
Brown-Rice-Wheat-Millets
Brown-Rice-Millets
Polished-Rice-Brown-Rice-Wheat
Brown-Rice-Wheat
Polished-Rice-Wheat-Maida-Millets
Rice-Wheat-Brown-Rice
Wheat-Brown-Rice
Rice-Wheat-Brown-Rice-Maida
Wheat-Brown-Rice-Millets
Brown-Rice-Millets-Maida
grains


350
300
250
200
150
100
50
0
count
Yes
No
sweets

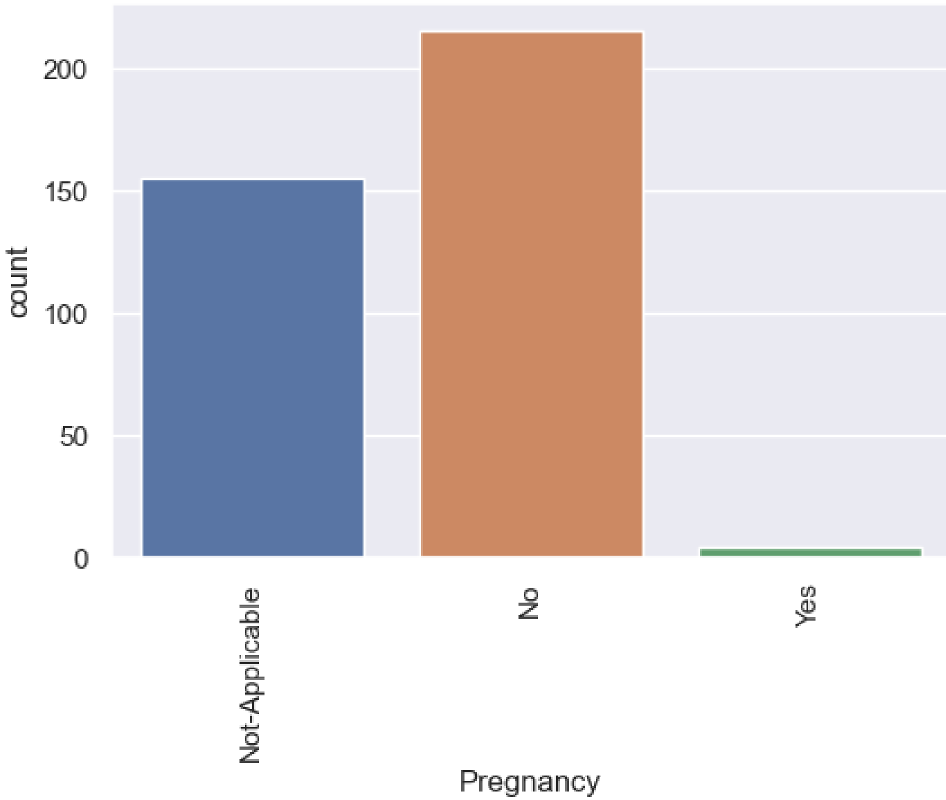

175
150
125
100
75
50
25
0
count
Weekly
Rarely
Monthly
Daily
Never
Sweets_in_a_week


count
250
200
150
100
50
0
Yes
No
Refined_Sugar

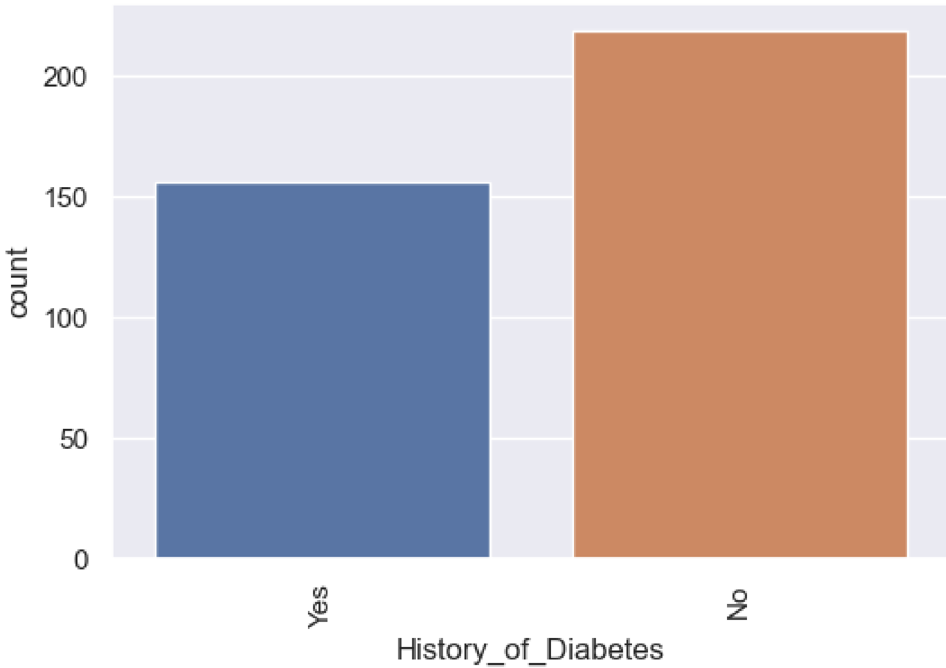

count
350
300
250
200
150
100
50
0
Yes
No
Milk_Products_consumption



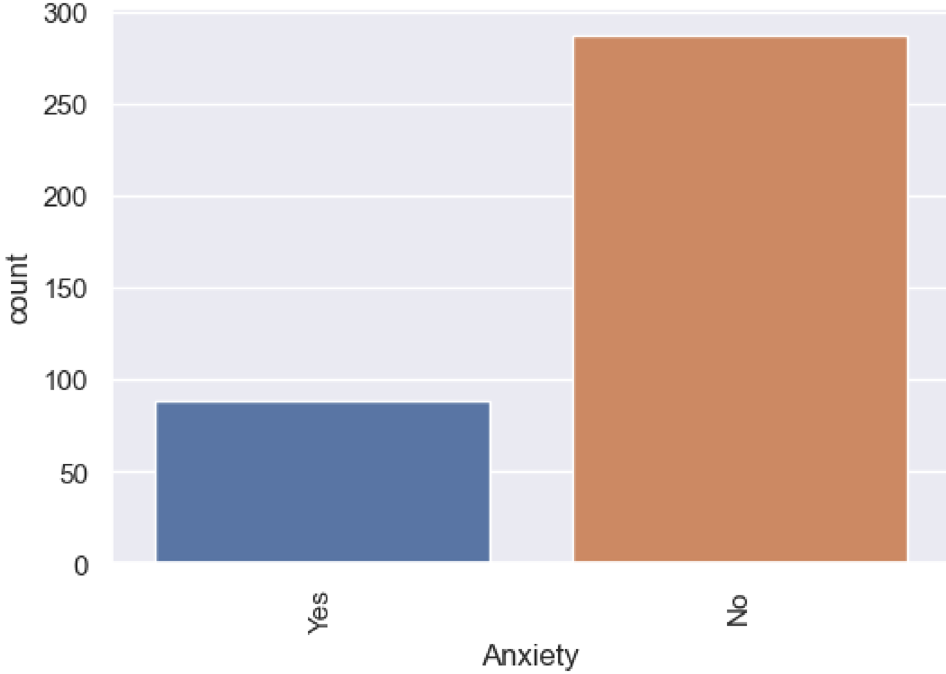
count
250
200
150
100
50
0
200-400
100-200
More than 600
400-600
Milk Quantity(ml/day)



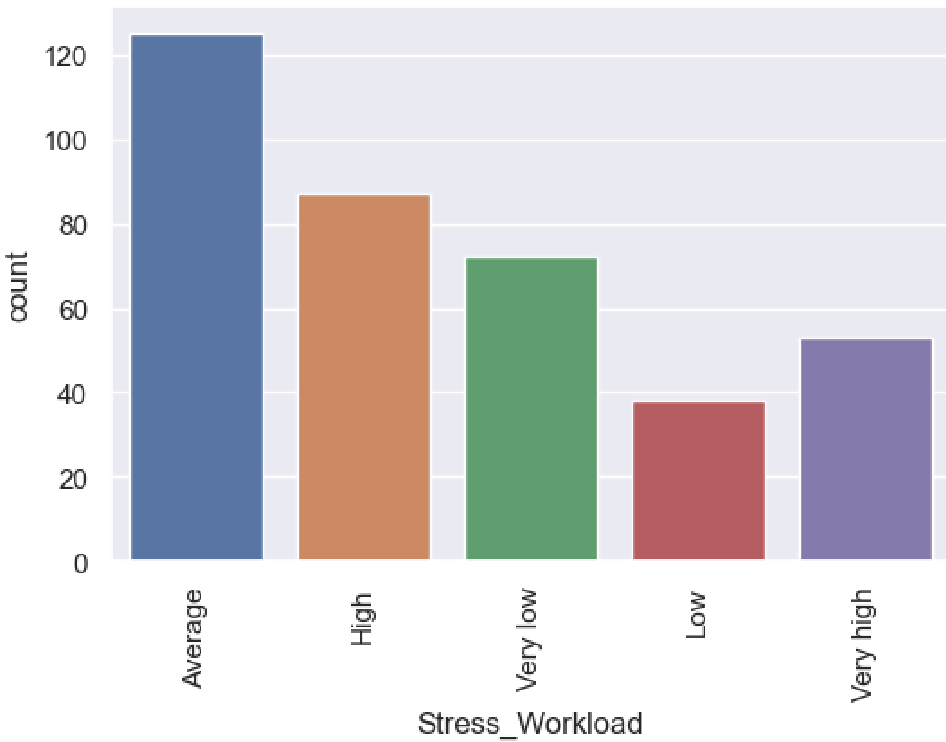
200
150
100
50
0
count
Not-Applicable
No
Yes
Pregnancy



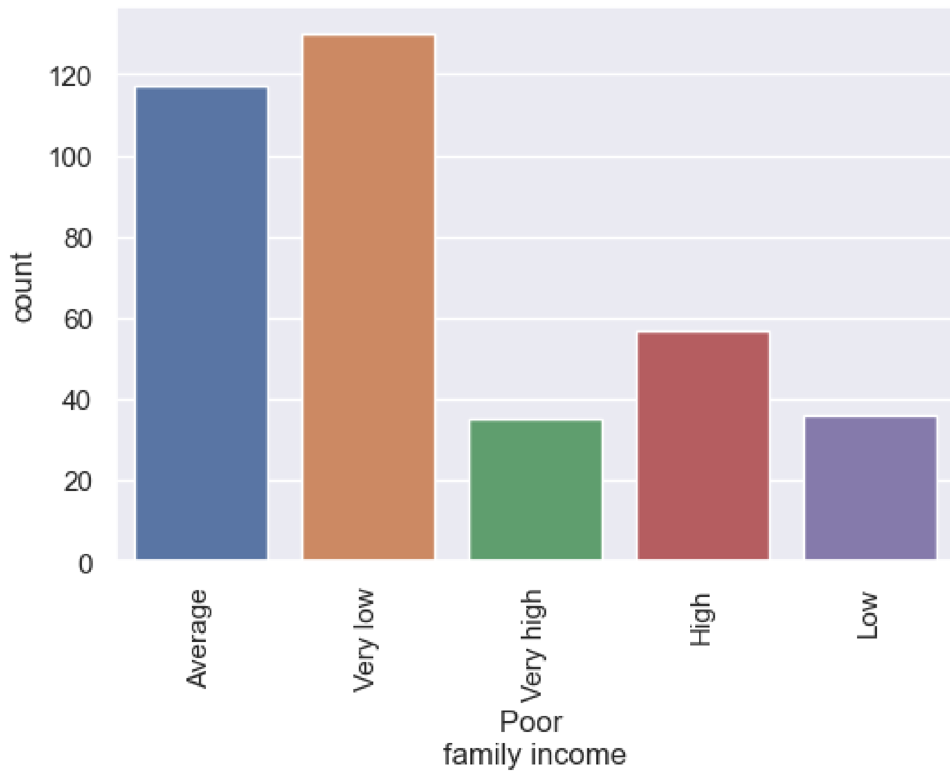
200
150
100
50
0
count
Yes
No
History_of_Diabetes

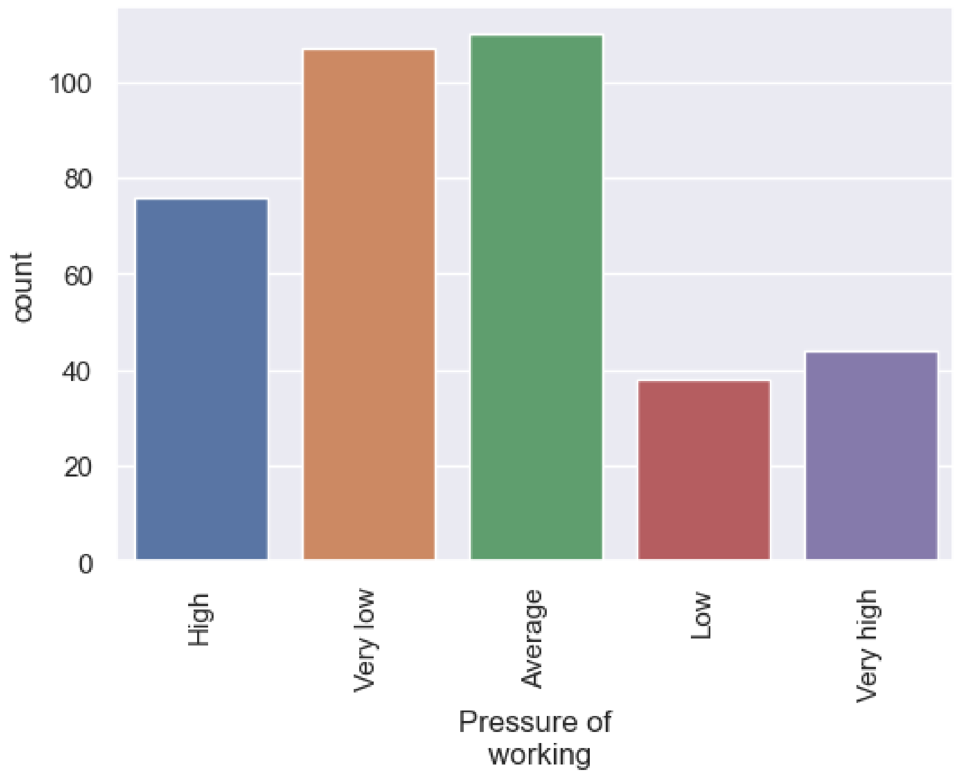


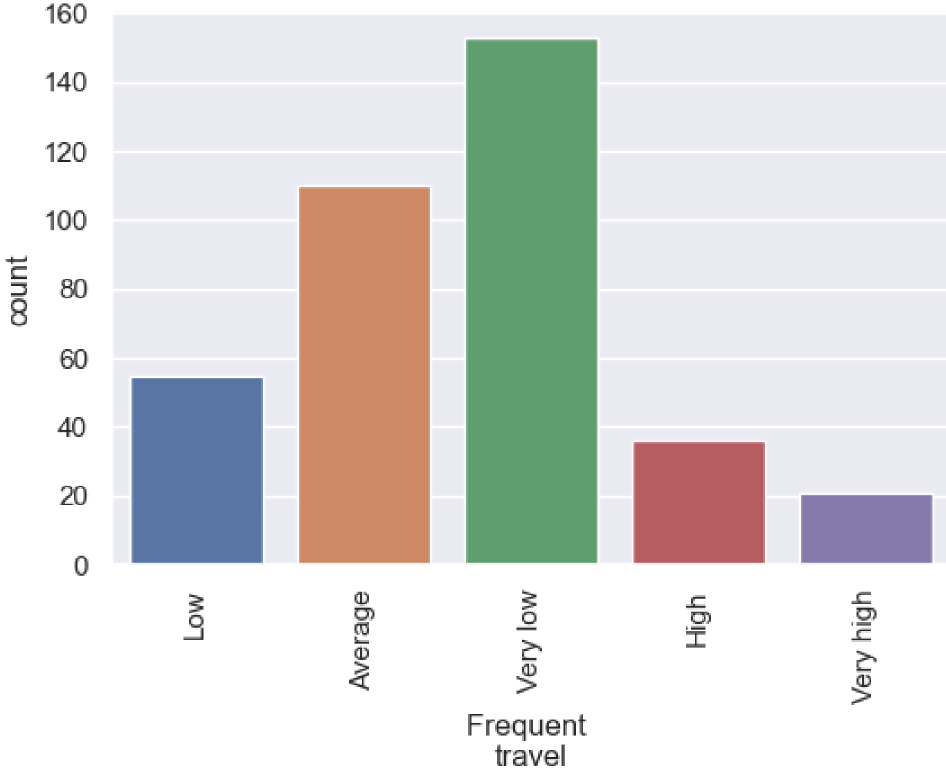
300
250
200
150
100
50
0
count
Yes
No
Anxiety

