AIML

Obesity level

prepared By :23BCE192 & 23BCE190

Guided By : Rebakah mam

About the topic

In this dataset there is a information about the obesity level.
In this daset 2111 people and 17 various detais about the people.
Its data domain is a health department
We get this dataset from Kaggel.

About the topic

Attributes

Gender: Identifies the gender of the individual (e.g., Male or Female).
Age: Indicates the age of the individual.
Height: Specifies the height of the individual.
Weight: Indicates the weight of the individual.
Family History with Overweight: Indicates whether the individual has a family history of overweight.
FAVC (Frequent Consumption of High Caloric Food): Indicates frequent consumption of high-caloric food.
FCVC (Frequency of Consumption of Vegetables): Represents the frequency of vegetable consumption.
NCP (Number of Main Meals): Specifies the number of main meals consumed by the individual.
CAEC (Consumption of Food Between Meals): Indicates the frequency of food consumption between meals.
CH2O (Consumption of Water Daily): Represents the daily consumption of water.
SCC (Calories Consumption Monitoring): Indicates whether the individual monitors calorie consumption.
FAF (Physical Activity Frequency): Represents the frequency of physical activity.
TUE (Time Using Technology Devices): Specifies the time spent using technology devices.
CALC (Consumption of Alcohol): Indicates alcohol consumption by the individual.
MTRANS (Transportation Used): Specifies the mode of transportation used by the individual.
NObeyesdad (Obesity Level): Represents the individual's obesity level (Target).

The Data Characteristics

1.Null Values

Age	0
Gender	0
Height	0
Weight	0
CALC	0
FAVC	0
FCVC	0
NCP	0
SCC	0
SMOKE	0
CH2O	0
family_history_with_overweight	0
FAF	0
TUE	0
CAEC	0
MTRANS	211
NObeyesdad	0

2.Not Null Values

Column	Non-Null Count
Age	2111 non-null
Gender	2111 non-null
Height	2111 non-null
Weight	2111 non-null
CALC	2111 non-null
FAVC	2111 non-null
FCVC	2111 non-null
NCP	2111 non-null
SCC	2111 non-null
SMOKE	2111 non-null
CH2O	2111 non-null
family_history_with_overweight	2111 non-null
FAF	2111 non-null
TUE	2111 non-null
CAEC	2111 non-null
MTRANS	1900 non-null
NObeyesdad	2111 non-null

The Data Characteristics

Unique values

```
1402
['Female' 'Male']
1574
1525
['no' 'Sometimes' 'Frequently' 'Always']
['no' 'yes']
810
635
['no' 'yes']
['no' 'yes']
1268
['yes' 'no']
1190
1026
['Sometimes' 'Frequently' 'Always' 'no']
['Public_Transportation' 'Walking' 'Automobile' 'Motorbike' 'Bike']
['Normal_Weight' 'Overweight_Level_I' 'Overweight_Level_II'
 'Obesity_Type_I' 'Insufficient_Weight' 'Obesity_Type_II'
 'Obesity_Type_III']
```

Data types

```
Age                               float64
Gender                             object
Height                            float64
Weight                            float64
CALC                               object
FAVC                               object
FCVC                              float64
NCP                               float64
SCC                                object
SMOKE                              object
CH2O                              float64
family_history_with_overweight     object
FAF                               float64
TUE                               float64
CAEC                               object
MTRANS                             object
NObeyesdad                         object
dtype: object
```

The output/class label of the data set

Class label is NObeyesdad

Its Unique value is 'Normal_Weight' 'Overweight_Level_I'
'Overweight_Level_II'
 'Obesity_Type_I' 'Insufficient_Weight' 'Obesity_Type_II'
 'Obesity_Type_III'

Q.2 Load the dataset, drop all the null records and replace the NA values in the numerical column with the mean value of the field as per the class label and categorical columns with the mode value of the field as per the class label.

Code:

```
age_mean=np.mean(data.loc[:,"Age"])
gender_mode=(data.loc[:,"Gender"]).mode
height_mean=np.mean(data.iloc[:,2])
weight_mean=np.mean(data.iloc[:,3])
clac_mode=(data.iloc[:,4]).mode
faac_mode=(data.iloc[:,5]).mode
fcvc_mean=np.mean(data.iloc[:,6])
ncp_mean=np.mean(data.iloc[:,7])
scc_mode=(data.iloc[:,8]).mode
smoke_mode=(data.iloc[:,9]).mode
ch20_mean=np.mean(data.iloc[:,10])
family_history_with_overweight_mode=(data.iloc[:,11]).mode
faf_mean=np.mean(data.iloc[:,12])
tue_mean=np.mean(data.iloc[:,13])
caec_mode=(data.iloc[:,14]).mode
mtrans_mode=(data.iloc[:,15]).mode
NObeyesdad_mode=(data.iloc[:,16]).mode
data.iloc[data.iloc[:,0].isnull(),0]=age_mean
```

```
data.iloc[data.iloc[:,0].isnull(),0]=age_mean
data.iloc[data.iloc[:,1].isnull(),1]=gender_mode
data.iloc[data.iloc[:,2].isnull(),2]=height_mean
data.iloc[data.iloc[:,3].isnull(),3]=weight_mean
data.iloc[data.iloc[:,4].isnull(),4]=clac_mode
data.iloc[data.iloc[:,5].isnull(),5]=faac_mode
data.iloc[data.iloc[:,6].isnull(),6]=fcvc_mean
data.iloc[data.iloc[:,7].isnull(),7]=ncp_mean
data.iloc[data.iloc[:,8].isnull(),8]=scc_mode
data.iloc[data.iloc[:,9].isnull(),9]=smoke_mode
data.iloc[data.iloc[:,10].isnull(),10]=ch20_mean
data.iloc[data.iloc[:,11].isnull(),11]=family_history_with_overweight_mode
data.iloc[data.iloc[:,12].isnull(),12]=faf_mean
data.iloc[data.iloc[:,13].isnull(),13]=tue_mean
data.iloc[data.iloc[:,14].isnull(),14]=caec_mode
data.iloc[data.iloc[:,15].isnull(),15]=mtrans_mode
data.iloc[data.iloc[:,16].isnull(),16]=NObeyesdad_mode
print(data.info())
```

Output:

```
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Age                             2111 non-null   float64
 1   Gender                          2111 non-null   object
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   CALC                            2111 non-null   object
 5   FAVC                            2111 non-null   object
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   SCC                             2111 non-null   object
 9   SMOKE                           2111 non-null   object
 10  CH2O                            2111 non-null   float64
 11  family_history_with_overweight  2111 non-null   object
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CAEC                            2111 non-null   object
 15  MTRANS                          2111 non-null   object
 16  NObeyesdad                      2111 non-null   object
dtypes: float64(8), object(9)
memory usage: 280.5+ KB
None
```

In this code we use numpy library for mean. we use loc and iloc. we find mean and mode and after assign this value to the null values.

Q.3 Perform statistical analysis on the selected dataset (count, sum, range, min, max, mean, median, mode, variance and Standard deviation).

code :

```
# 3
print(data.describe())
```

Output :

	Age	Height	Weight	FCVC	NCP	\
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	
mean	24.312600	1.701677	86.586058	2.419043	2.685628	
std	6.345968	0.093305	26.191172	0.533927	0.778039	
min	14.000000	1.450000	39.000000	1.000000	1.000000	
25%	19.947192	1.630000	65.473343	2.000000	2.658738	
50%	22.777890	1.700499	83.000000	2.385502	3.000000	
75%	26.000000	1.768464	107.430682	3.000000	3.000000	
max	61.000000	1.980000	173.000000	3.000000	4.000000	

	CH2O	FAF	TUE
count	2111.000000	2111.000000	2111.000000
mean	2.008011	1.010298	0.657866
std	0.612953	0.850592	0.608927
min	1.000000	0.000000	0.000000
25%	1.584812	0.124505	0.000000
50%	2.000000	1.000000	0.625350
75%	2.477420	1.666678	1.000000
max	3.000000	3.000000	2.000000

Code :

```
print("Sum of Age :" ,np.sum(data.iloc[:,0]))
print("med of Age :" ,np.median(data.iloc[:,0]))
print("mode of Age :" ,data.iloc[:,0].mode())
print("var of Age :" ,np.var(data.iloc[:,0]))
print("Std of Age :" ,np.std(data.iloc[:,0]))
max0=np.max(data.iloc[:,0])
min0=np.min(data.iloc[:,0])
print("Range of Age : ", max0-min0)


print("Sum of Height  :" ,np.sum(data.iloc[:,2]))
print("med of Height :" ,np.median(data.iloc[:,2]))
print("mode of Height :" ,data.iloc[:,2].mode())
print("var of Height :" ,np.var(data.iloc[:,2]))
print("Std of Height :" ,np.std(data.iloc[:,2]))
max0=np.max(data.iloc[:,2])
min0=np.min(data.iloc[:,2])
print("Range of Height  : ", max0-min0)

print("Sum of Weight  :" ,np.sum(data.iloc[:,3]))
print("med of Weight :" ,np.median(data.iloc[:,3]))
print("mode of Weight :" ,data.iloc[:,3].mode())
print("var of Weight :" ,np.var(data.iloc[:,3]))
print("Std of Weight :" ,np.std(data.iloc[:,3]))
max0=np.max(data.iloc[:,3])
min0=np.min(data.iloc[:,3])
print("Range of Weight : ", max0-min0)
```

Output :

```
Sum of Age : 51323.898407
med of Age : 22.77789
mode of Age : 0    18.0
Name: Age, dtype: float64
var of Age : 40.252236441907115
Std of Age : 6.344465024090456
Range of Age :  47.0
Sum of Height  : 3592.240893
med of Height : 1.700499
mode of Height : 0    1.7
Name: Height, dtype: float64
var of Height : 0.008701665398547784
Std of Height : 0.093282717576986 28
Range of Height  :  0.53
Sum of Weight  : 182783.16870500002
med of Weight : 83.0
mode of Weight : 0    80.0
Name: Weight, dtype: float64
var of Weight : 685.6525235841625
Std of Weight : 26.184967511611745
Range of Weight :  134.0
```

Code :

```
print("Sum of FCVC :" ,np.sum(data.iloc[:,6]))
print("med of FCVC :" ,np.median(data.iloc[:,6]))
print("mode of FCVC :" ,data.iloc[:,6].mode())
print("var of FCVC :" ,np.var(data.iloc[:,6]))
print("Std of FCVC :" ,np.std(data.iloc[:,6]))
max0=np.max(data.iloc[:,6])
min0=np.min(data.iloc[:,6])
print("Range of FCVC : ", max0-min0)

print("Sum of NCP :" ,np.sum(data.iloc[:,7]))
print("med of NCP :" ,np.median(data.iloc[:,7]))
print("mode of NCP :" ,data.iloc[:,7].mode())
print("var of NCP :" ,np.var(data.iloc[:,7]))
print("Std of NCP :" ,np.std(data.iloc[:,7]))
max0=np.max(data.iloc[:,7])
min0=np.min(data.iloc[:,7])
print("Range of NCP : ", max0-min0)

print("Sum of CH2O :" ,np.sum(data.iloc[:,10]))
print("med of CH2O :" ,np.median(data.iloc[:,10]))
print("mode of CH2O :" ,data.iloc[:,10].mode())
print("var of CH2O :" ,np.var(data.iloc[:,10]))
print("Std of CH2O :" ,np.std(data.iloc[:,10]))
max0=np.max(data.iloc[:,10])
min0=np.min(data.iloc[:,10])
print("Range of CH2O : ", max0-min0)
```

Output :

```
Sum of FCVC : 5106.599903
med of FCVC : 2.385502
mode of FCVC : 0    3.0
Name: FCVC, dtype: float64
var of FCVC : 0.28494254737092756
Std of FCVC : 0.5338001005722344
Range of FCVC :  2.0
Sum of NCP : 5669.360812999999
med of NCP : 3.0
mode of NCP : 0    3.0
Name: NCP, dtype: float64
var of NCP : 0.6050573820385702
Std of NCP : 0.7778543450020513
Range of NCP :  3.0
Sum of CH2O : 4238.912074
med of CH2O : 2.0
mode of CH2O : 0    2.0
Name: CH2O, dtype: float64
var of CH2O : 0.3755339558915529
Std of CH2O : 0.6128082537723794
Range of CH2O :  2.0
```

Code :

```
print("Sum of FAF :" ,np.sum(data.iloc[:,12]))
print("med of FAF :" ,np.median(data.iloc[:,12]))
print("mode of FAF :" ,data.iloc[:,12].mode())
print("var of FAF :" ,np.var(data.iloc[:,12]))
print("Std of FAF :" ,np.std(data.iloc[:,12]))
max0=np.max(data.iloc[:,12])
min0=np.min(data.iloc[:,12])
print("Range of FAF : ", max0-min0)

print("Sum of TUE :" ,np.sum(data.iloc[:,13]))
print("med of TUE :" ,np.median(data.iloc[:,13]))
print("mode of TUE :" ,data.iloc[:,13].mode())
print("var of TUE :" ,np.var(data.iloc[:,13]))
print("Std of TUE :" ,np.std(data.iloc[:,13]))
max0=np.max(data.iloc[:,13])
min0=np.min(data.iloc[:,13])
print("Range of TUE : ", max0-min0)
```

Output:

```
Sum of FAF : 2132.738436
med of FAF : 1.0
mode of FAF : 0    0.0
Name: FAF, dtype: float64
var of FAF : 0.7231647512870681
Std of FAF : 0.850390940266339
Range of FAF :  3.0
Sum of TUE : 1388.754965
med of TUE : 0.62535
mode of TUE : 0    0.0
Name: TUE, dtype: float64
var of TUE : 0.3706167598235124
Std of TUE : 0.6087830153868555
Range of TUE :  2.0
```

In this code we use numpy library finding the statistical value .
Using describe function to get statistic value.

Code :

```
print("Mode of Gender",(data.loc[:,"Gender"]).mode())
print("Mode of CLAC ", data.iloc[:,4].mode())
print("Mode of FAVC", data.iloc[:,5].mode())
print("Mode of SCC", data.iloc[:,8].mode())
print("Mode of SMOKE", data.iloc[:,9].mode())
print("Mode of family_history_with_overweight", data.iloc[:,11].mode())
print("Mode of CAEC", data.iloc[:,14].mode())
print("Mode of MTRANS", data.iloc[:,15].mode())
print("Mode of NObeyesdad", data["NObeyesdad"].mode())
```

Output :

```
Mode of Gender 0    Male
Name: Gender, dtype: object
Mode of CLAC  0    Sometimes
Name: CALC, dtype: object
Mode of FAVC 0    yes
Name: FAVC, dtype: object
Mode of SCC 0    no
Name: SCC, dtype: object
Mode of SMOKE 0    no
Name: SMOKE, dtype: object
Mode of family_history_with_overweight 0    yes
Name: family_history_with_overweight, dtype: object
Mode of CAEC 0    Sometimes
Name: CAEC, dtype: object
Mode of MTRANS 0    Public_Transportation
Name: MTRANS, dtype: object
Mode of NObeyesdad 0    Obesity_Type_I
Name: NObeyesdad, dtype: object
```

Q. 3 Display all the unique value counts and unique values of all the columns of the dataset.

Code :

```
print(" Unique count of Age ",data.iloc[:,0].nunique())
print(" Unique count of Gender ",data.iloc[:,1].nunique())
print(" Unique count of Height ",data.iloc[:,2].nunique())
print(" Unique count of Weight ",data.iloc[:,3].nunique())
print(" Unique count of CALC ",data.iloc[:,4].nunique())
print(" Unique count of FAVC ",data.iloc[:,5].nunique())
print(" Unique count of FCVC ",data.iloc[:,6].nunique())
print(" Unique count of NCP ",data.iloc[:,7].nunique())
print(" Unique count of SCC ",data.iloc[:,8].nunique())
print(" Unique count of SMOKE ",data.iloc[:,9].nunique())
print(" Unique count of CH2O ",data.iloc[:,10].nunique())
print(" Unique count of family_history_with_overweight ",data.iloc[:,11].nunique())
print(" Unique count of FAF ",data.iloc[:,12].nunique())
print(" Unique count of TUE ",data.iloc[:,13].nunique())
print(" Unique count of CAEC ",data.iloc[:,14].nunique())
print(" Unique count of MTRANS ",data.iloc[:,15].nunique())
print(" Unique count of NObeyesdad ",data.iloc[:,16].nunique())
```