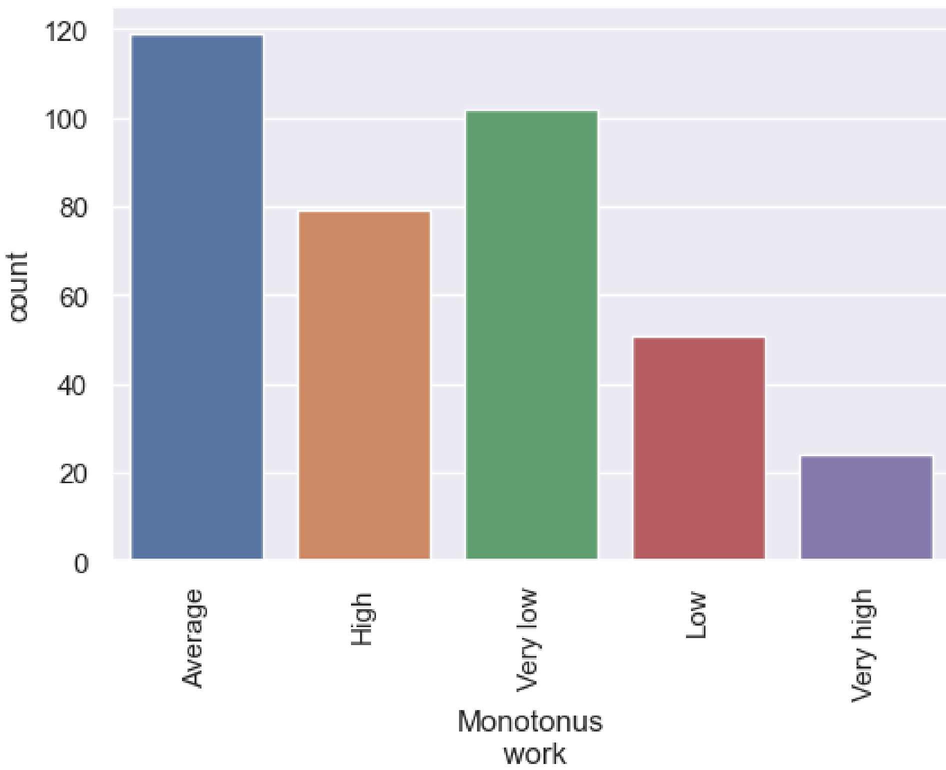


count
0
20
40
60
80
100
120
Average
High
Very low
Low
Very high
Stress_Workload

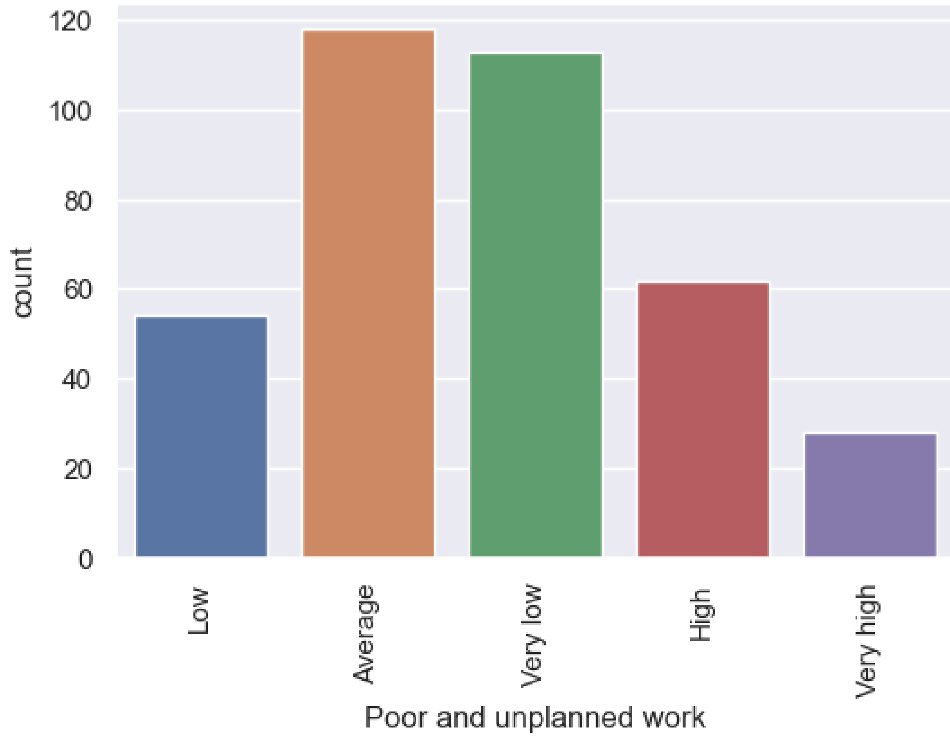

count
120
100
80
60
40
20
0
Average
Very low
Very high
High
Low
Poor
family income


count
100
80
60
40
20
0
High
Very low
Average
Low
Very high
Pressure of working

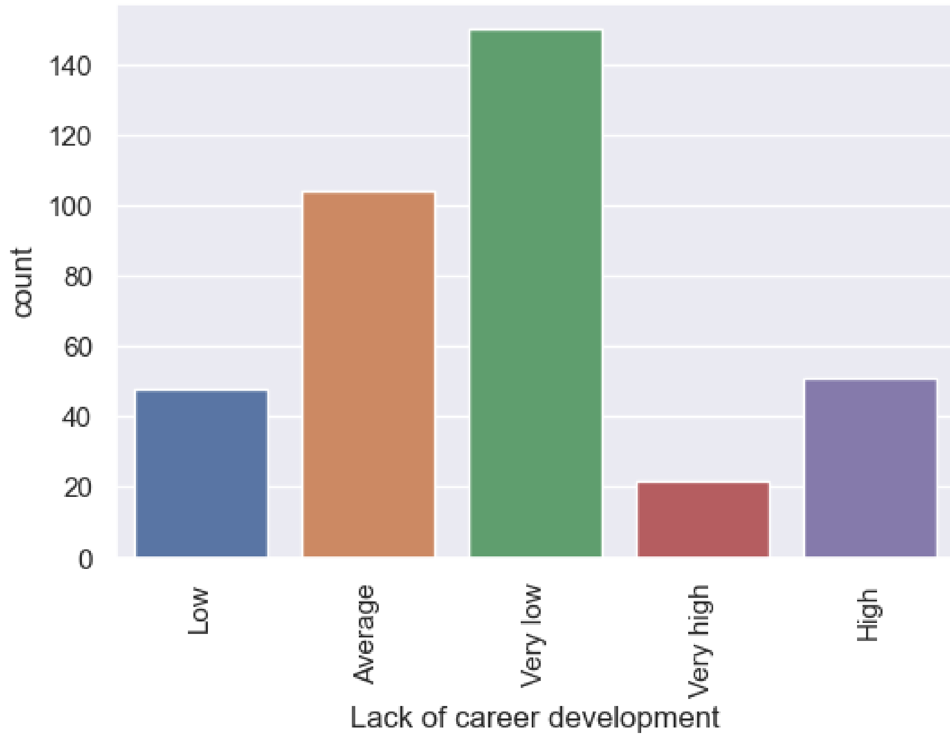

160
140
120
100
80
60
40
20
0
count
Low
Average
Very low
High
Very high
Frequent
travel

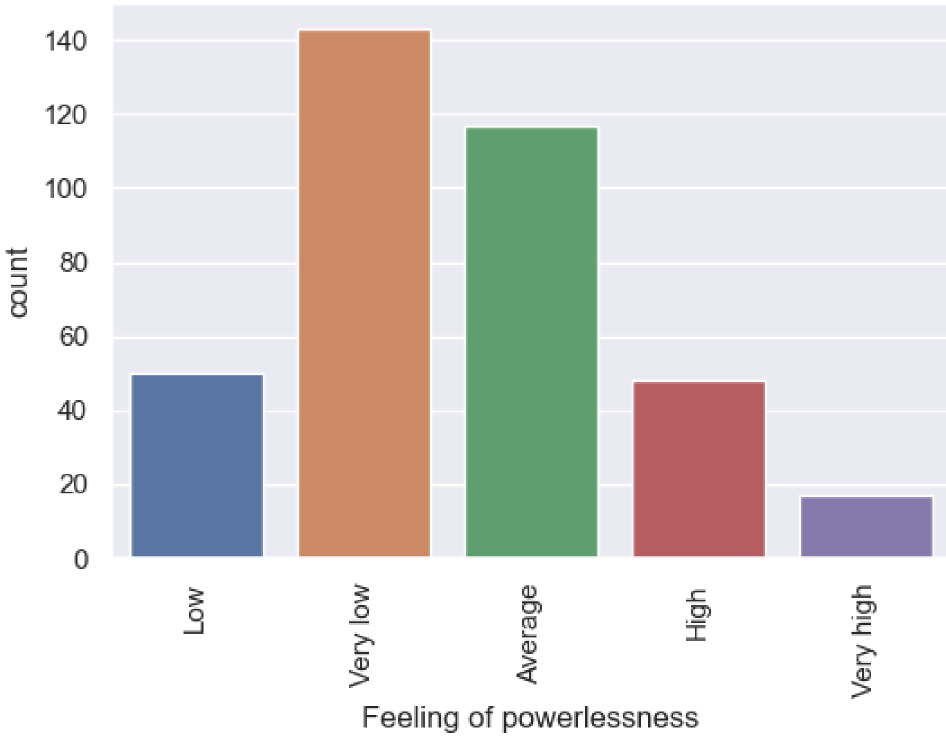


count
120
100
80
60
40
20
0
Average
High
Very low
Low
Very high
Monotonus
work

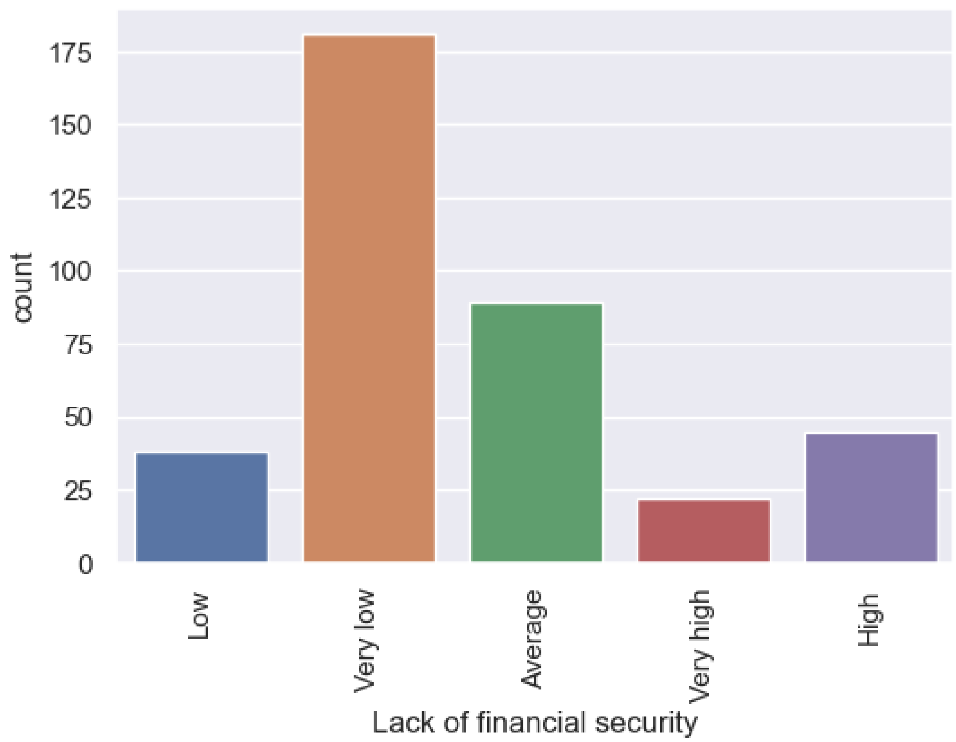


count
0
20
40
60
80
100
120
Low
Average
Very low
High
Very high
Poor and unplanned work

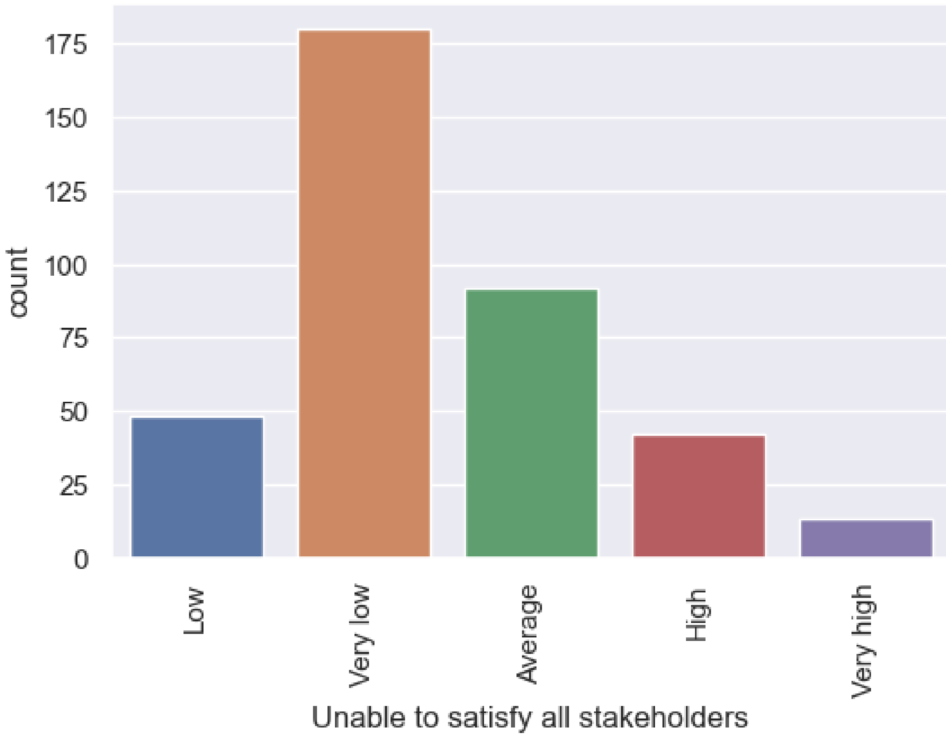

count
0
20
40
60
80
100
120
140
Low
Average
Very low
Very high
High
Lack of career development



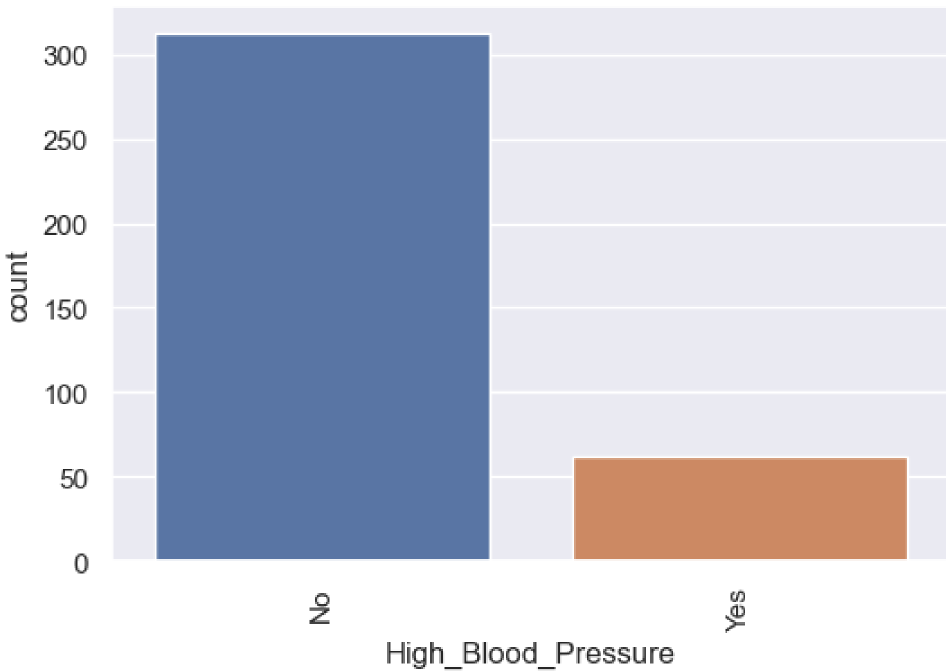
count
140
120
100
80
60
40
20
0
Low
Very low
Average
High
Very high
Feeling of powerlessness



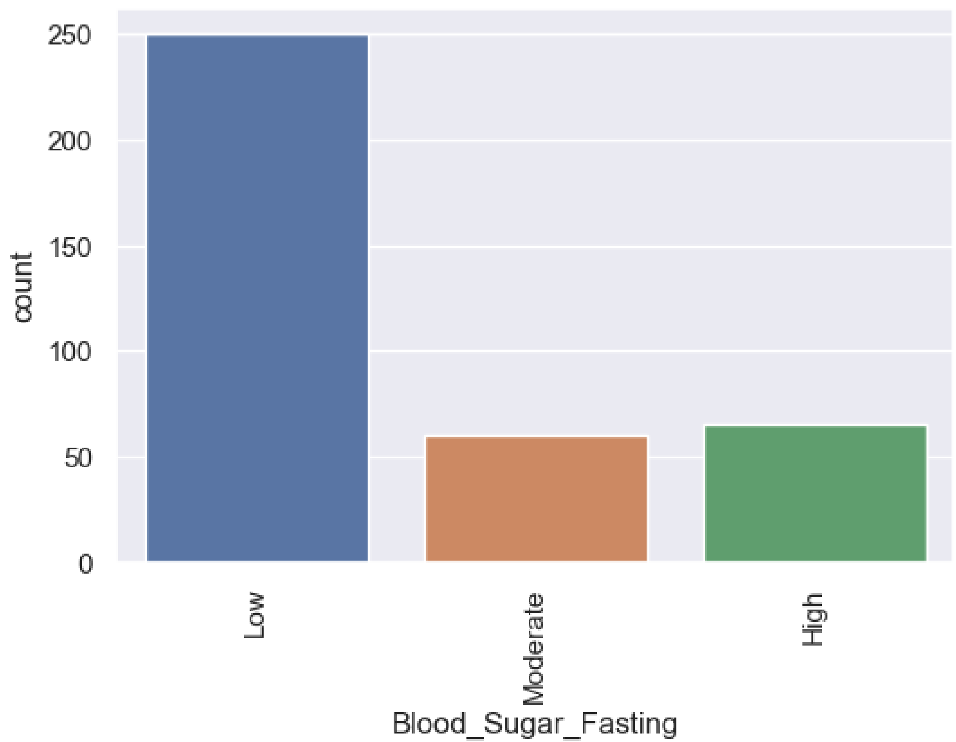
count
175
150
125
100
75
50
25
0
Low
Very low
Average
Very high
High
Lack of financial security

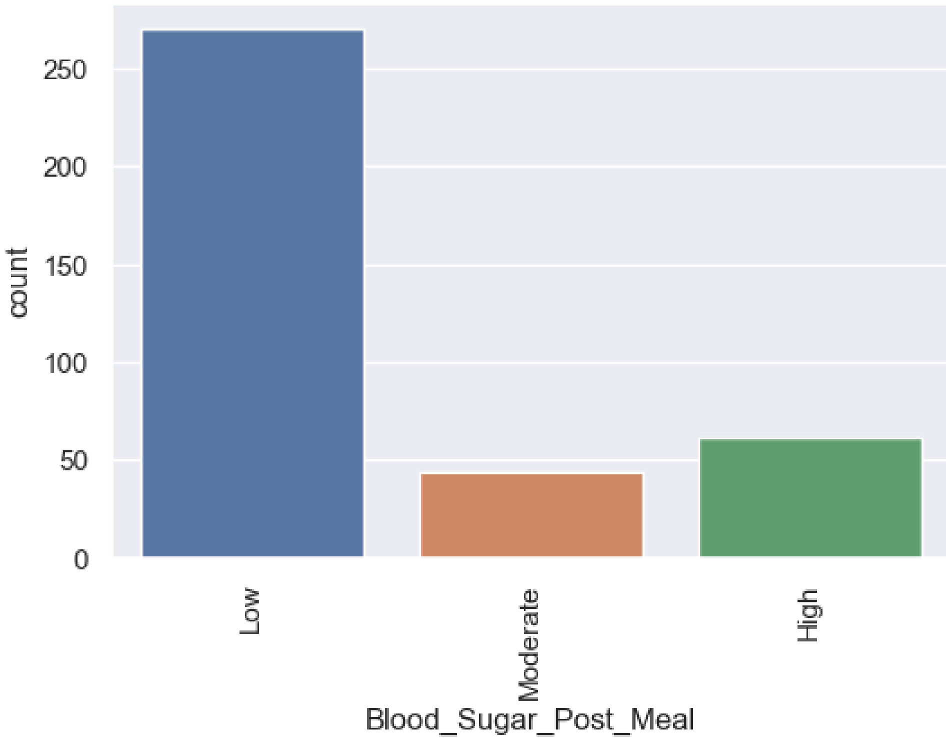


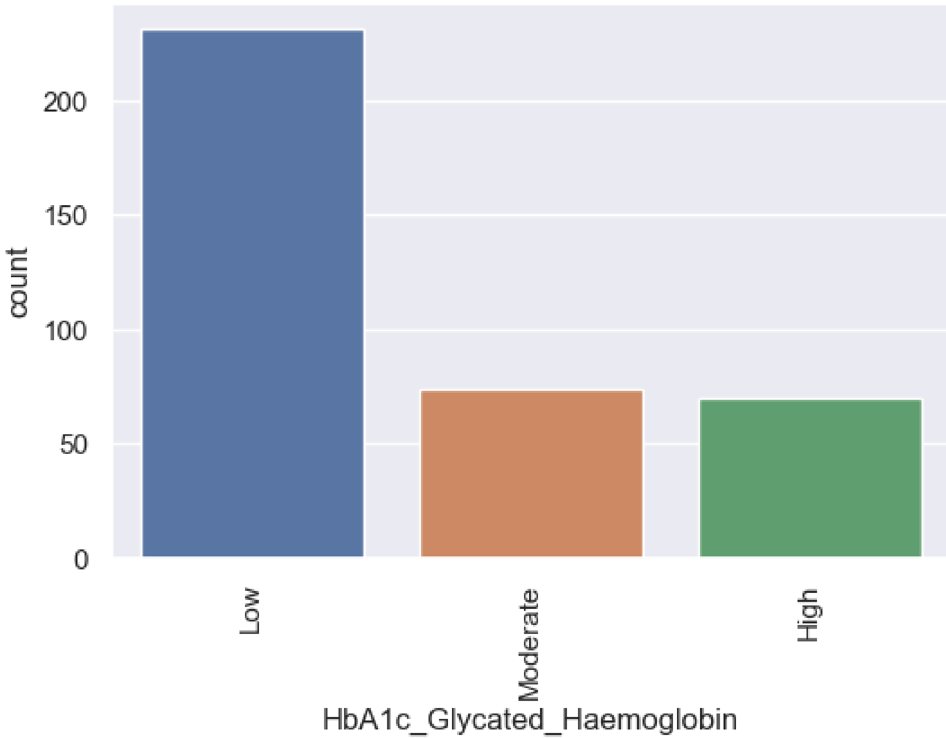
count
175
150
125
100
75
50
25
0
Low
Very low
Average
High
Very high
Unable to satisfy all stakeholders



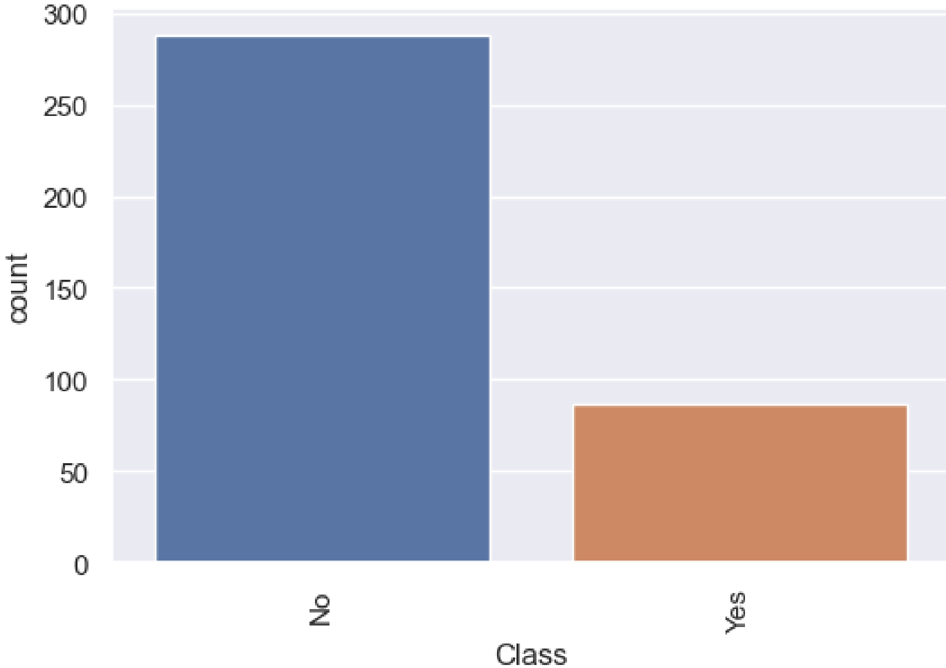
count
300
250
200
150
100
50
0
No
Yes
High_Blood_Pressure


250
200
150
100
50
0
count
Low
Moderate
High
Blood_Sugar_Fasting


count
250
200
150
100
50
0
Low
Moderate
High
Blood_Sugar_Post_Meal



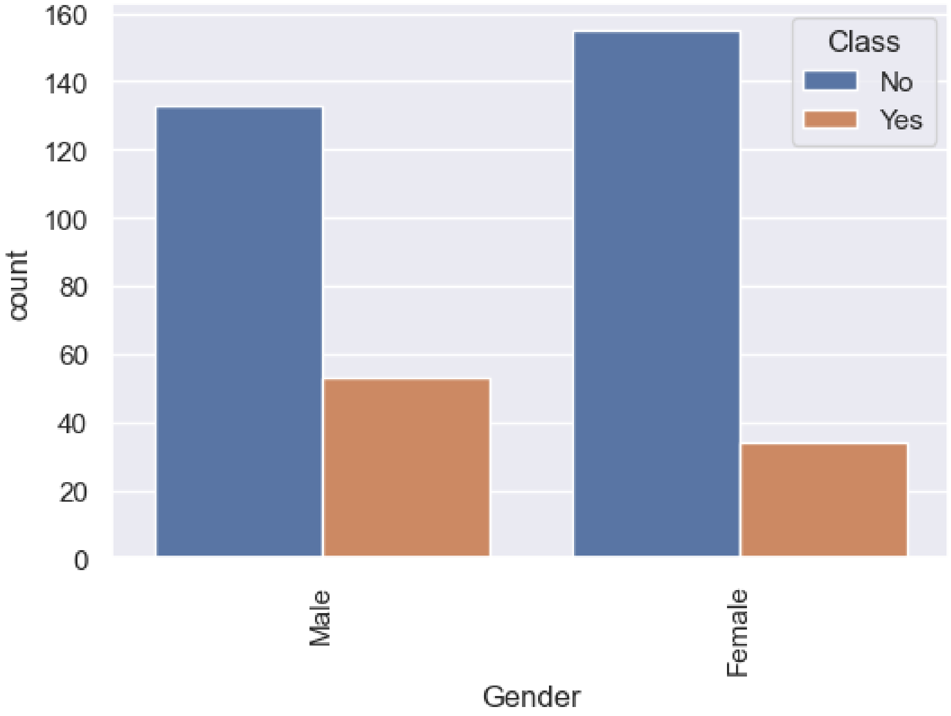
count
HbA1c_Glycated_Haemoglobin
Low
Moderate
High
0
50
100
150
200

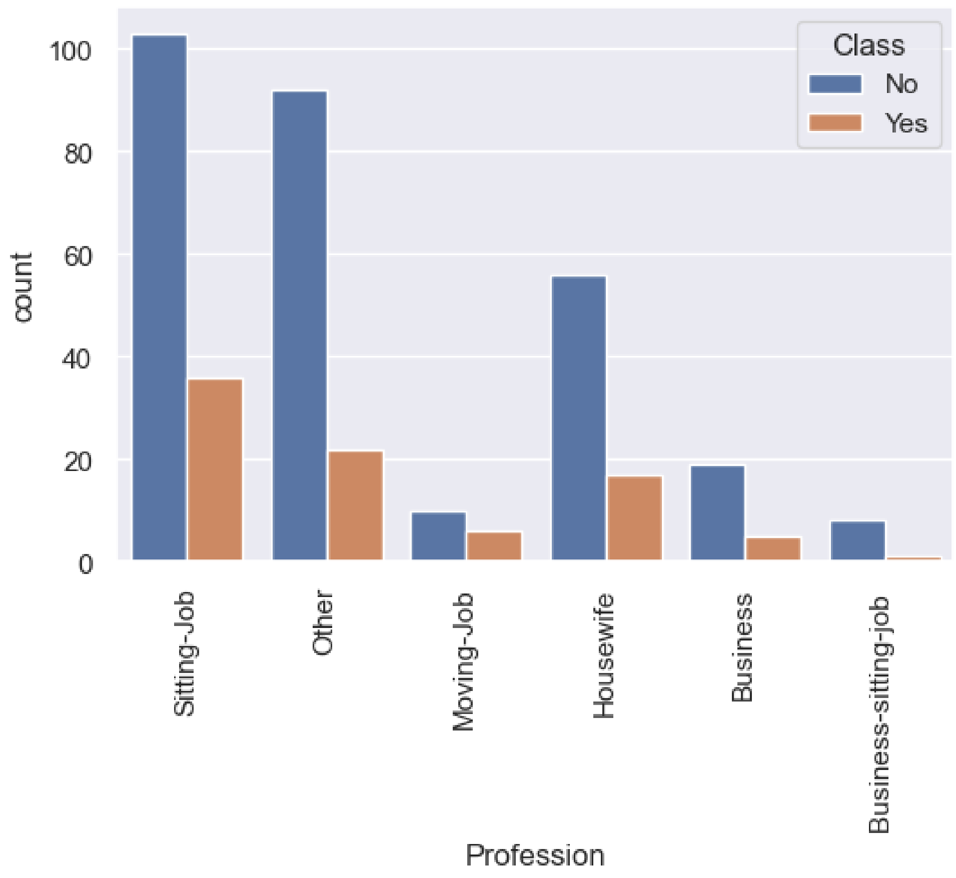


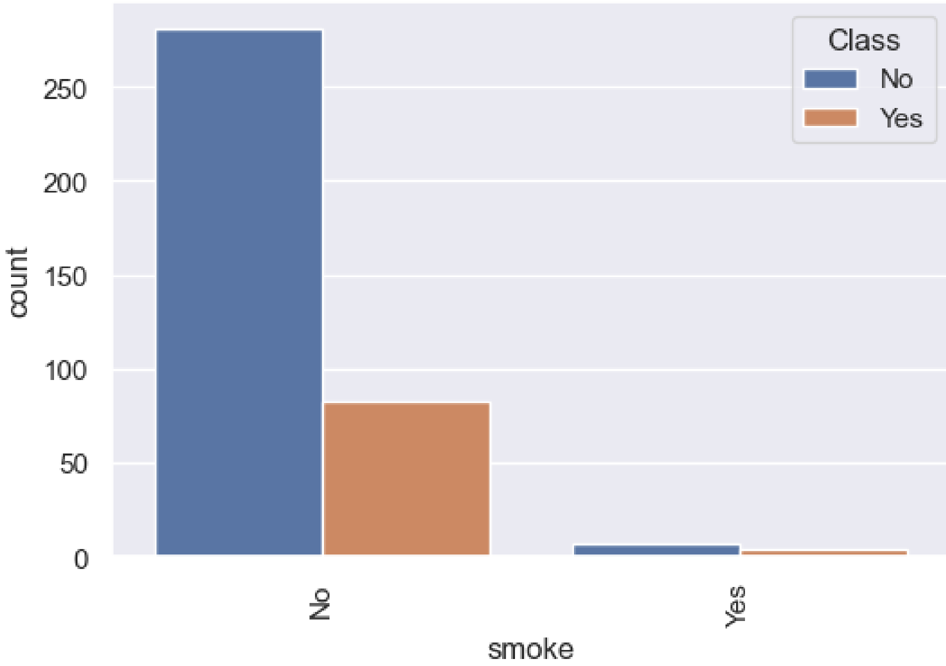
count
Class
No
Yes
0
50
100
150
200
250
300

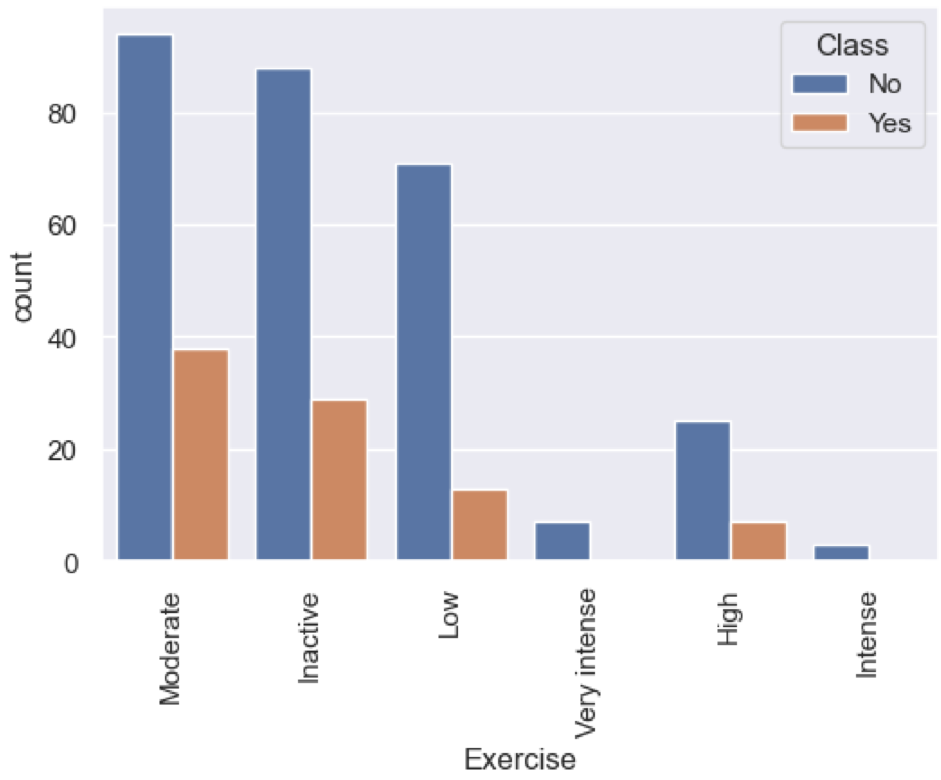
In [17]:

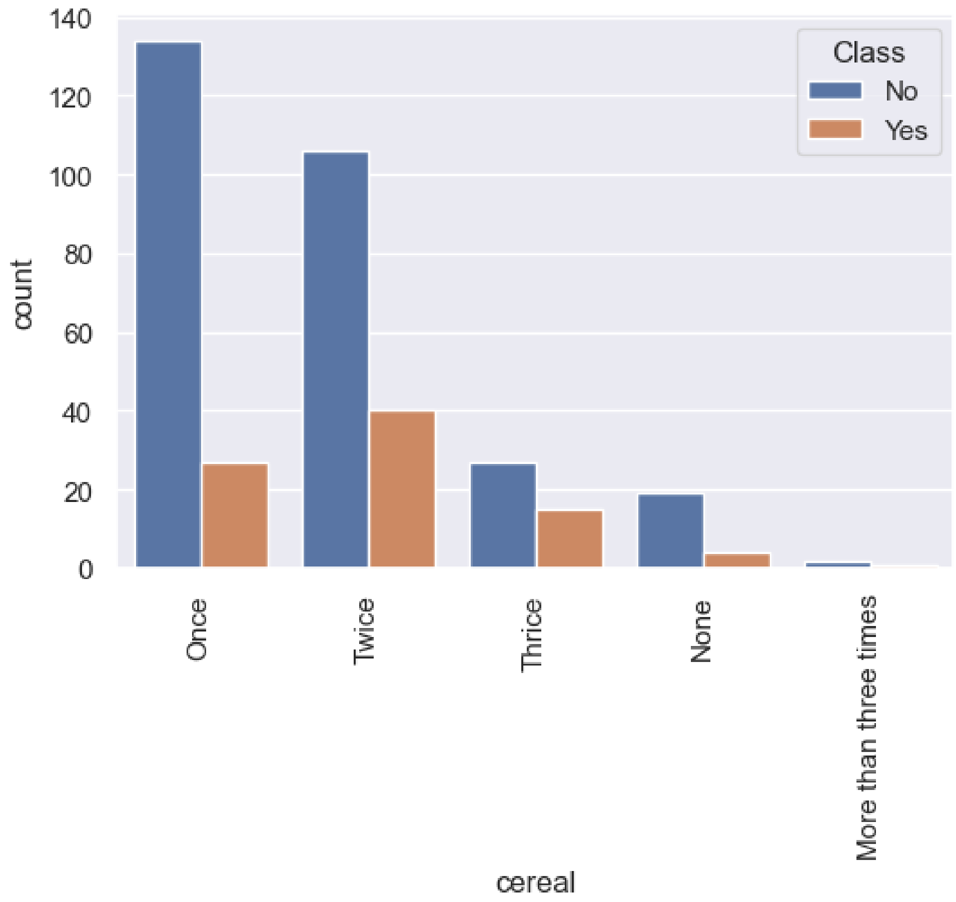
```
# Count Plot: Displays the count of occurrences for each category as bars.
for col in categorical_variables_df.columns[:-1]:
    plt.xticks(rotation=90)
    sns.countplot(data=categorical_variables_df, x=col, hue="Class")
    plt.show()
```

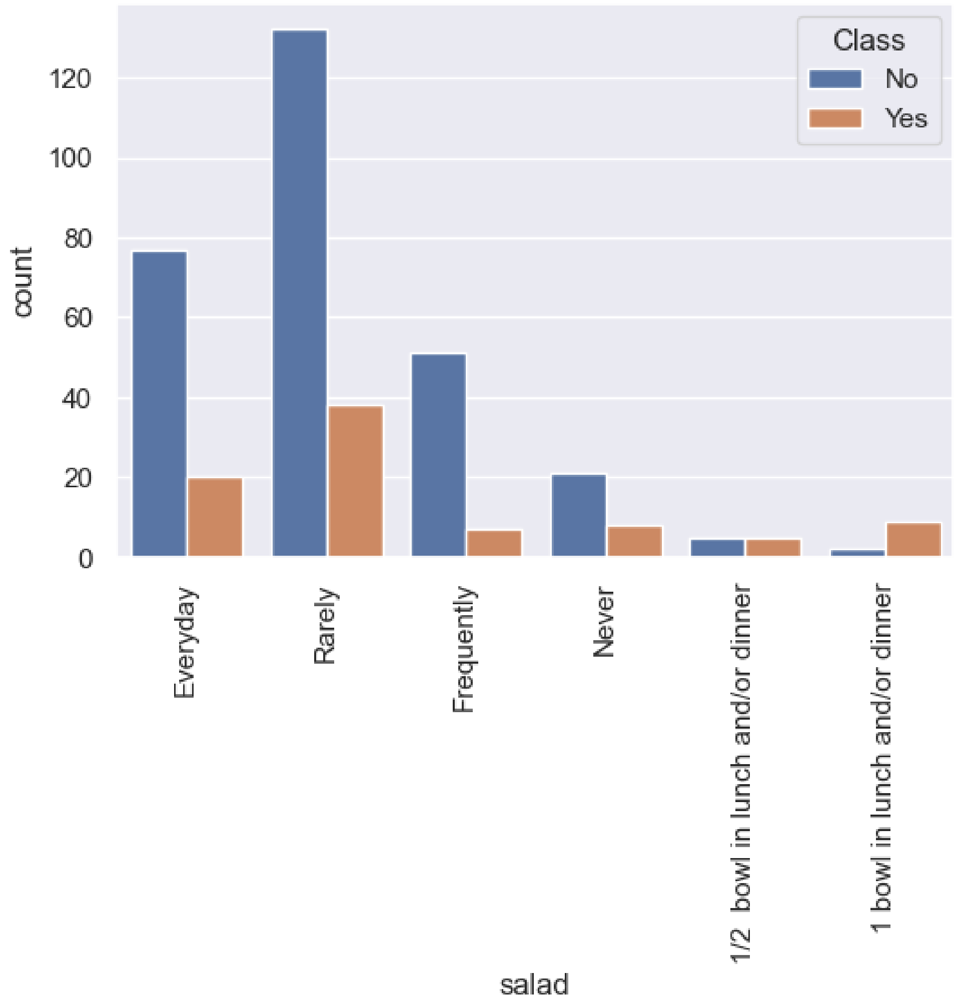



Class
No
Yes
count
100
80
60
40
20
0
Sitting-Job
Other
Moving-Job
Housewife
Business
Business-sitting-job
Profession

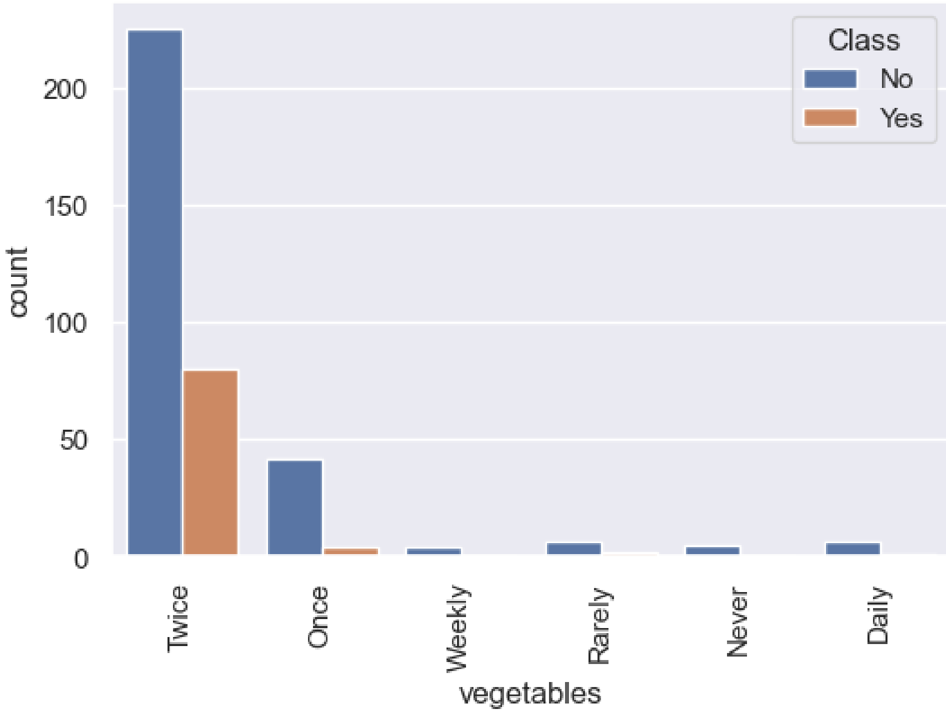

Class
No
Yes
count
250
200
150
100
50
0
No
Yes
smoke

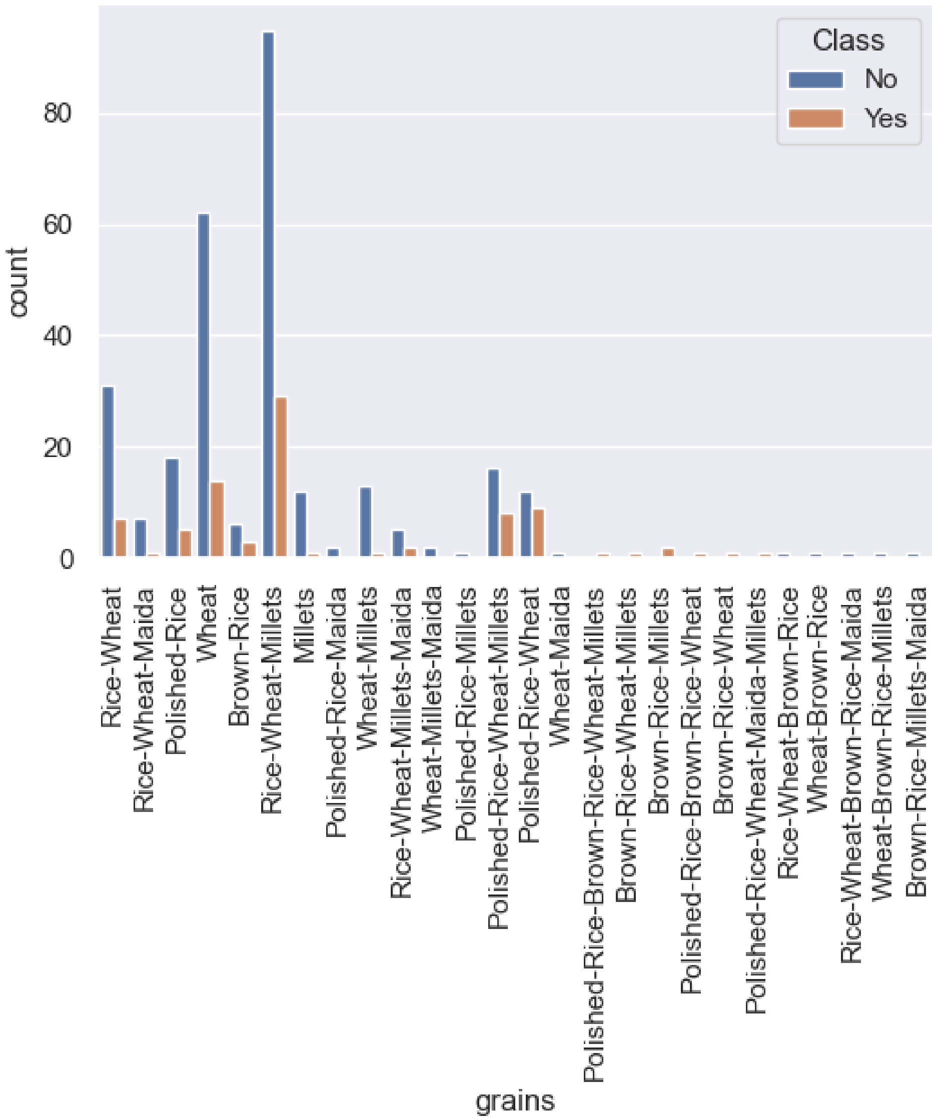

Class
No
Yes
count
0
20
40
60
80
Moderate
Inactive
Low
Very intense
High
Intense
Exercise


count
140
120
100
80
60
40
20
0
Class
No
Yes
Once
Twice
Thrice
None
More than three times
cereal

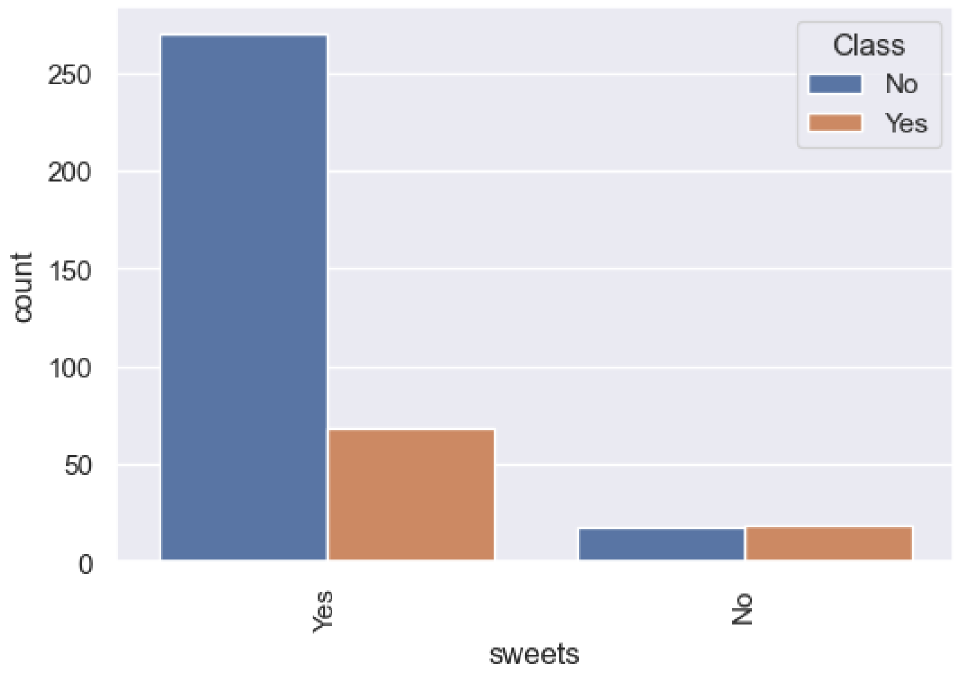


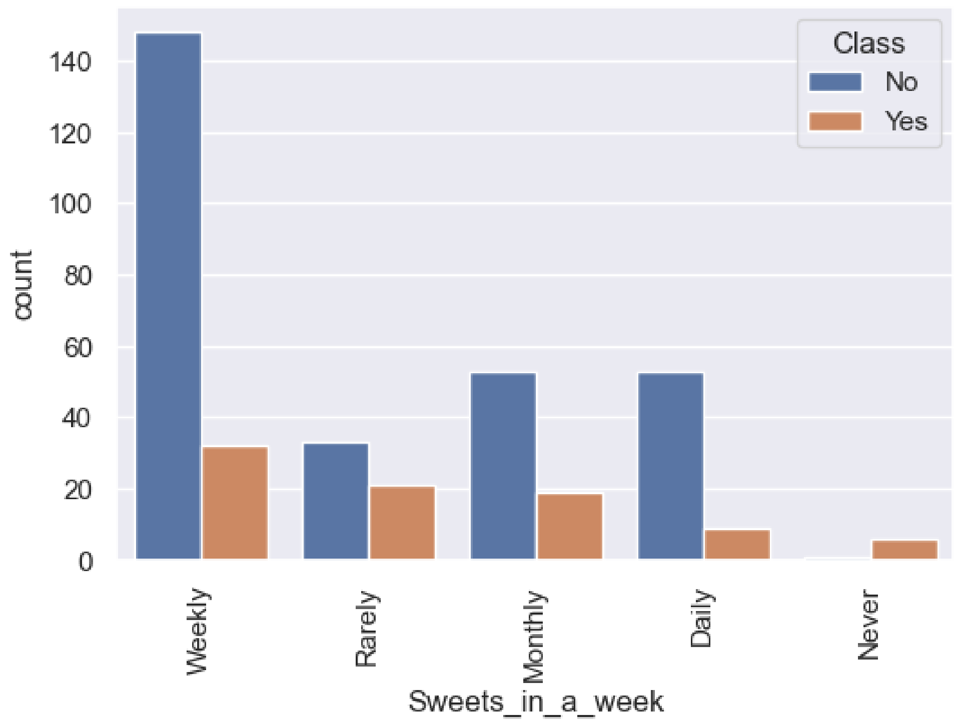
Class
No
Yes
count
120
100
80
60
40
20
0
Everyday
Rarely
Frequently
Never
1/2 bowl in lunch and/or dinner
1 bowl in lunch and/or dinner
salad