Output :

```
Unique count of Age  1402
Unique count of Gender  2
Unique count of Height  1574
Unique count of Weight  1525
Unique count of CALC  5
Unique count of FAVC  2
Unique count of FCVC  810
Unique count of NCP  635
Unique count of SCC  2
Unique count of SMOKE  2
Unique count of CH2O  1268
Unique count of family_history_with_overweight  2
Unique count of FAF  1190
Unique count of TUE  1129
Unique count of CAEC  4
Unique count of MTRANS  5
Unique count of NObeyesdad  7
```

In this code we calculated the number of unique values .

We use the nunique function .

Code :

```
print(" Unique values of Age ",data.iloc[:,0].unique())
print(" Unique values of Gender ",data.iloc[:,1].unique())
print(" Unique values of Height ",data.iloc[:,2].unique())
print(" Unique values of Weight ",data.iloc[:,3].unique())
print(" Unique values of CALC ",data.iloc[:,4].unique())
print(" Unique values of FAVC ",data.iloc[:,5].unique())
print(" Unique values of FCVC ",data.iloc[:,6].unique())
print(" Unique values of NCP ",data.iloc[:,7].unique())
print(" Unique values of SCC ",data.iloc[:,8].unique())
print(" Unique values of SMOKE ",data.iloc[:,9].unique())
print(" Unique values of CH2O ",data.iloc[:,10].unique())
print(" Unique values of family_history_with_overweight ",data.iloc[:,11].unique())
print(" Unique values of FAF ",data.iloc[:,12].unique())
print(" Unique values of TUE ",data.iloc[:,13].unique())
print(" Unique values of CAEC ",data.iloc[:,14].unique())
print(" Unique values of MTRANS ",data.iloc[:,15].unique())
print(" Unique values of NObeyesdad ",data.iloc[:,16].unique())
```