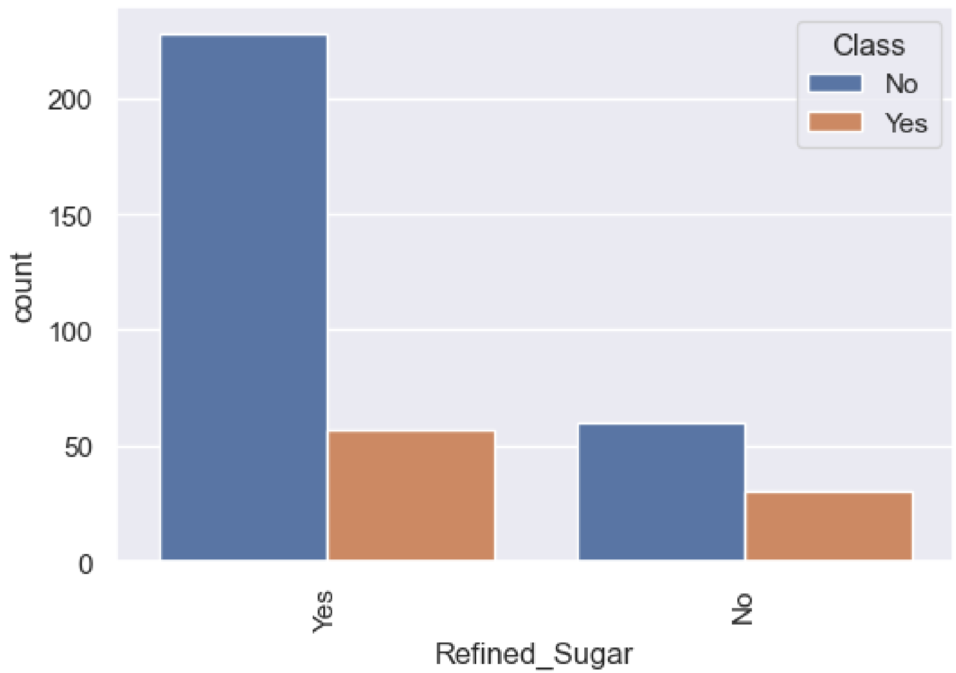

Class
No
Yes
count
200
150
100
50
0
Twice
Once
Weekly
Rarely
Never
Daily
vegetables

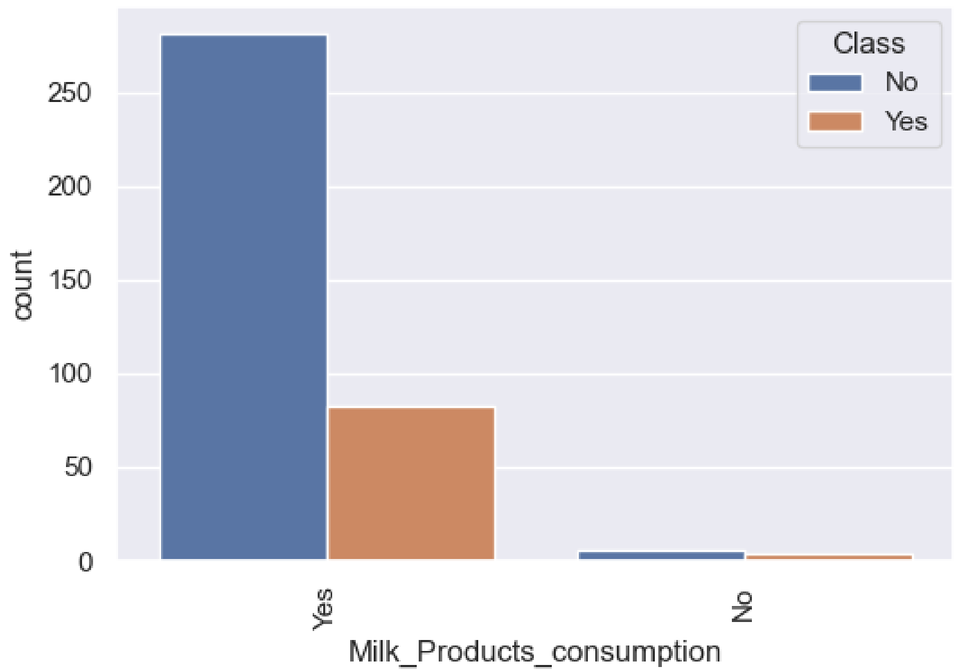


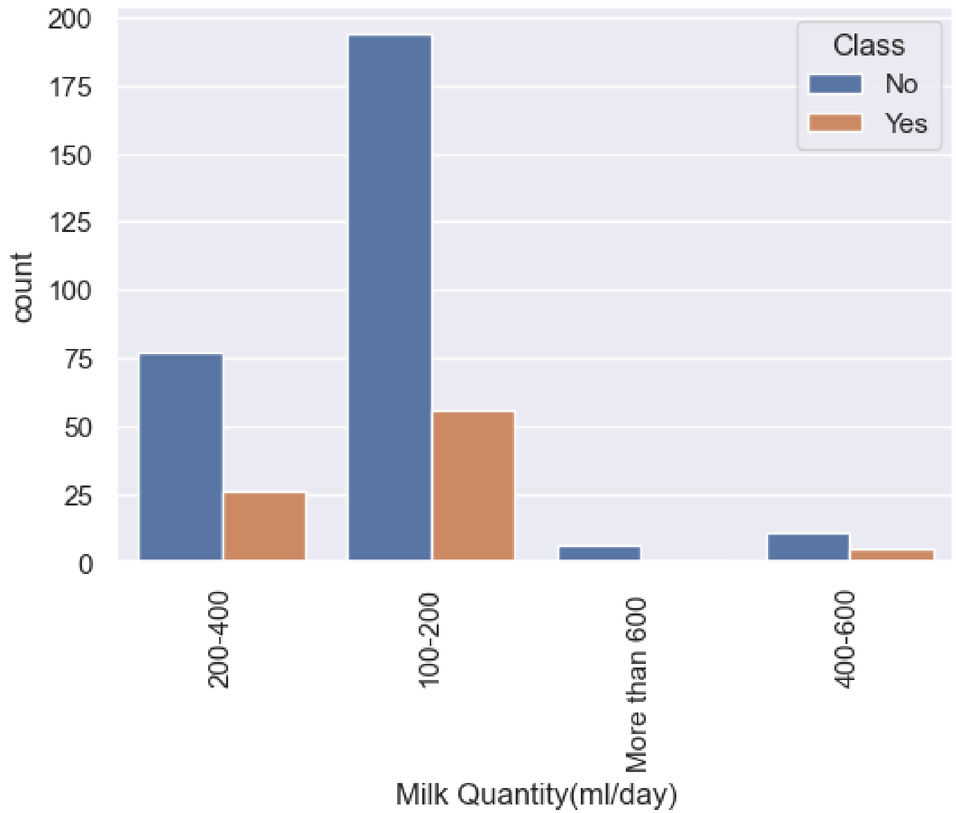
Class
No
Yes
count
0
20
40
60
80
grains
Rice-Wheat
Rice-Wheat-Maida
Polished-Rice
Wheat
Brown-Rice
Rice-Wheat-Millets
Millets
Polished-Rice-Maida
Wheat-Millets
Rice-Wheat-Millets-Maida
Wheat-Millets-Maida
Polished-Rice-Millets
Polished-Rice-Wheat-Millets
Polished-Rice-Wheat
Wheat-Maida
Polished-Rice-Brown-Rice-Wheat-Millets
Brown-Rice-Wheat-Millets
Brown-Rice-Millets
Polished-Rice-Brown-Rice-Wheat
Brown-Rice-Wheat
Polished-Rice-Wheat-Maida-Millets
Rice-Wheat-Brown-Rice
Wheat-Brown-Rice
Rice-Wheat-Brown-Rice-Maida
Wheat-Brown-Rice-Millets
Brown-Rice-Millets-Maida


Class
No
Yes
250
200
150
100
50
0
count
Yes
No
sweets

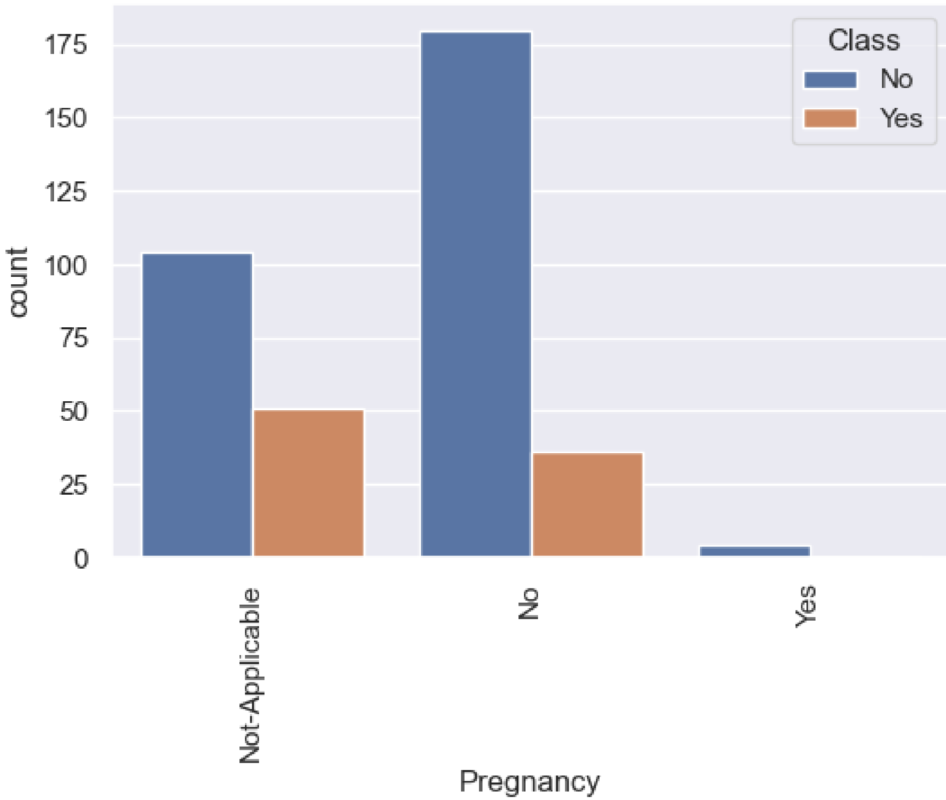

Class
No
Yes
140
120
100
80
60
40
20
0
count
Weekly
Rarely
Monthly
Daily
Never
Sweets_in_a_week

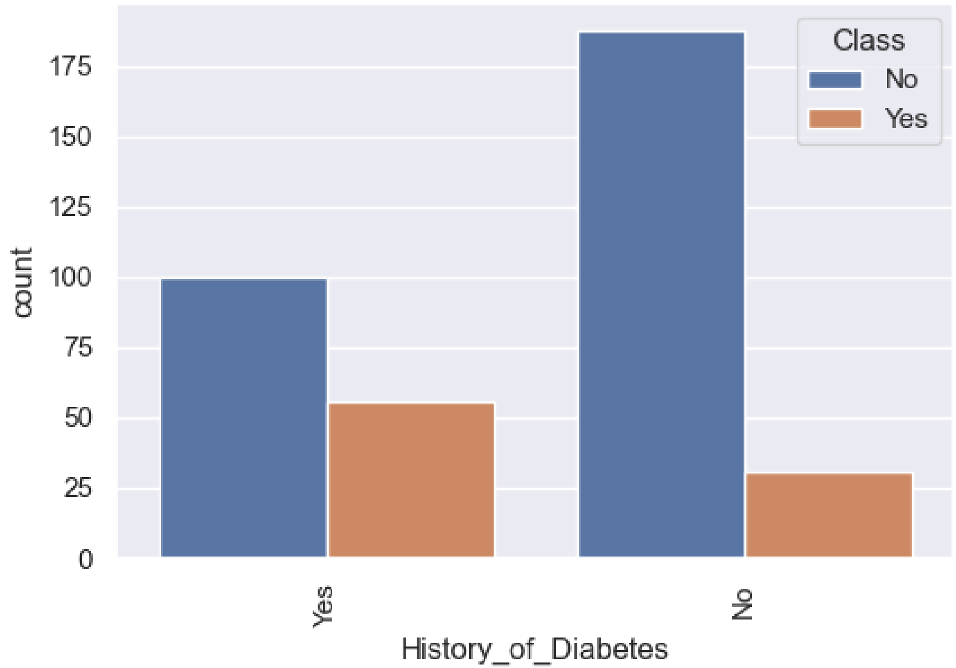

Class
No
Yes
count
200
150
100
50
0
Yes
No
Refined_Sugar

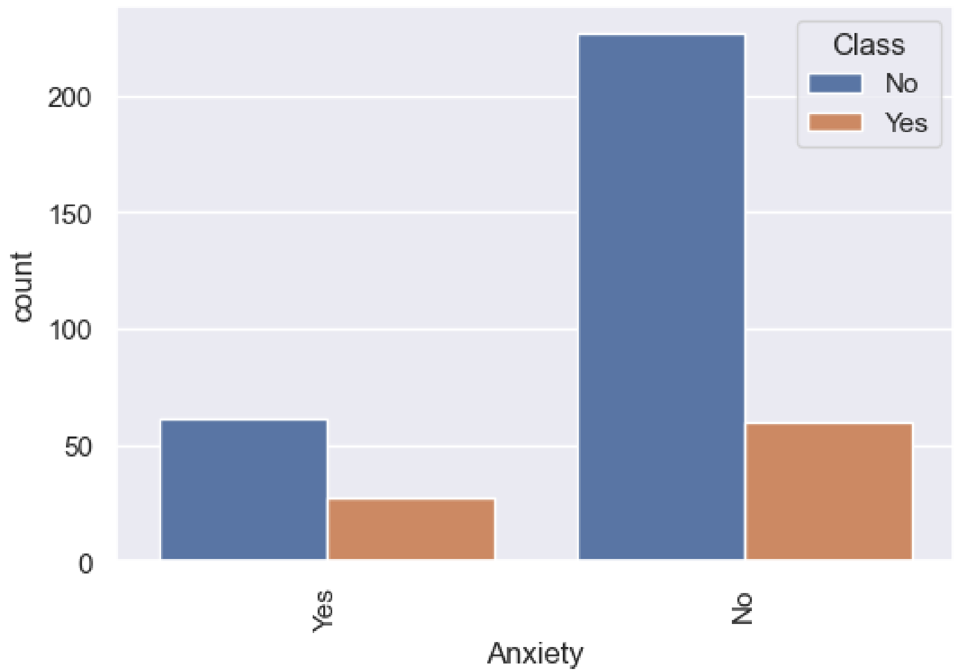

Class
No
Yes
count
250
200
150
100
50
0
Yes
No
Milk_Products_consumption

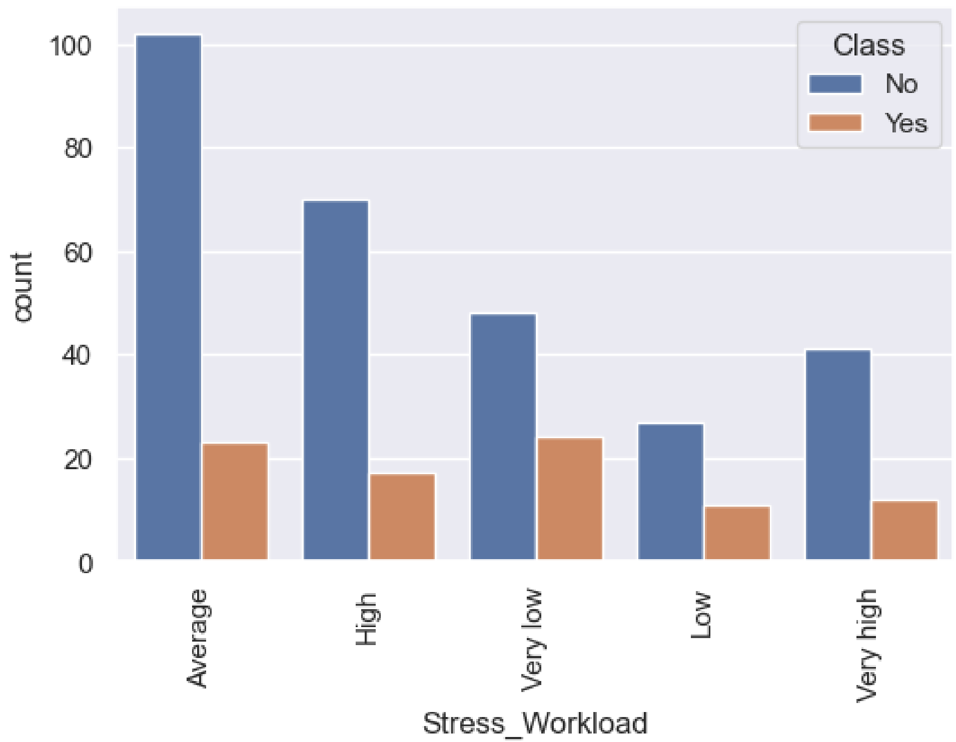


Class
No
Yes
count
200
175
150
125
100
75
50
25
0
200-400
100-200
More than 600
400-600
Milk Quantity(ml/day)

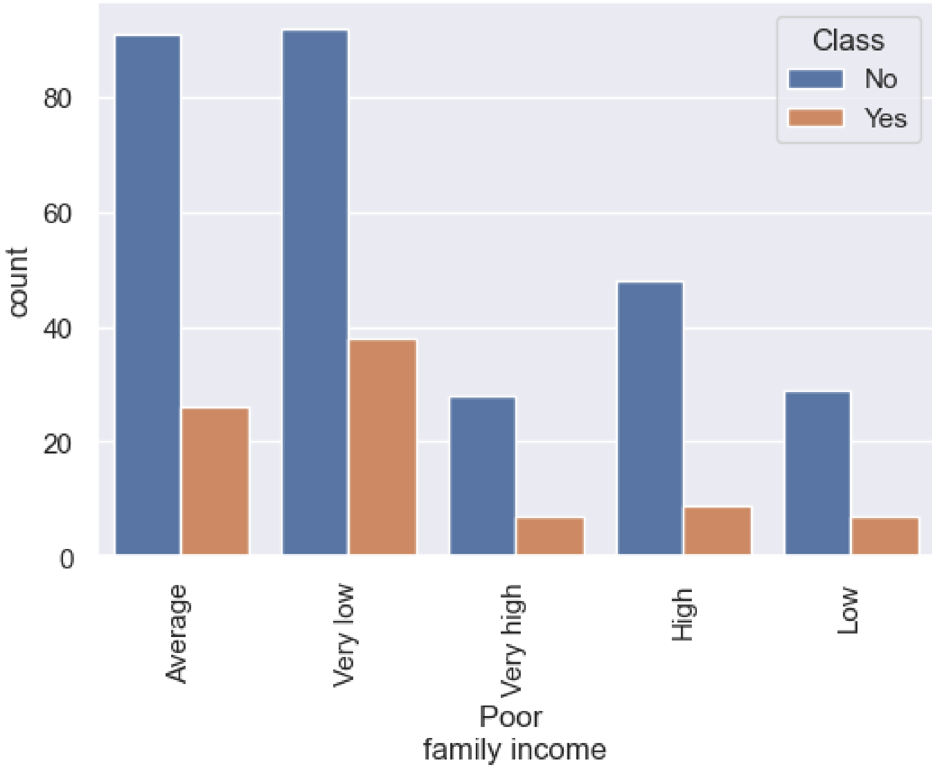

Class
No
Yes
count
175
150
125
100
75
50
25
0
Not-Applicable
No
Yes
Pregnancy


Class
No
Yes
count
0
25
50
75
100
125
150
175
Yes
No
History_of_Diabetes

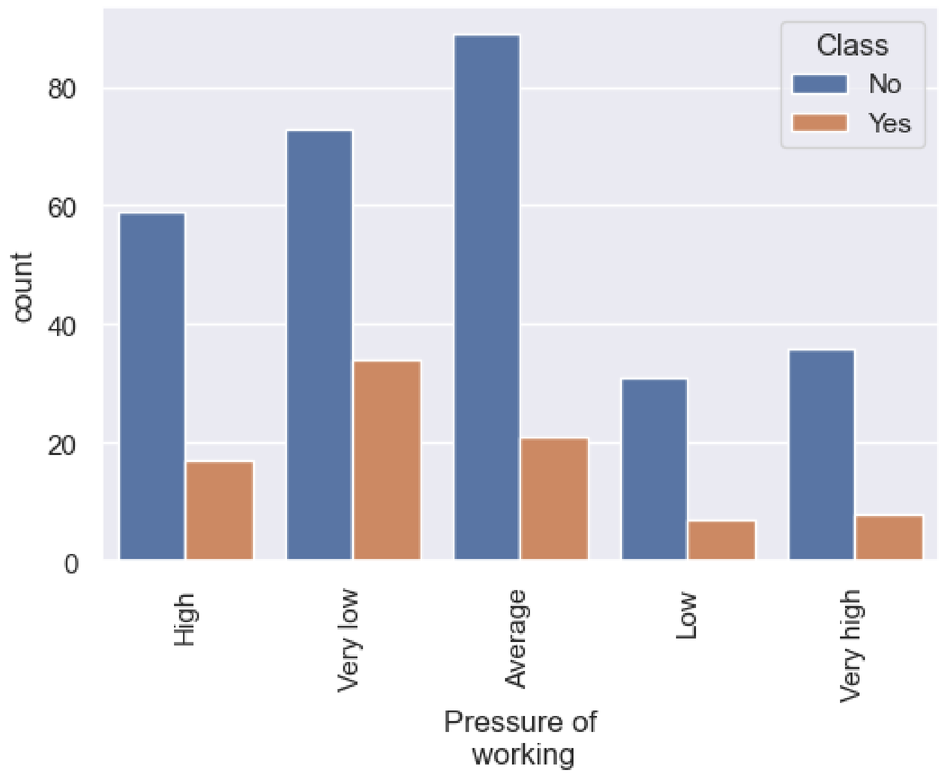

Class
No
Yes
count
0
50
100
150
200
Yes
No
Anxiety



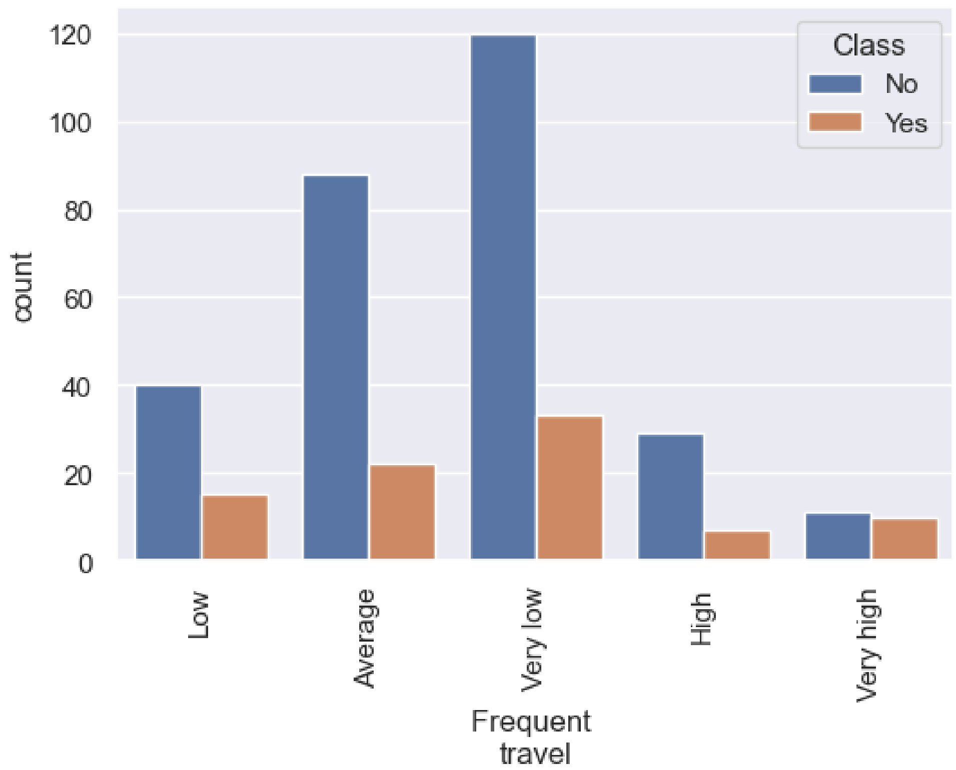
Class
No
Yes
100
80
60
40
20
0
count
Average
High
Very low
Low
Very high
Stress_Workload



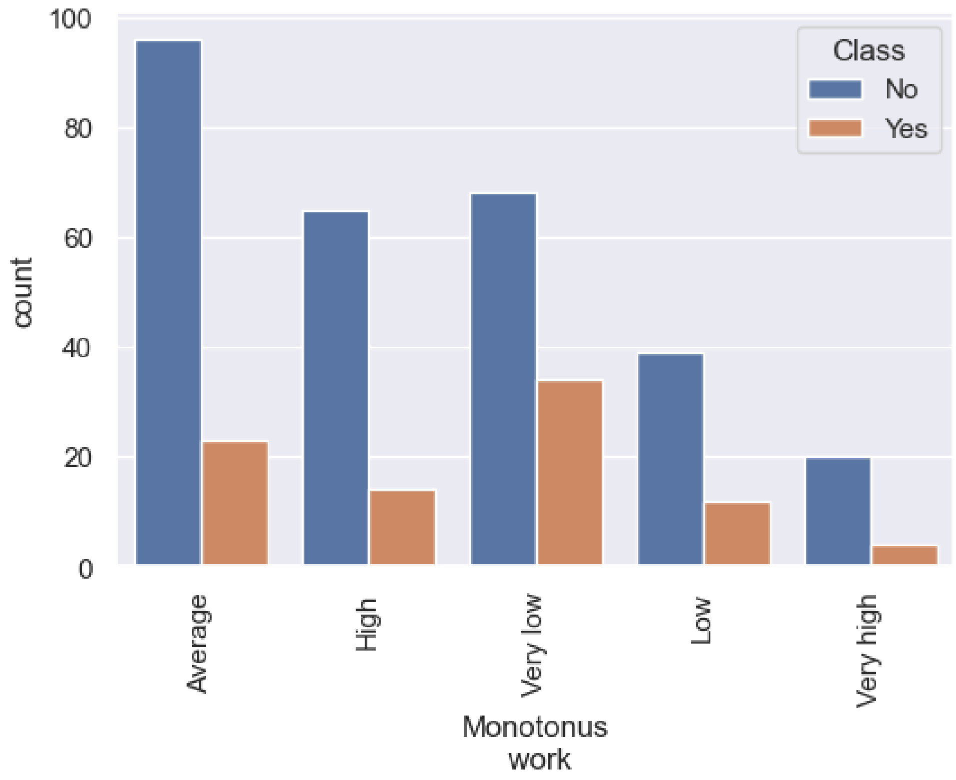
Class
No
Yes
count
0
20
40
60
80
Average
Very low
Very high
High
Low
Poor
family income

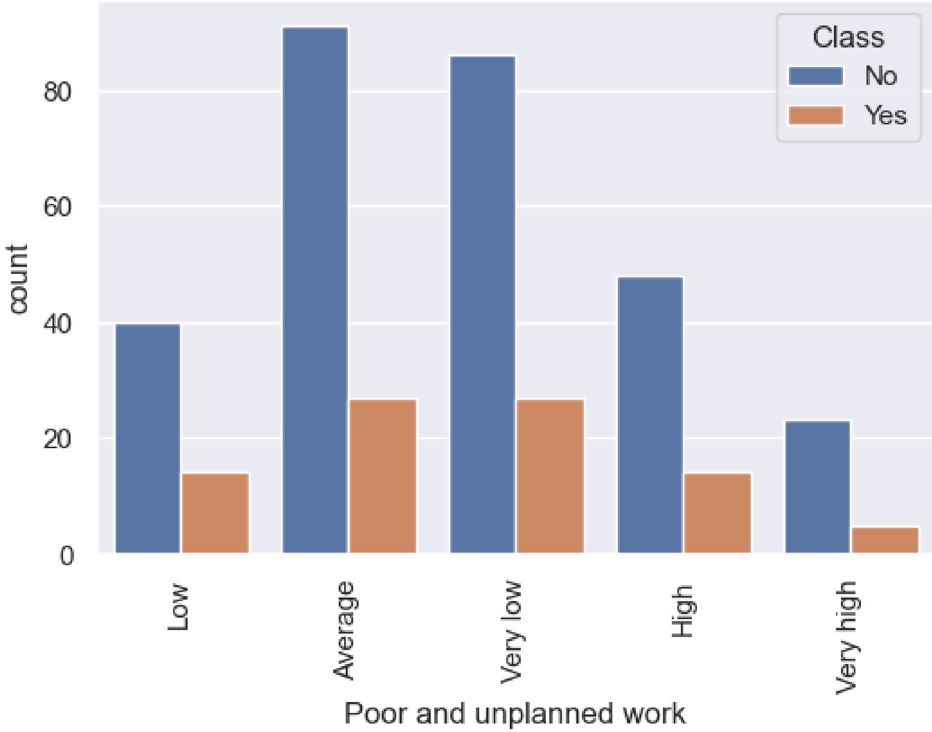


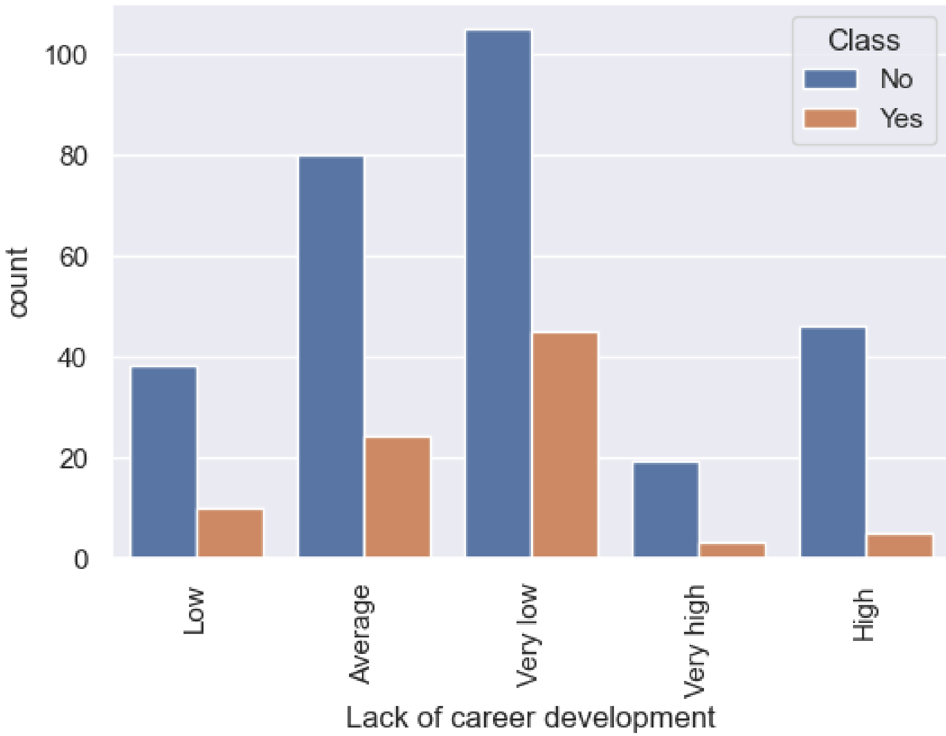
Class
No
Yes
count
0
20
40
60
80
High
Very low
Average
Low
Very high
Pressure of
working



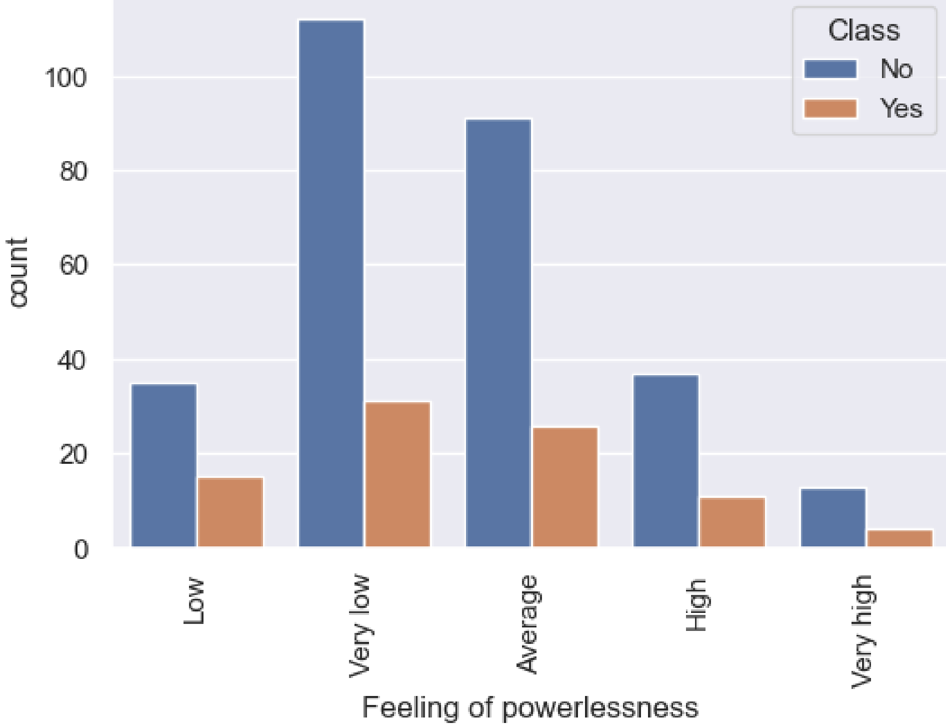
Class
No
Yes
count
120
100
80
60
40
20
0
Low
Average
Very low
High
Very high
Frequent travel

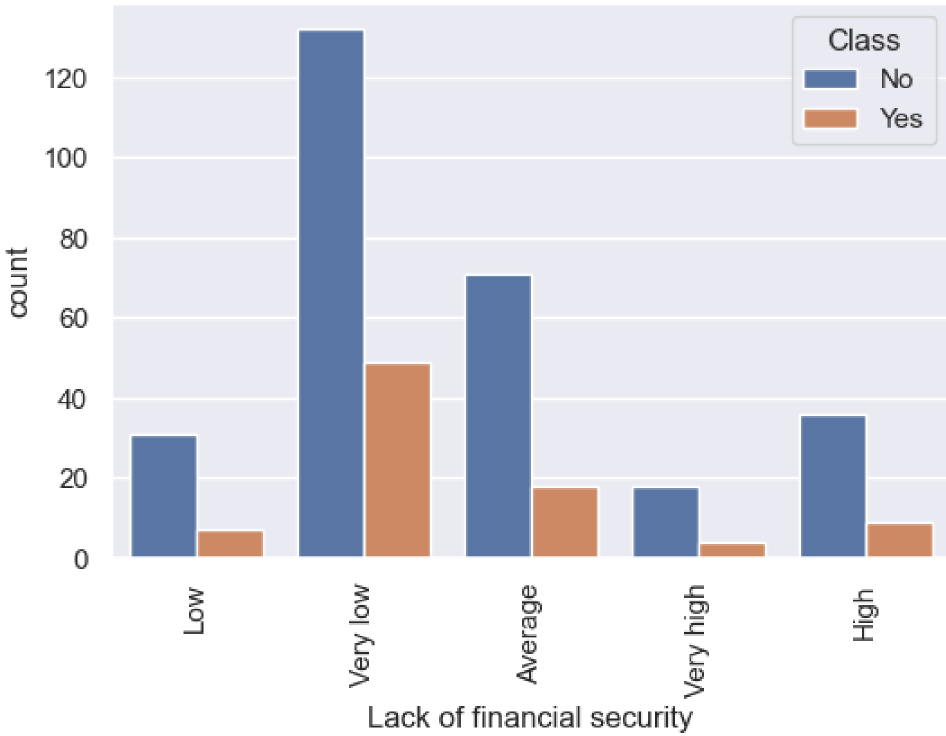

Class
No
Yes
count
100
80
60
40
20
0
Average
High
Very low
Low
Very high
Monotonus
work