Output :

```
Unique values of FAVC  ['no' 'yes']
Unique values of FCVC  [2.         3.         2.41887399 1.         2.450218   2.880161
2.00876    2.596579   2.591439   2.392665   1.123939   2.027574
2.88626    2.714447   2.750715   1.4925     2.205439   2.059138
2.310423   2.823179   2.052932   2.767731   2.815157   2.737762
2.568063   2.524428   2.971574   1.0816     1.270448   1.344854
2.959658   2.844607   2.44004    2.592247   2.929889   2.015258
1.031149   1.592183   1.21498    1.522001   2.703436   2.14084
2.336044   1.813234   2.724285   2.71897    1.133844   1.757466
2.979383   2.204914   2.927218   2.88853    2.530066   2.241606
1.003566   2.652779   2.897899   2.483979   2.945967   2.478891
1.005578   2.938031   2.842102   1.889199   2.943749   1.950742
2.277436   2.371338   2.984425   2.663421   2.753752   2.318355
2.594653   2.886157   2.619835   1.053534   2.530233   2.8813
2.824559   2.762325   2.070964   2.68601    2.794197   2.720701
2.880792   2.55996    1.212908   1.140615   2.562409   2.004146
2.690754   2.19005    2.21498    2.708965   2.853513   2.580872
2.508835   2.896562   2.911877   2.910733   2.966126   2.613249
2.627031   2.919751   2.494451   1.69427    1.601236   1.204855
1.052699   2.910345   2.866383   2.913486   2.432886   2.883745
2.707666   2.919584   2.486189   1.642241   1.567101   1.036414
1.649974   1.118436   2.673638   2.120185   2.34222    2.86099
2.559571   2.424977   1.786841   1.303878   1.889883   2.984004
2.749268   1.202075   1.871213   2.341133   1.450218   2.81646
1.758394   2.577427   2.052152   2.954996   2.555401   2.108711
2.915279   1.570089   1.94313    2.903545   1.75375    2.543563
2.39728    2.37464    2.278644   2.061952   2.838969   2.652958
1.27785    1.729824   1.452524   2.303367   2.948425   2.291846
1.906194   1.834155   2.293705   2.510583   2.366949   2.615788
2.801514   2.188722   2.971351   2.086093   1.901611   1.977298
2.446872   2.839048   2.21232    2.427689   2.357496   1.078529
1.064162   1.993101   2.620963   2.95118    2.021446   2.000466
2.5621     2.96008    2.53915    2.244142   2.253371   2.851664
1.31415    1.321028   2.253998   2.778079   2.838037   2.814453
2.013782   2.459976   2.643183   2.22399    2.104105   1.972545
2.286481   2.971588   2.872121   2.109162   2.178889   1.142468
2.047069   2.416044   1.766849   1.188089   1.910176   2.956671
```

```
2.002796   2.288604   2.277077   2.138334   2.029634   2.048216
2.8557     2.995599   2.987148   1.887951   2.786008   2.342323
1.874935   2.780699   1.687569   1.989905   1.947405   2.162519
2.923916   2.507841   1.836554   1.773265   2.388168   2.487167
2.185938   2.206399   1.952987   2.908757   2.628791   2.749629
1.595746   2.885178   2.372494   2.153639   2.793561   2.992329
2.927409   2.706134   2.010684   2.300408   2.119643   2.901924
2.451009   2.417635   2.512719   1.771693   1.57223    2.661556
2.097373   2.061461   1.317729   1.882235   2.951591   2.54527
2.694281   2.821977   2.252472   2.033745   2.595128   2.759286
1.925064   2.846981   2.650629   2.631565   2.522399   2.784471
1.650505   1.961347   2.133955   2.684528   2.265973   1.306844
2.258795   2.689929   2.712747   2.598051   1.718156   2.795086
2.030256   2.442536   2.003951   1.34138    2.607335   2.061384
2.696381   1.116068   2.116432   2.031246   2.884212   2.976975
2.766036   1.096455   2.129969   2.019674   2.735297   1.588782
2.01054    2.689577   2.923433   2.882522   1.973499   2.156065
2.992205   2.944287   1.780746   2.164062   2.247704   2.011656
2.03414    2.1239     2.332074   2.748243   2.303656   1.904732
1.979944   2.407817   2.734314   2.784383   2.009952   2.008656
1.122127   1.963965   2.319776   1.899116   2.042762   2.129668
2.182401   2.802696   2.492758   2.733129   2.294067   2.315932
2.802128   2.79606    2.086898   2.333503   1.653081   2.535315
2.765769   2.215464   2.63165    2.942154   2.716909   1.83746
2.045027   2.760607   2.09449    2.907062   2.399531   1.521604
2.501683   2.76802    2.742796   2.325623   2.271306   2.33361
2.909853   2.490507   1.063449   2.699282   2.87599    2.177243
2.846452   2.19011    1.362441   2.09663    2.496455   1.70825
2.262171   2.740633   1.081585   2.587789   2.252653   2.609123
1.036159   1.851262   2.501236   2.031185   2.014194   2.028225
2.037585   1.853314   1.00876    2.103335   2.060922   2.533605
2.457547   2.061969   2.450784   2.467002   2.043359   2.150054
2.977585   2.274491   1.557287   2.152264   2.08841    2.180047
1.826885   1.754401   1.524428   2.259679   2.044326   2.843456
2.864776   2.282803   2.020502   2.020785   2.737149   1.899793
2.274164   1.897796   2.85916    2.340405   2.921576   2.92711
2.805533   2.845961   2.252698   2.02091    2.108163   2.260543
```

```
2.522183   2.186322   2.76632    2.071622   1.918251   1.967061
2.964419   2.640801   2.870152   2.482575   2.074843   2.14128
2.334474   2.607747   2.66889    2.921225   1.492834   1.220024
2.667676   2.569075   2.949242   2.104772   2.871768   2.540949
2.222282   2.04516    2.094184   1.123672   2.049112   1.880534
1.203754   2.73691    2.736628   2.121909   2.642744   2.387426
1.996638   2.107854   2.253707   2.393837   2.348745   2.490776
2.078082   2.805512   1.276858   2.663866   2.685484   2.055209
2.025479   2.385502   2.465575   2.818502   2.081238   1.368978
1.3307     1.961069   2.976509   2.319648   1.261288   2.76533
2.317734   1.588114   2.549782   2.206276   1.99124    2.408561
2.758394   2.956297   1.369529   1.392665   2.630401   2.869778
1.868212   2.907744   2.197261   2.048962   2.139196   2.72989
1.994679   2.403421   2.964319   1.617093   1.631144   2.501224
2.983042   2.323351   1.428289   1.992889   1.562804   1.794825
2.305349   2.754645   2.633855   2.200588   2.499108   2.05687
2.247795   2.239634   2.09283    2.9673     2.225731   2.178308
2.065752   2.263245   2.443538   2.744994   2.630137   2.113843
2.821727   2.312528   2.938801   2.050619   1.457758   2.19331
2.028571   1.455602   2.177896   2.252382   2.668949   2.736298
2.722161   2.871016   2.96405    2.653721   2.859097   1.572036
2.941929   2.973569   2.041376   2.02472    2.133964   2.702457
1.996646   2.475892   2.941627   2.998441   2.382705   2.176317
2.303041   2.323003   1.99953    1.585183   2.281963   2.561638
2.600217   2.08868    2.206119   2.328469   2.002784   1.975675
2.002076   2.218599   2.392179   2.381164   2.008245   2.155182
2.969233   2.765063   2.195964   2.619987   2.918113   2.889485
2.939727   1.264234   1.517912   2.032883   2.046651   2.667229
2.819934   2.954417   2.826251   2.935157   2.869833   1.878251
1.750809   1.168856   2.826036   2.9553     2.774562   2.772027
2.499626   2.108638   2.467548   2.585942   2.14961    2.938616
2.611222   1.800122   1.482722   1.735664   1.655684   2.598207
2.225149   2.519592   2.970983   2.907542   1.451337   1.537505
1.893428   2.007845   2.487781   2.541785   2.813775   2.922511
2.836055   2.880759   2.766441   2.997524   2.320201   2.181057
1.924632   1.397468   2.191429   2.100177   2.075321   1.626369
1.263216   1.067909   2.310751   2.294259   2.57649    2.679664
```

```
2.938687   2.654792   2.731368   2.671238   2.57691    2.341999
2.117121   2.247037   2.244654   2.128574   2.068834   2.073224
2.145114   2.997951   2.911312   2.159033   2.175276   2.21965
2.203962   1.947495   2.262292   2.219186   2.314175   2.423291
2.637202   2.974006   2.347942   2.654076   2.739      2.591292
2.766612   2.777165   2.232836   2.571274   2.49619    2.151335
2.01695    1.412566   1.289315   1.624366   1.528331   2.037042
2.191108   1.431346   2.499388   2.230742   2.240757   2.996186
2.649406   2.736647   2.684335   2.425503   1.469384   2.906269
2.808027   2.636719   2.996717   2.880483   2.913452   2.919526
2.724121   2.801992   2.748971   2.680375  ]
Unique values of NCP  [3.       1.       4.       3.28926  3.995147 1.72626  2.581015 1.600812
1.73762  1.10548  2.0846   1.894384 2.857787 3.765526 3.285167 3.691226
3.156153 1.07976  3.559841 3.891994 3.240578 3.904858 3.11158  3.590039
2.057935 3.558637 2.000986 3.821168 3.897078 3.092116 3.286431 3.592415
3.754599 3.566082 3.725797 3.520555 3.731212 1.259803 1.273128 3.304123
3.647154 3.300666 3.535016 1.717608 2.884479 3.626815 1.473088 3.16645
3.494849 2.99321  2.127797 3.90779  3.699594 3.179995 1.075553 3.238258
3.804944 1.630846 3.762778 3.371832 2.705445 3.34175  2.217651 2.893778
3.502604 3.998766 3.193671 1.69608  2.812377 1.612747 1.082304 1.882158
2.326233 1.989398 1.735493 2.974568 3.715118 3.489918 3.378859 3.263201
3.994588 3.24934  3.087544 1.163666 3.409363 3.281391 3.98525  3.207071
3.471536 3.488342 3.443456 3.03779  3.642802 2.645858 3.420618 2.64155
3.887906 3.435905 3.747163 2.625475 3.098399 3.12544  3.96981  3.712183
3.832911 3.576103 3.56544  3.266644 3.433908 3.531038 3.998618 1.226342
1.060796 3.595761 3.737914 3.697831 3.21043  3.45259  3.205587 1.513835
2.779379 3.732126 3.937099 3.047959 2.975362 1.394539 3.131032 1.578521
3.985442 3.623364 3.36313  1.146052 3.981997 1.9154   2.105616 3.714833
3.292386 1.104642 3.654061 1.296156 2.656588 2.809716 2.77684  1.999014
2.644692 2.449723 2.794156 1.146794 1.131695 2.488189 3.999591 3.612941
3.950553 2.951837 2.799979 2.228113 3.042774 1.198643 1.555557 3.987707
3.995957 1.047197 1.193589 3.390143 3.546352 3.266501 3.554974 2.372311
2.10601  3.715306 3.376717 1.211606 3.98955  3.171082 3.105007 1.590982
1.704828 2.725012 2.372339 1.097312 1.068196 1.411685 2.752318 2.737571
2.973504 2.608055 2.625942 1.411808 1.095223 3.30846  3.821461 2.658639
2.339614 1.713762 2.743277 3.051804 3.245148 3.563744 3.829101 3.058539
3.196043 1.391778 1.13715  3.322522 3.269088 2.938135 1.259628 2.965494
```

```
2.845307 3.095663 3.483449 3.156309 3.728377 3.60885  3.370362 2.39007
1.101404 2.756405 3.054899 3.118013 3.335876 3.205009 3.648194 1.865238
2.118153 2.961113 2.400943 2.870005 1.030416 2.696051 2.762883 2.401341
2.298612 3.618722 2.562895 2.849848 2.805436 3.715148 3.788602 1.893811
1.010319 2.669766 1.322087 2.041558 2.468421 2.110937 3.071028 3.292956
2.283673 1.124977 2.9948   2.983297 1.890213 1.888067 2.256119 2.6648
2.547086 2.812283 2.278652 1.240046 1.660768 2.597608 2.044035 2.038373
1.474836 3.986652 2.720642 1.976744 2.679724 1.619796 2.59257  1.546665
3.339914 3.884861 2.358298 1.802305 2.7976   2.627173 1.032887 3.014808
2.271734 2.122545 2.902766 2.687502 1.178708 2.279546 2.677693 3.53009
1.835543 2.853676 2.337035 1.631184 1.005391 3.250467 2.948721 1.80993
1.94671  2.9796   2.994198 2.579291 2.449067 1.532833 2.961706 1.152521
1.729553 1.346987 1.977221 2.443812 2.491315 2.843319 2.157164 2.601675
1.171027 3.362758 1.672706 2.714115 2.521546 1.338033 1.081805 3.087119
2.909117 1.466393 1.471053 2.983201 2.783336 1.863012 3.053598 3.165837
2.741413 1.058123 2.997414 1.672958 1.680838 3.070386 2.608416 1.599464
2.815255 2.395785 2.844138 2.716106 1.402771 2.865657 1.836226 2.884848
2.375026 2.9774   2.693646 2.832018 2.270163 2.646717 1.193729 1.477581
3.209508 2.756622 2.658478 1.971472 1.818026 1.820779 1.724887 3.129155
3.856434 2.753418 2.116195 2.049908 1.044628 2.98212  1.009426 2.667711
3.000974 2.698883 2.598079 2.837388 2.946063 1.873484 2.473911 1.79558
2.623079 2.174968 2.676148 3.394788 2.137068 2.937607 1.313403 2.113575
2.092179 2.974204 1.001633 1.2919   1.139317 2.029858 2.977543 1.476204
2.57038  2.879541 1.458507 1.105617 1.854536 1.08687  1.25535  2.870661
1.482103 1.81698  2.582591 2.164839 2.141839 2.877583 2.95833  2.119826
2.992606 2.050121 1.923607 2.27374  1.418985 1.320768 2.964024 2.933409
2.962004 1.271624 1.00061  1.068443 2.952821 1.851088 2.119682 2.970675
2.966803 1.508685 2.209314 1.114564 2.036794 2.988539 1.630506 2.977999
2.733077 2.427137 1.867836 1.548407 2.937989 1.237454 1.169173 2.391753
2.475444 2.478794 2.070033 2.282392 1.049534 1.66338  1.672751 1.734762
1.92822  2.475228 2.093831 1.231915 1.374791 1.326982 2.900915 1.317884
2.977909 1.355752 1.015488 2.986172 2.993623 1.001383 1.001542 2.590283
1.773916 2.989112 1.496776 2.994046 1.000283 1.000414 1.135278 2.463113
1.782109 2.376374 2.976098 2.968098 1.792695 1.116401 2.984523 2.978103
1.578751 1.874532 2.735706 1.014916 2.622055 2.806566 1.355354 1.478334
1.130751 1.099151 2.358455 1.706551 1.24884  1.250548 1.202179 1.194815
2.880817 2.918124 1.073421 1.416309 2.18162  2.152733 1.046144 1.974233
```

```
2.880794 2.765213 2.967089 2.37985  2.655265 1.941307 1.709546 2.883984
2.935381 2.961192 2.973476 2.392811 1.293342 2.930044 2.129909 2.831771
2.595126 2.52751  2.120936 1.468948 3.990925 2.834253 3.734323 2.049565
2.272214 2.218285 1.418833 1.109956 3.914454 3.169089 2.419656 2.175432
2.806298 2.987652 2.842848 1.134321 1.924168 2.701689 2.992903 2.658837
2.80742  1.834472 2.123138 1.265463 1.340361 3.989492 2.650088 2.89292
2.938902 2.986637 2.701521 2.014671 1.971659 1.311797 1.262831 3.770379
2.87747  2.175153 2.13229  2.993634 2.999346 2.976211 2.842035 1.09749
1.91863  2.510135 2.65772  2.604998 2.567567 2.434347 1.845858 1.627555
1.569176 2.371658 2.378211 2.301129 2.454432 2.915921 2.992083 2.849347
2.954446 3.755976 2.902639 2.372705 2.100918 2.272801 2.692889 2.280545
2.675411 1.487674 1.02075  1.240424 1.154318 3.219347 3.656401 3.220181
2.850948 2.187145 2.415522 2.138375 1.077331 2.888193 2.574108 2.740492
2.956422 2.749334 2.993856 2.993084 2.989791 2.920373 2.911568 1.134042
1.120102 2.040582 2.015675 2.711238 2.695396 2.996834 2.996444 2.894142
2.683061 2.806341 2.791366 1.703299 1.685134 2.269799 2.142328 1.437959
1.343117 1.213431 1.089048]
Unique values of SCC  ['no' 'yes']
Unique values of SMOKE  ['no' 'yes']
Unique values of CH2O  [2.       3.       1.       ... 2.054193 2.852339 2.863513]
Unique values of family_history_with_overweight  ['yes' 'no']
Unique values of FAF  [0.       3.       2.       ... 1.414209 1.139107 1.026452]
Unique values of TUE  [1.       0.       2.       ... 0.646288 0.586035 0.714137]
Unique values of CAEC  ['Sometimes' 'Frequently' 'Always' 'no']
Unique values of MTRANS  ['Public_Transportation' 'Walking' 'Automobile' 'Motorbike' 'Bike']
Unique values of NObeyesdad  ['Normal_Weight' 'Overweight_Level_I' 'Overweight_Level_II'
'Obesity_Type_I' 'Insufficient_Weight' 'Obesity_Type_II'
'Obesity_Type_III']
```