Class
No
Yes
count
0
20
40
60
80
Low
Average
Very low
High
Very high
Poor and unplanned work

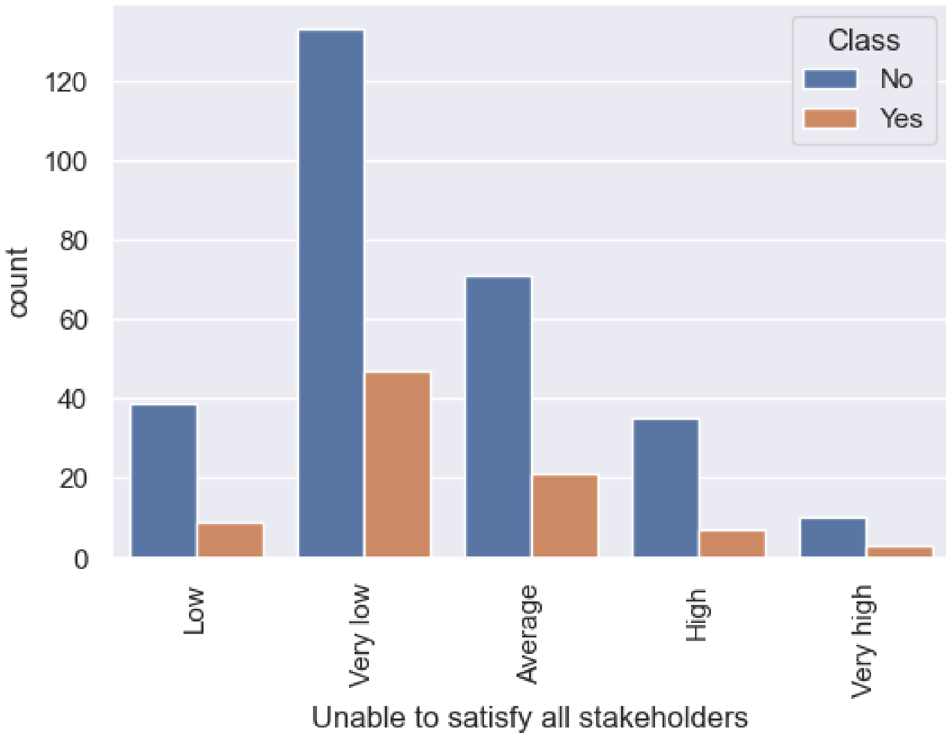


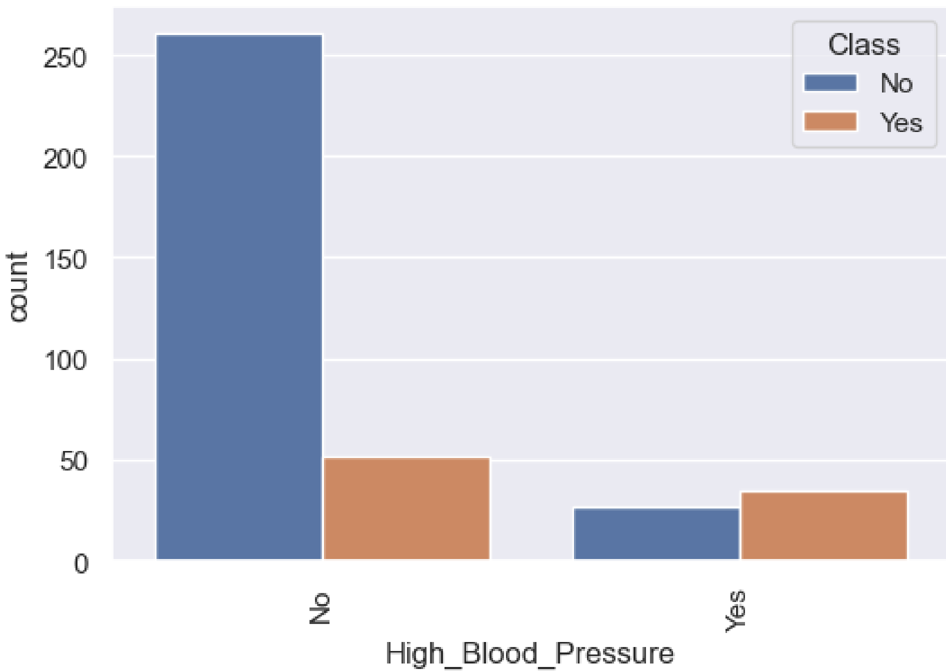
Class
No
Yes
count
100
80
60
40
20
0
Low
Average
Very low
Very high
High
Lack of career development

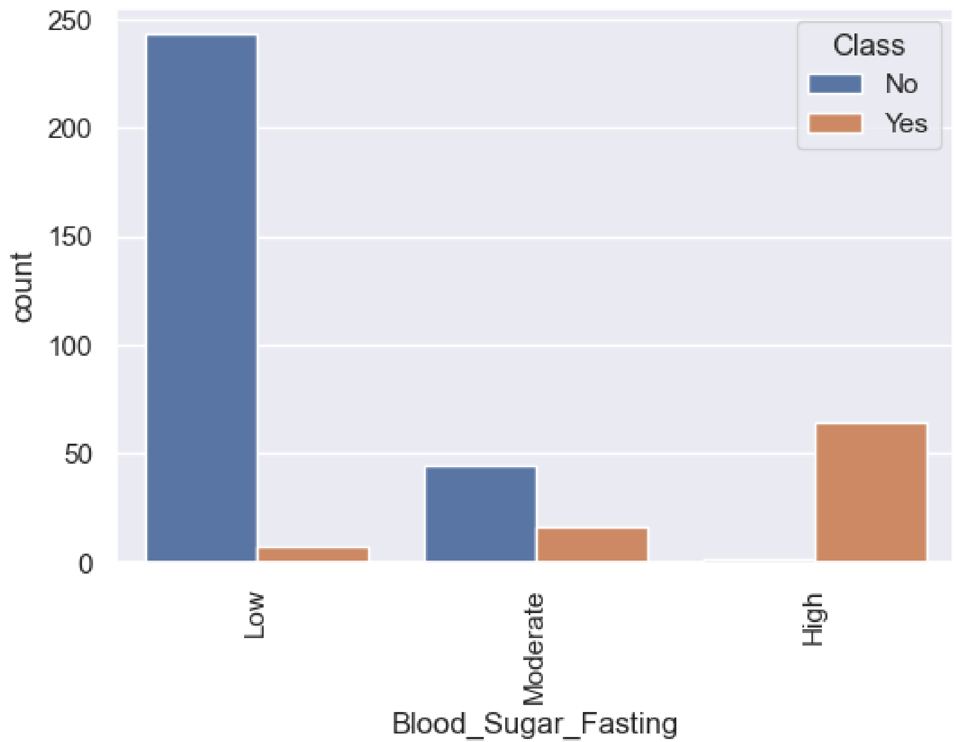

Class
No
Yes
count
100
80
60
40
20
0
Low
Very low
Average
High
Very high
Feeling of powerlessness



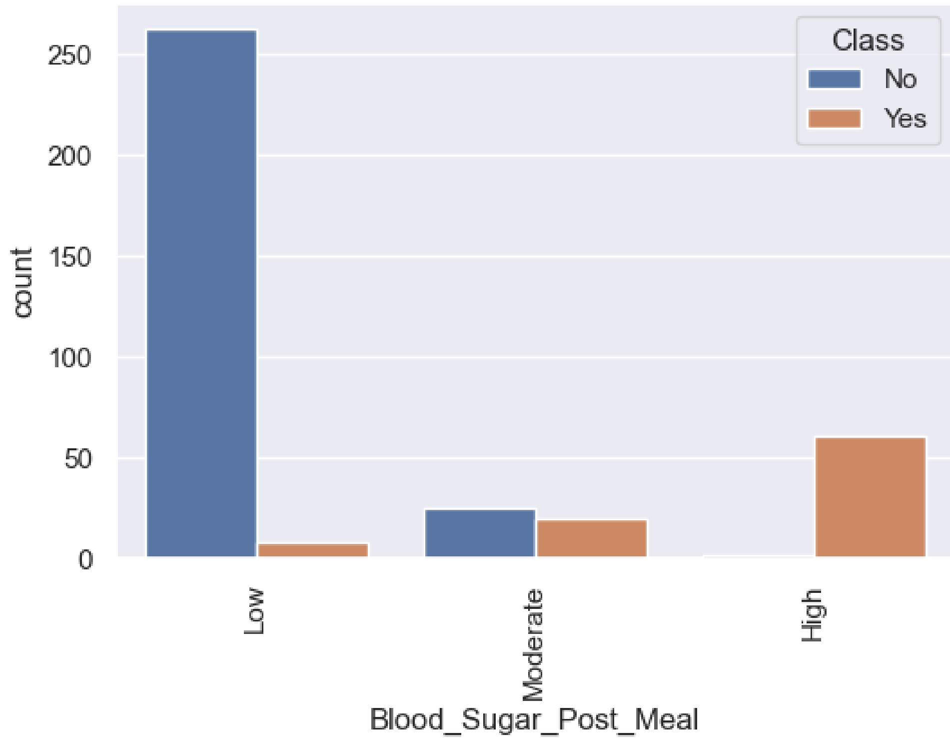
Class
No
Yes
count
120
100
80
60
40
20
0
Low
Very low
Average
Very high
High
Lack of financial security

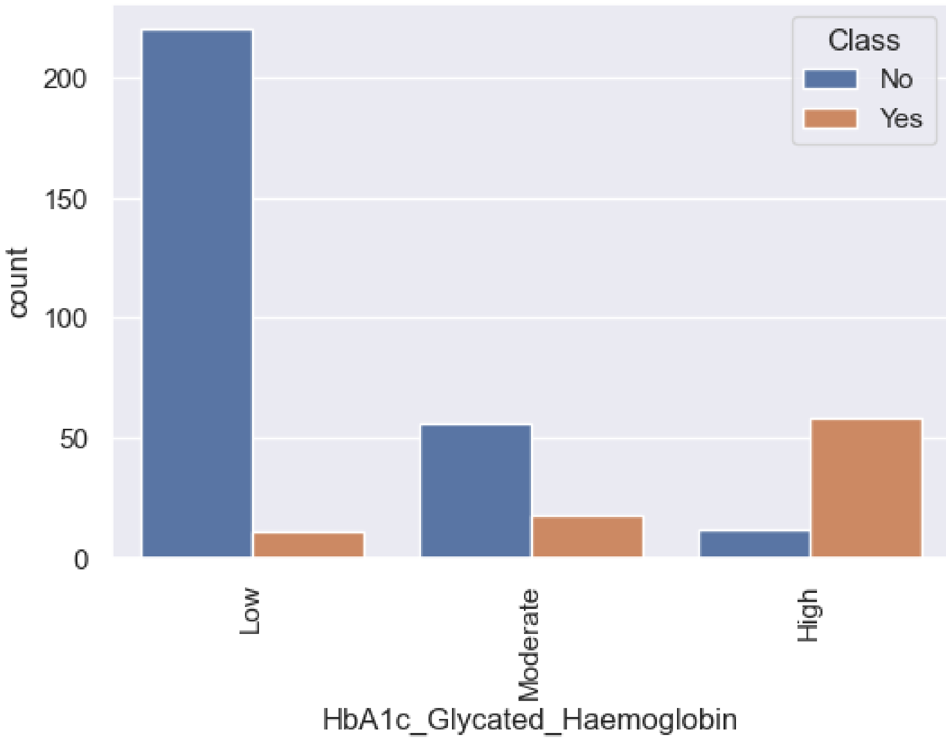

Class
No
Yes
count
0
20
40
60
80
100
120
Low
Very low
Average
High
Very high
Unable to satisfy all stakeholders


Class
No
Yes
count
0
50
100
150
200
250
No
Yes
High_Blood_Pressure



Class
No
Yes
250
200
150
100
50
0
count
Low
Moderate
High
Blood_Sugar_Fasting


Class
No
Yes
250
200
150
100
50
0
count
Low
Moderate
High
Blood_Sugar_Post_Meal



In [18]:

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

In [19]:
```
categorical_variables_df.head()
```
Out[19]:

	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_Blo
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 [20]:
```
dataset = pd.concat([numerical_variables_df, categorical_variables_df], axis=1)
dataset.head()
```
Out[20]:

	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_P
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 [21]:
```
X, Y = dataset.drop(['Class'], axis = 1),  dataset['Class']
# 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 [22]:

```
# 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 [25]:

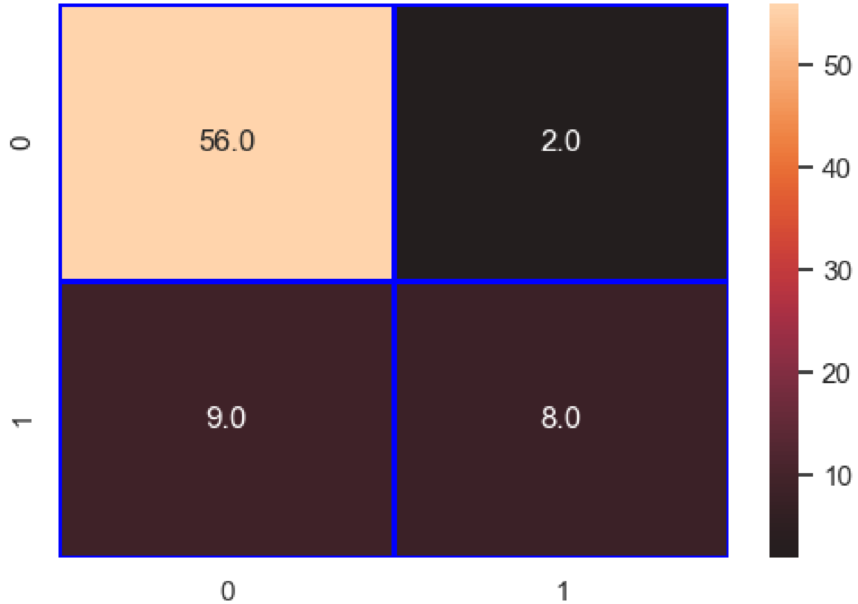
```
# def run_pipeline(X_train, X_test, y_train, y_test, scaler, classifier):
def run_pipeline(X_train, X_test, y_train, y_test, classifier):
    # Model Information
    print(f"Modele name : {type(classifier).__name__}")
    # print(f"Scaler name : {type(scaler).__name__}")

    # 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)
    classifier.fit(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)
    y_pred = classifier.predict(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')
    scores = cross_val_score(classifier, 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()
    print("Root Mean Square Error (RMSE):", round(sqrt(mean_squared_error(y_test, y_pred)), 4))

    # process 7 : save model in pkl file
    # filename = f'Moduls\\{str(type(classifier).__name__)}_{str(type(scaler).__name__)}_01_LS_Disease_Prediction.pkl'
    filename = f'Moduls\\{str(type(classifier).__name__)}_01_LS_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__),
                           round((accuracy_score(y_test, y_pred))*100,2)])

    # collect data for bar plot (RSME)
    global plot_RSME_list
    plot_RSME_list.append([str(type(classifier).__name__),
                           round(sqrt(mean_squared_error(y_test, y_pred)), 4)])

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

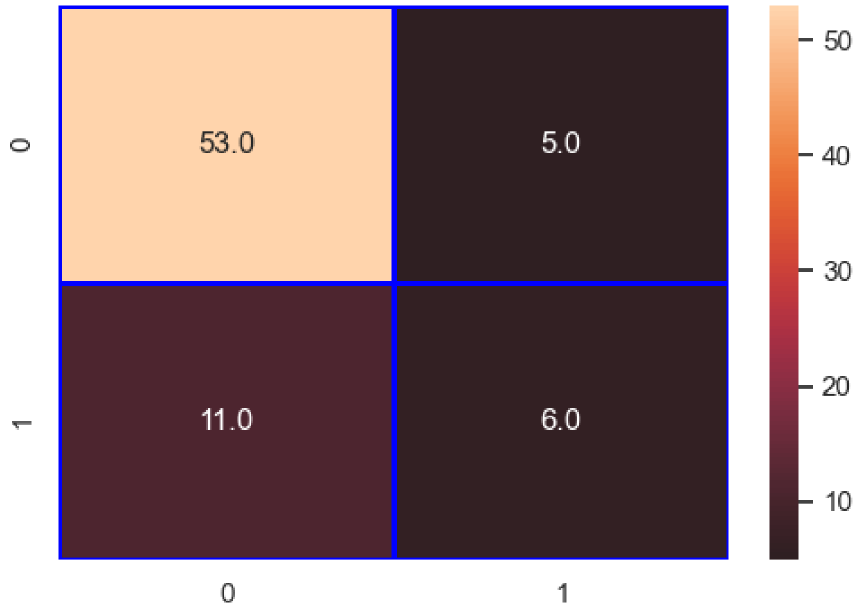
In [26]:

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

# plot data
# plot_df = pd.DataFrame(plot_data_list, columns=['classifier', 'scaler', 'accuracy_score'])
plot_df = pd.DataFrame(plot_data_list, columns=['classifier', 'accuracy_score'])
plot_df.to_csv(f"Dataset\\accuracy_score_plot_data_01_LS_Disease_Prediction.csv", index=False)

sns.set(rc={'figure.figsize':(10,6)})
# ax = sns.barplot(data=plot_df, x="classifier", y="accuracy_score", hue="scaler")
ax = sns.barplot(data=plot_df, x="classifier", y="accuracy_score") # , hue="classifier"
plt.title('Accuracy Score Plot')
plt.xlabel('Classifier')
plt.ylabel('Accuracy Score')
ax.tick_params(axis='x', rotation=5)
for i in ax.containers:
    ax.bar_label(i,)
plt.show()


plot_RSME = pd.DataFrame(plot_RSME_list, columns=['classifier', 'RSME_score'])
plot_RSME.to_csv(f"Dataset\\RSME_score_plot_data_01_LS_Disease_Prediction.csv", index=False)

sns.set(rc={'figure.figsize':(10,6)})
# ax = sns.barplot(data=plot_df, x="classifier", y="accuracy_score", hue="scaler")
ax = sns.barplot(data=plot_RSME, x="classifier", y="RSME_score") # , hue="classifier"
plt.title('RSME Score Plot')
plt.xlabel('Classifier')
plt.ylabel('RSME Score')
ax.tick_params(axis='x', rotation=5)
for i in ax.containers:
    ax.bar_label(i,)
plt.show()


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

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

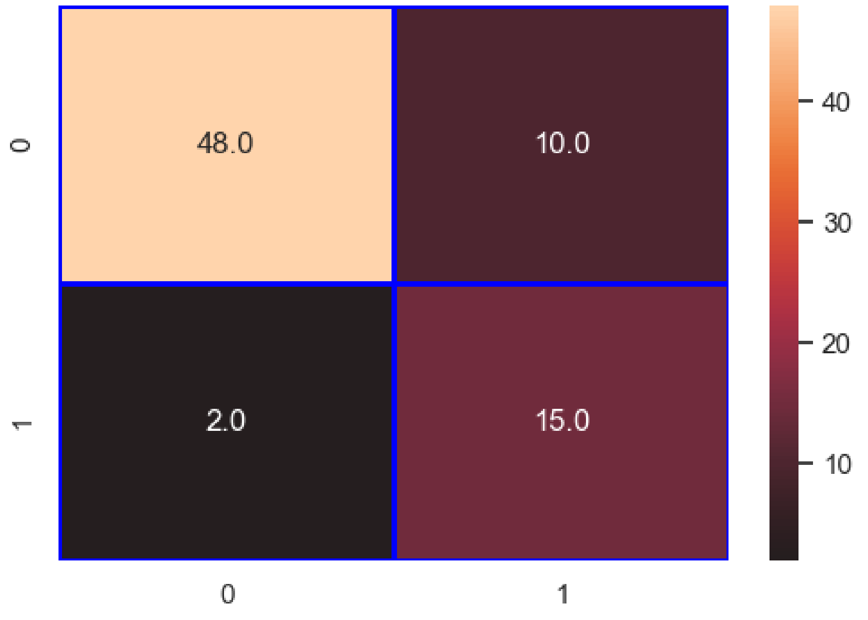
```
Modele name : LogisticRegression
10 K-Fold Accuracy_score : [0.9667 0.9    0.9    0.9333 0.9    0.9333 0.7667 0.9    0.7333 0.8667]
10 K-Fold Average Accuracy_score : 88.0 %
Accuracy_score: 85.33 %
Loss: 14.67 %
Cohen_kappa_score: 51.04 %
Classification_report:
               precision    recall  f1-score   support