In this code we calculated the unique values .

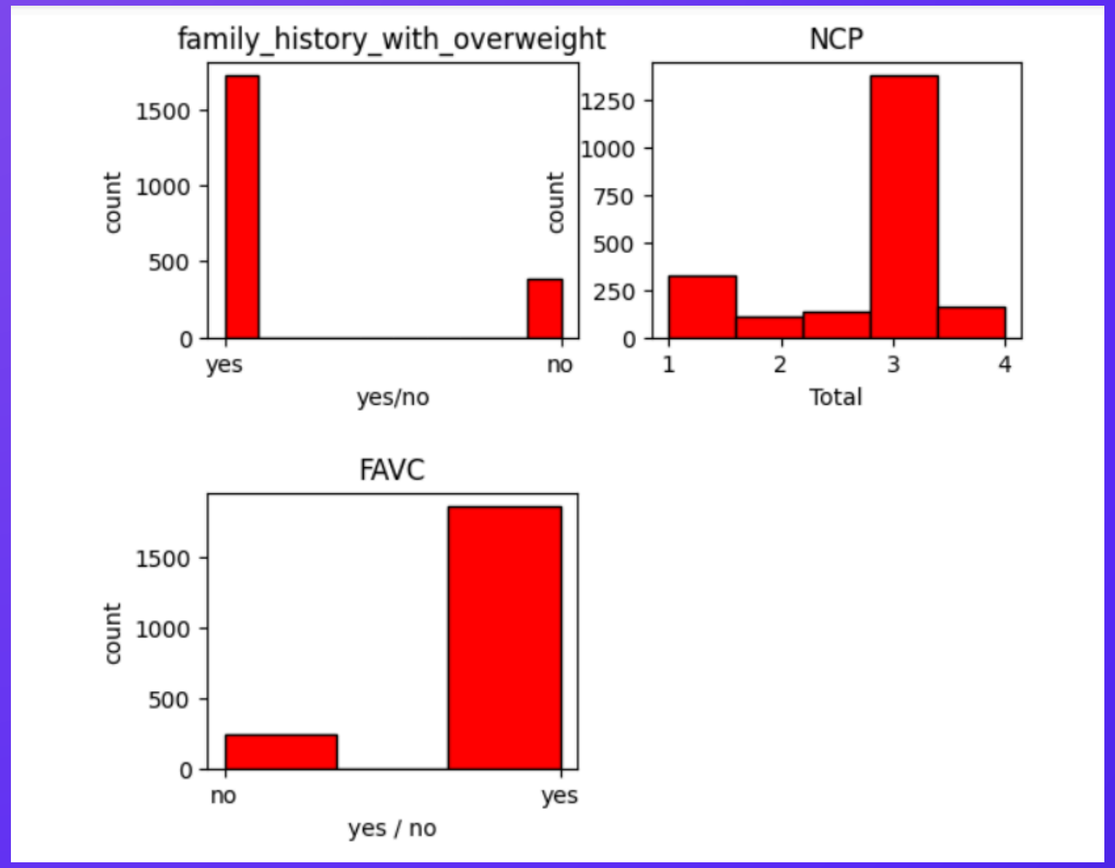
We use the unique function .

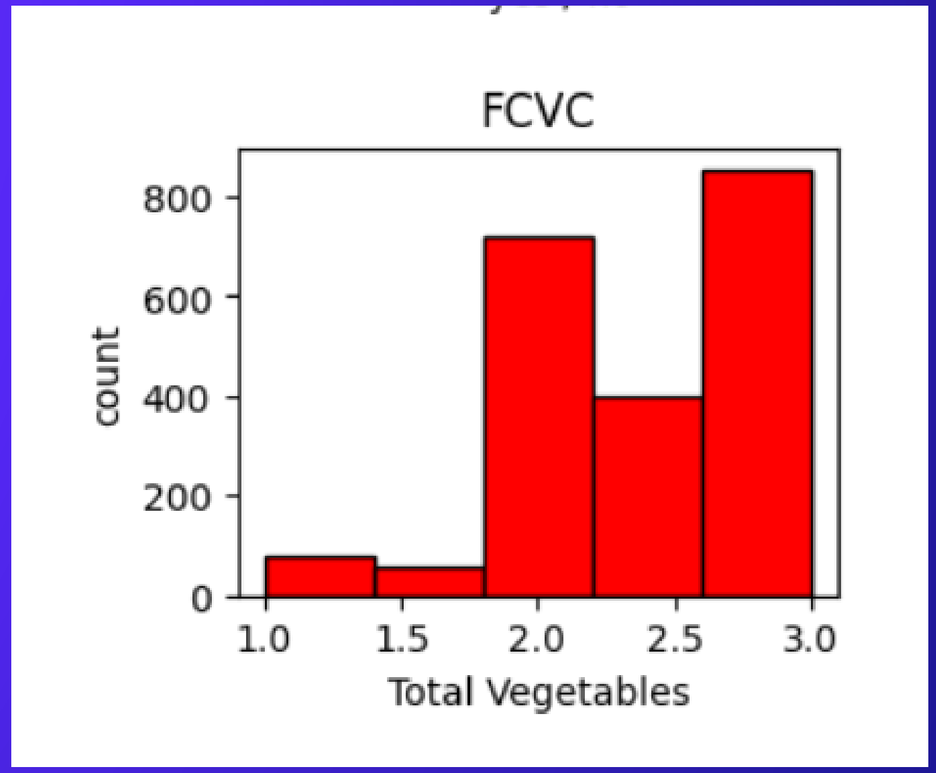
Q.5 Draw applicable plots to visualise data using the subplot concept on the dataset. (scatter plot/ line graph/ histogram etc.)

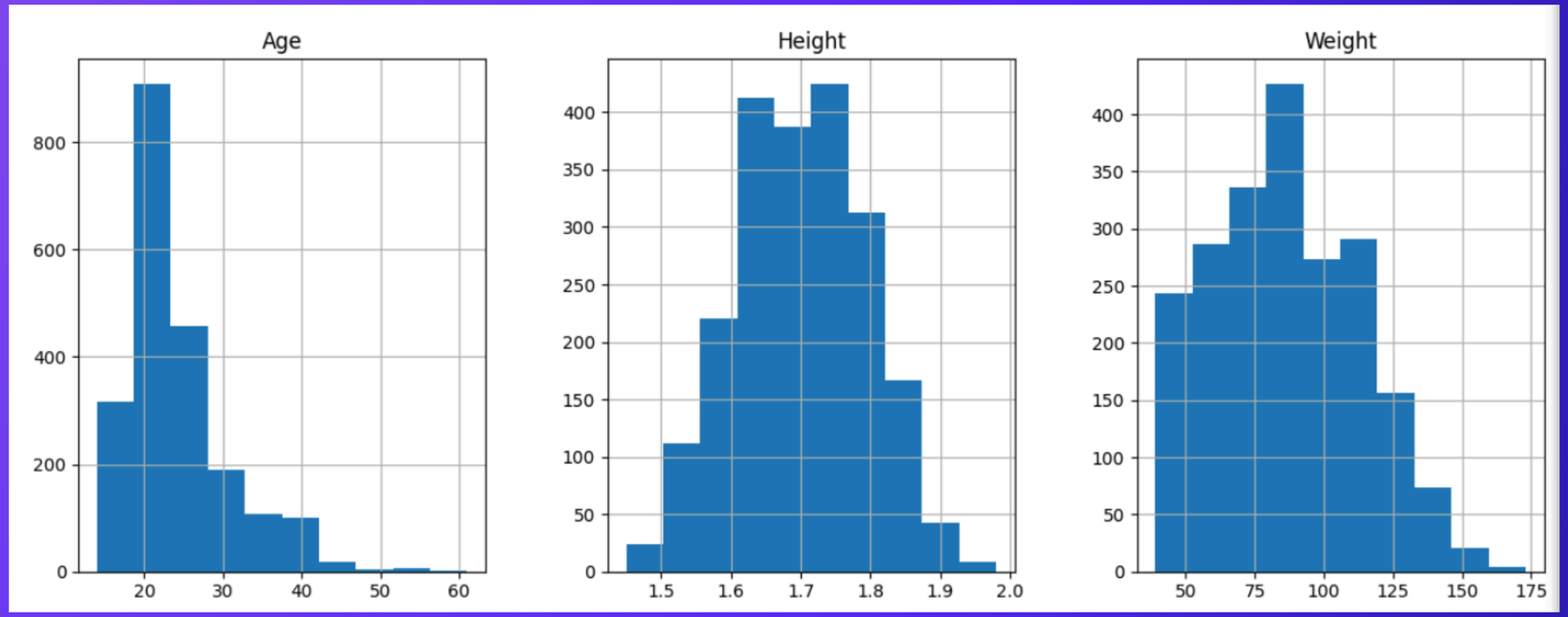
Code :

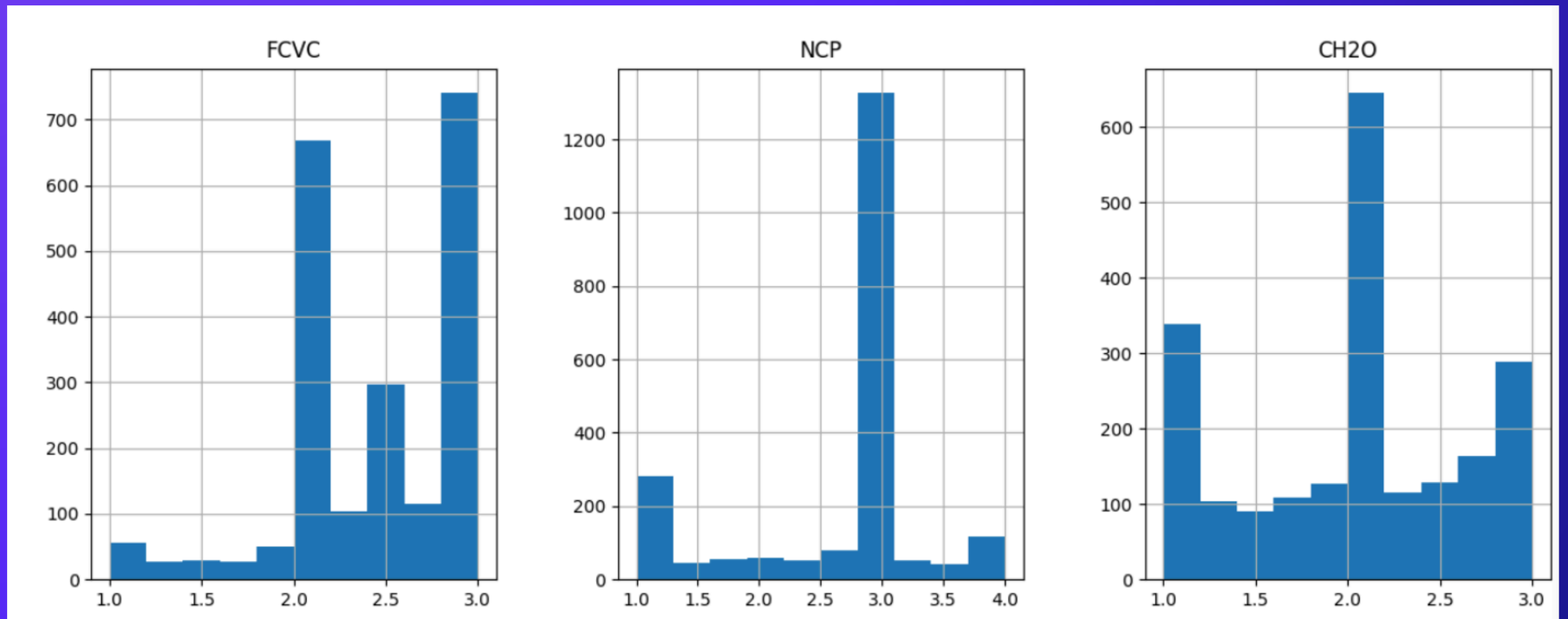
```
from matplotlib import pyplot as plt

plt.subplot(2,2,1)
plt.hist(data['family_history_with_overweight'] , color='red', edgecolor='black')
plt.title("family_history_with_overweight")
plt.xlabel("yes/no")
plt.ylabel("count")

plt.subplot(2,2,2)
plt.hist(data['FAVC'], bins=3, color='red', edgecolor='black')
plt.title('FAVC')
plt.xlabel('yes / no')
plt.ylabel('count')
plt.show()

plt.subplot(2,2,3)
plt.hist(data['FCVC'], bins=5, color='red', edgecolor='black')
plt.title('FCVC')
plt.xlabel('Total Vegetables')
plt.ylabel('count')
plt.show()

plt.subplot(2,2,4)
plt.hist(data['NCP'], bins=5, color='red', edgecolor='black')
plt.title('NCP')
plt.xlabel('Total')
plt.ylabel('count')
plt.show()

data.hist(figsize=(15,19))
plt.show()
```