           0       0.86      0.97      0.91        58
           1       0.80      0.47      0.59        17

    accuracy                           0.85        75
   macro avg       0.83      0.72      0.75        75
weighted avg       0.85      0.85      0.84        75

confusion_matrix:
 [[56  2]
 [ 9  8]]
```



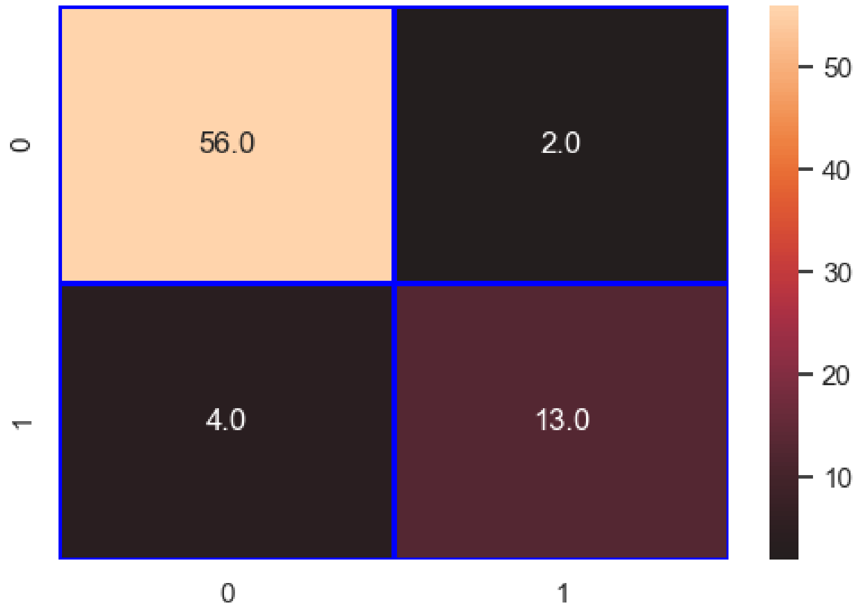
```
Root Mean Square Error (RMSE): 0.383
==========================================================================================



Modele name : KNeighborsClassifier
10 K-Fold Accuracy_score : [0.8333 0.7333 0.8667 0.8    0.7667 0.7333 0.8    0.8    0.7667 0.8    ]
10 K-Fold Average Accuracy_score : 79.0 %
Accuracy_score: 78.67 %
Loss: 21.33 %
Cohen_kappa_score: 30.48 %
Classification_report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87        58
           1       0.55      0.35      0.43        17

    accuracy                           0.79        75
   macro avg       0.69      0.63      0.65        75
weighted avg       0.76      0.79      0.77        75

confusion_matrix:
 [[53  5]
 [11  6]]
```



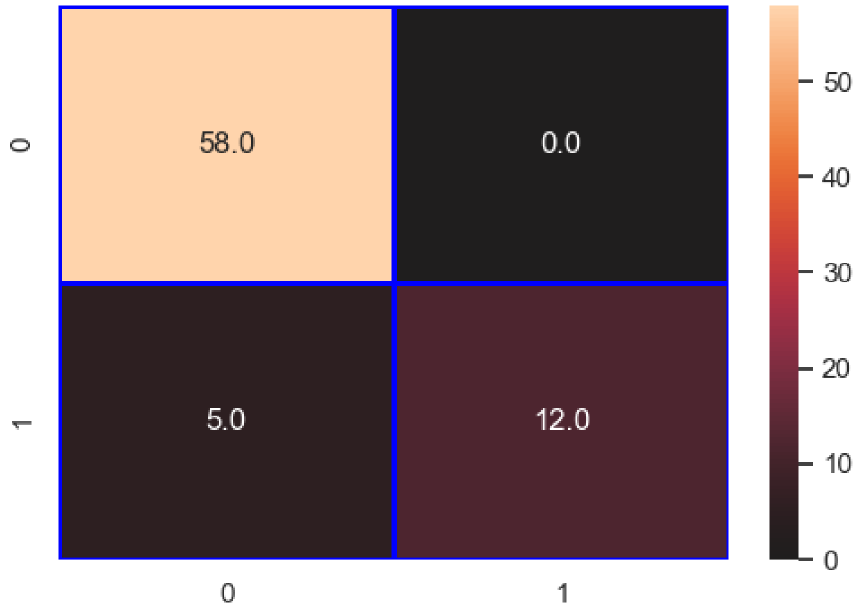
```
Root Mean Square Error (RMSE): 0.4619
===========================================================================================



Modele name : GaussianNB
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]]
```



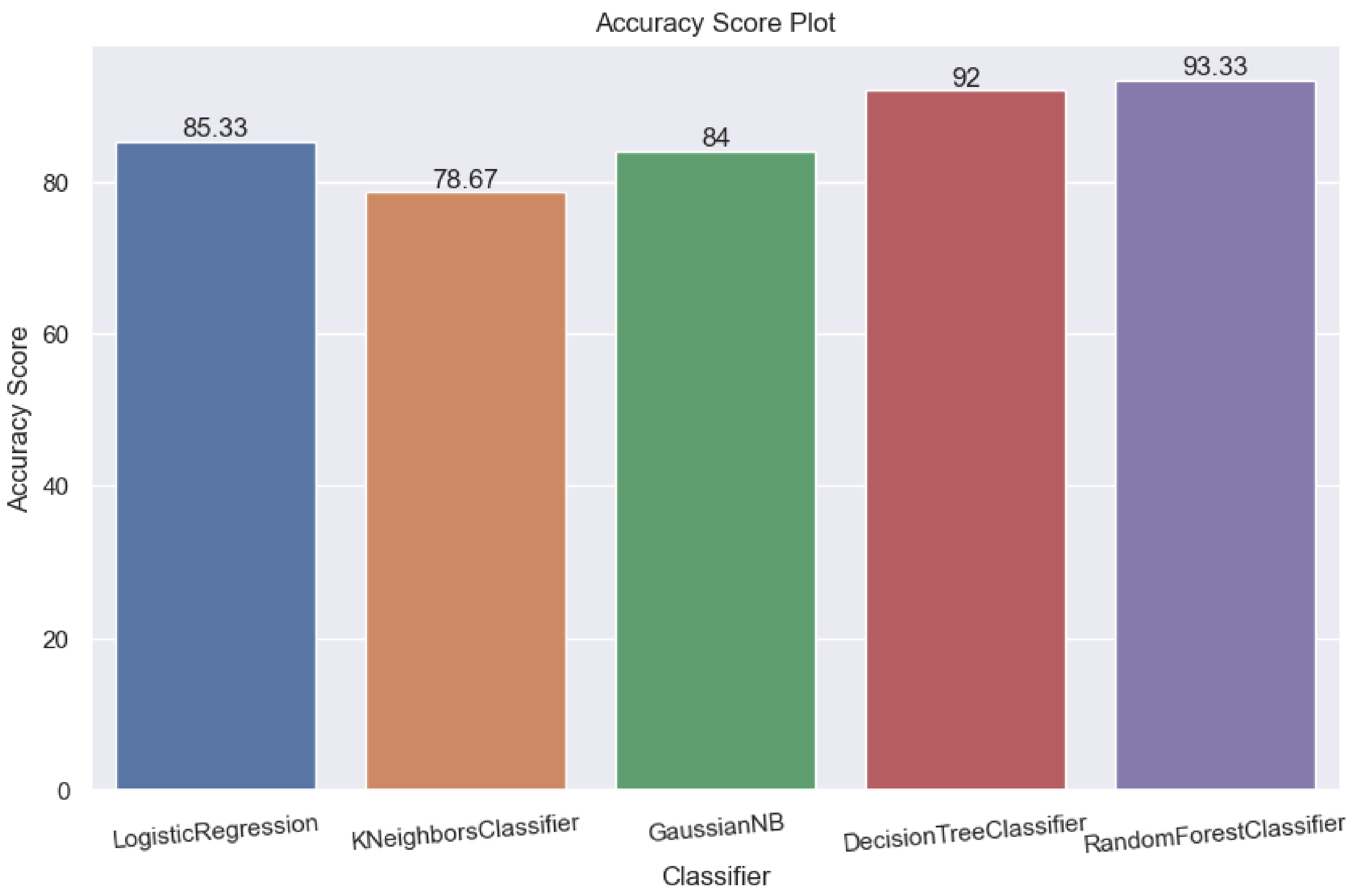
```
Root Mean Square Error (RMSE): 0.4
===========================================================================================



Modele name : DecisionTreeClassifier
10 K-Fold Accuracy_score : [0.9667 0.9    0.9667 0.9    0.8333 0.9667 0.9    0.9    0.9    0.9    ]
10 K-Fold Average Accuracy_score : 91.33 %
Accuracy_score: 92.0 %
Loss: 8.0 %
Cohen_kappa_score: 76.19 %
Classification_report:
               precision    recall  f1-score   support

           0       0.93      0.97      0.95        58
           1       0.87      0.76      0.81        17

    accuracy                           0.92        75
   macro avg       0.90      0.87      0.88        75
weighted avg       0.92      0.92      0.92        75

confusion_matrix:
 [[56  2]
 [ 4 13]]
```

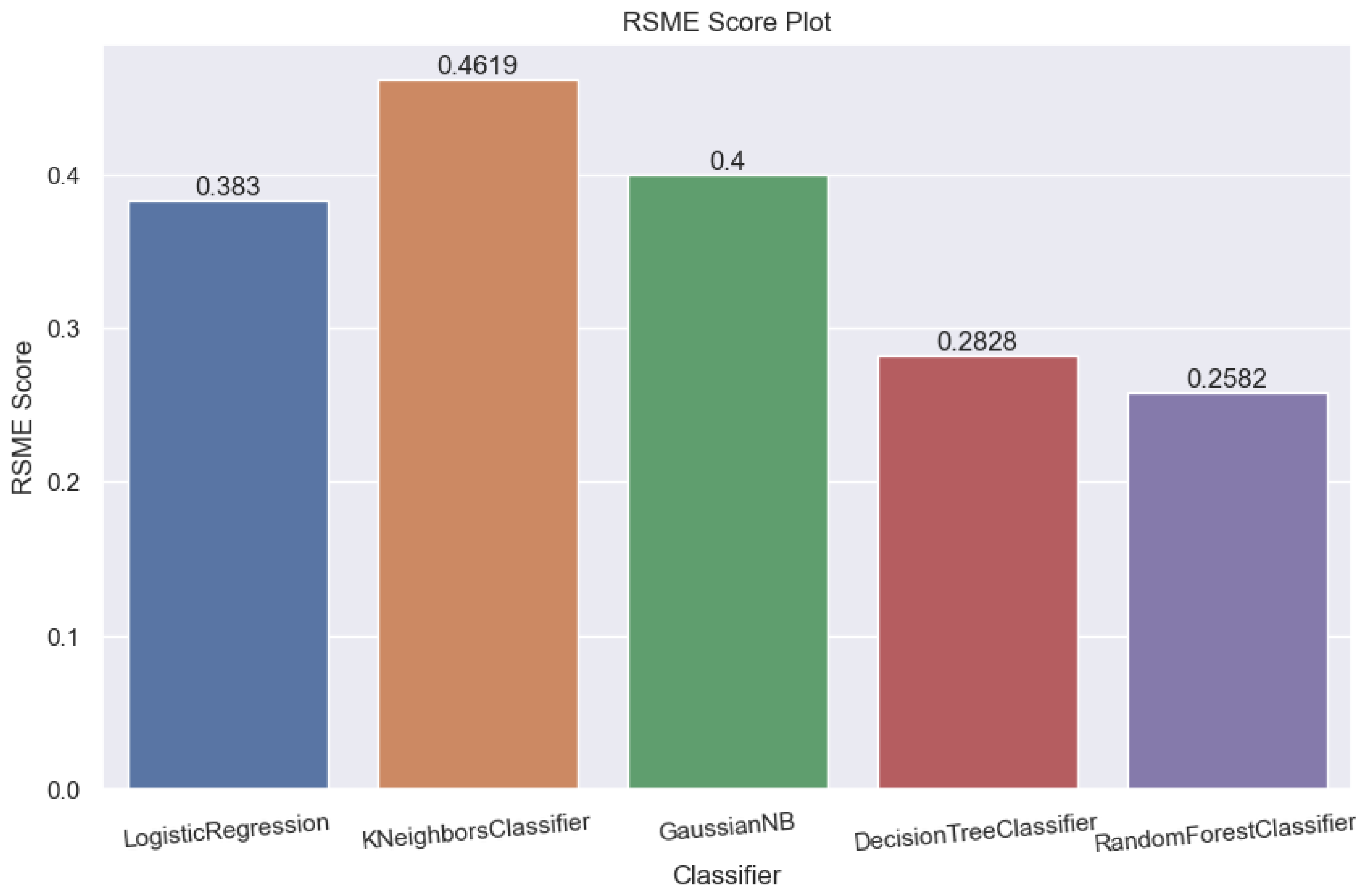


```
Root Mean Square Error (RMSE): 0.2828
==========================================================================================



Modele name : RandomForestClassifier
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: 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]]
```



```
Root Mean Square Error (RMSE): 0.2582
==================================================================================
```




RSME Score Plot
RSME Score
0.0
0.1
0.2
0.3
0.4
0.383
0.4619
0.4
0.2828
0.2582
LogisticRegression
KNeighborsClassifier
GaussianNB
DecisionTreeClassifier
RandomForestClassifier
Classifier

```
Done...
```

In [ ]:

```
1
```