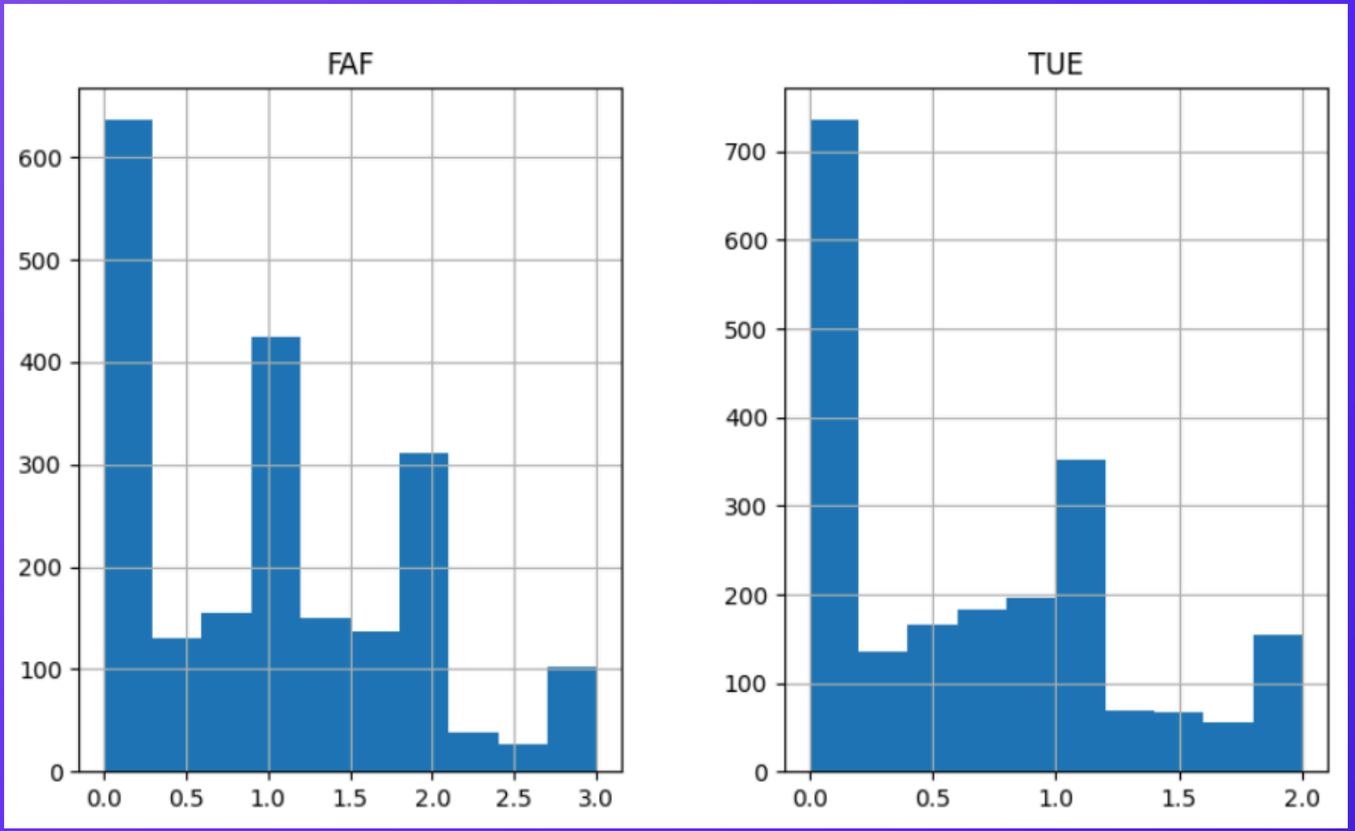
Output :


family_history_with_overweight
count
1500
1000
500
0
yes
no
yes/no
NCP
count
1250
1000
750
500
250
0
1
2
3
4
Total
FAVC
count
1500
1000
500
0
no
yes
yes / no


FCVC
count
800
600
400
200
0
1.0
1.5
2.0
2.5
3.0
Total Vegetables


Age
Height
Weight


FCVC
NCP
CH2O


FAF
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
TUE
700
600
500
400
300
200
100
0
0.0
0.5
1.0
1.5
2.0

Q. 6 Train the model of the K-nearest Neighbors Classifier/Regressor with 80% of the data and predict the class label for the rest 20% of the data.
Evaluate the model with all appropriate measures.

Code :

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.neighbors import KNeighborsClassifier

data2=pd.read_csv("D:\AI&Ml\Assigenment\ObesityDataSet_raw_and_data_sinthetic.csv",header='infer')
print(data2.isnull().sum().sum())

data2['Gender']=data2['Gender'].map({'Male':1,'Female':0})
data2['CALC']=data2['CALC'].map({'Sometimes':1,'Frequently':2,'Always':3,'no':0})
data2['FAVC']=data2['FAVC'].map({'no':1,'yes':0})
data2['SCC']=data2['SCC'].map({'no':1,'yes':0})
data2['SMOKE']=data2['SMOKE'].map({'no':1,'yes':0})
data2['family_history_with_overweight']=data2['family_history_with_overweight'].map({'no':1,'yes':0})
data2['CAEC']=data2['CAEC'].map({'Sometimes':1,'Frequently':2,'Always':3,'no':0})
data2['MTRANS']=data2['MTRANS'].map({'Public_Transportation':1,'Walking':2,'Automobile':3,'Motorbike':4,'Bike':0})


data2=data2.values
X=data2[:,0:-1]
y=data2[:,-1]


X_train, X_test, y_train, y_test= train_test_split(X,y,test_size=0.2, stratify=y)
print("A", X_train.shape, X_test.shape, y_train.shape, y_test.shape)


k=int(input("Enter the number of nearest neighbours to be used, i.e. k:"))
model=KNeighborsClassifier(n_neighbors=k, weights='distance')
model.fit(X_train, y_train)
pred=model.predict(X_test)
print(pred)
accuracy=accuracy_score(y_test, pred)
print("Accuracy:",accuracy)
```

Output :

```
0
(1688, 16) (423, 16) (1688,) (423,)

Enter the number of nearest neighbours to be used, i.e. k: 3

['Overweight_Level_I' 'Obesity_Type_II' 'Obesity_Type_I'
 'Overweight_Level_II' 'Obesity_Type_III' 'Overweight_Level_II'
 'Obesity_Type_II' 'Obesity_Type_III' 'Overweight_Level_II'
 'Insufficient_Weight' 'Insufficient_Weight' 'Overweight_Level_II'
 'Overweight_Level_I' 'Overweight_Level_II' 'Overweight_Level_I'
 'Obesity_Type_II' 'Obesity_Type_II' 'Overweight_Level_II'
 'Obesity_Type_I' 'Overweight_Level_II' 'Overweight_Level_I'
 'Insufficient_Weight' 'Obesity_Type_III' 'Obesity_Type_III'
 'Obesity_Type_I' 'Insufficient_Weight' 'Overweight_Level_II'
 'Obesity_Type_II' 'Obesity_Type_I' 'Overweight_Level_II'
 'Obesity_Type_III' 'Insufficient_Weight' 'Overweight_Level_II'
 'Obesity_Type_I' 'Obesity_Type_II' 'Obesity_Type_II'
 'Insufficient_Weight' 'Overweight_Level_II' 'Obesity_Type_I'
 'Obesity_Type_I' 'Obesity_Type_II' 'Insufficient_Weight' 'Normal_Weight'
 'Obesity_Type_III' 'Obesity_Type_III' 'Obesity_Type_II'
 'Obesity_Type_III' 'Normal_Weight' 'Obesity_Type_II'
 'Overweight_Level_II' 'Overweight_Level_I' 'Normal_Weight'
 'Overweight_Level_II' 'Normal_Weight' 'Obesity_Type_III'
 'Obesity_Type_III' 'Obesity_Type_II' 'Obesity_Type_II' 'Obesity_Type_II'
 'Insufficient_Weight' 'Obesity_Type_III' 'Obesity_Type_II'
 'Obesity_Type_I' 'Overweight_Level_II' 'Obesity_Type_I'
 'Insufficient_Weight' 'Obesity_Type_I' 'Normal_Weight'
 'Overweight_Level_I' 'Obesity_Type_III' 'Normal_Weight' 'Obesity_Type_I'
 'Normal_Weight' 'Obesity_Type_III' 'Overweight_Level_II' 'Obesity_Type_I'
 'Insufficient_Weight' 'Overweight_Level_II' 'Obesity_Type_I'
 'Obesity_Type_III' 'Obesity_Type_II' 'Normal_Weight' 'Overweight_Level_I'
 'Overweight_Level_I' 'Obesity_Type_I' 'Obesity_Type_II'
 'Insufficient_Weight' 'Overweight_Level_II' 'Obesity_Type_II'
 'Obesity_Type_II' 'Obesity_Type_II' 'Obesity_Type_II'
 'Overweight_Level_II' 'Overweight_Level_I' 'Insufficient_Weight'
 'Obesity_Type_I' 'Obesity_Type_I' 'Obesity_Type_I' 'Obesity_Type_III'
```

```
'Overweight_Level_I' 'Insufficient_Weight' 'Obesity_Type_II'
'Overweight_Level_II' 'Obesity_Type_I' 'Obesity_Type_I'
'Insufficient_Weight' 'Overweight_Level_I' 'Insufficient_Weight'
'Normal_Weight' 'Obesity_Type_III' 'Normal_Weight' 'Overweight_Level_II'
'Obesity_Type_I' 'Obesity_Type_II' 'Normal_Weight' 'Obesity_Type_I'
'Obesity_Type_I' 'Insufficient_Weight' 'Insufficient_Weight'
'Obesity_Type_II' 'Obesity_Type_I' 'Insufficient_Weight'
'Obesity_Type_III' 'Overweight_Level_I' 'Insufficient_Weight'
'Obesity_Type_II' 'Insufficient_Weight' 'Obesity_Type_I'
'Obesity_Type_III' 'Obesity_Type_III' 'Overweight_Level_II'
'Obesity_Type_II' 'Normal_Weight' 'Overweight_Level_I'
'Insufficient_Weight' 'Obesity_Type_I' 'Obesity_Type_III'
'Obesity_Type_I' 'Obesity_Type_II' 'Normal_Weight' 'Overweight_Level_I'
'Obesity_Type_II' 'Overweight_Level_I' 'Obesity_Type_I'
'Overweight_Level_I' 'Overweight_Level_II' 'Overweight_Level_II'
'Overweight_Level_I' 'Obesity_Type_III' 'Obesity_Type_III'
'Overweight_Level_II' 'Overweight_Level_II' 'Obesity_Type_II'
'Normal_Weight' 'Obesity_Type_I' 'Overweight_Level_I'
'Overweight_Level_II' 'Insufficient_Weight' 'Overweight_Level_II'
'Obesity_Type_III' 'Overweight_Level_II' 'Obesity_Type_III'
'Overweight_Level_I' 'Insufficient_Weight' 'Normal_Weight'
'Overweight_Level_II' 'Obesity_Type_I' 'Obesity_Type_III' 'Normal_Weight'
'Insufficient_Weight' 'Obesity_Type_II' 'Insufficient_Weight'
'Obesity_Type_II' 'Overweight_Level_II' 'Obesity_Type_I' 'Normal_Weight'
'Obesity_Type_III' 'Overweight_Level_II' 'Insufficient_Weight'
'Obesity_Type_I' 'Obesity_Type_I' 'Obesity_Type_II' 'Obesity_Type_III'
'Normal_Weight' 'Insufficient_Weight' 'Obesity_Type_II' 'Obesity_Type_I'
'Overweight_Level_II' 'Insufficient_Weight' 'Obesity_Type_II'
'Normal_Weight' 'Obesity_Type_I' 'Obesity_Type_II' 'Overweight_Level_I'
'Overweight_Level_I' 'Obesity_Type_I' 'Obesity_Type_II' 'Obesity_Type_I'
'Obesity_Type_III' 'Overweight_Level_I' 'Overweight_Level_II'
'Normal_Weight' 'Overweight_Level_I' 'Overweight_Level_I'
'Obesity_Type_II' 'Overweight_Level_II' 'Insufficient_Weight'
'Overweight_Level_I' 'Overweight_Level_I' 'Normal_Weight'
'Insufficient_Weight' 'Normal_Weight' 'Obesity_Type_I' 'Obesity_Type_II'
```

```
'Overweight_Level_I' 'Obesity_Type_I' 'Overweight_Level_II'
'Obesity_Type_III' 'Obesity_Type_I' 'Obesity_Type_II'
'Overweight_Level_I' 'Overweight_Level_I' 'Obesity_Type_I'
'Insufficient_Weight' 'Overweight_Level_II' 'Overweight_Level_I'
'Obesity_Type_II' 'Overweight_Level_II' 'Obesity_Type_I'
'Obesity_Type_III' 'Normal_Weight' 'Obesity_Type_II' 'Obesity_Type_III'
'Overweight_Level_I' 'Obesity_Type_III' 'Overweight_Level_II'
'Insufficient_Weight' 'Obesity_Type_III' 'Normal_Weight'
'Obesity_Type_III' 'Obesity_Type_I' 'Overweight_Level_II'
'Overweight_Level_II' 'Obesity_Type_III' 'Overweight_Level_I'
'Obesity_Type_I' 'Obesity_Type_III' 'Obesity_Type_III'
'Overweight_Level_II' 'Overweight_Level_II' 'Obesity_Type_III'
'Normal_Weight' 'Obesity_Type_I' 'Overweight_Level_I'
'Overweight_Level_I' 'Overweight_Level_I' 'Obesity_Type_III'
'Overweight_Level_I' 'Normal_Weight' 'Overweight_Level_I'
'Overweight_Level_II' 'Overweight_Level_II' 'Insufficient_Weight'
'Obesity_Type_III' 'Obesity_Type_III' 'Insufficient_Weight'
'Obesity_Type_III' 'Obesity_Type_III' 'Overweight_Level_I'
'Obesity_Type_II' 'Overweight_Level_I' 'Overweight_Level_I'
'Overweight_Level_II' 'Overweight_Level_II' 'Overweight_Level_I'
'Overweight_Level_II' 'Normal_Weight' 'Insufficient_Weight'
'Insufficient_Weight' 'Insufficient_Weight' 'Overweight_Level_I'
'Obesity_Type_III' 'Insufficient_Weight' 'Overweight_Level_II'
'Overweight_Level_II' 'Obesity_Type_III' 'Insufficient_Weight'
'Obesity_Type_III' 'Obesity_Type_III' 'Overweight_Level_II'
'Insufficient_Weight' 'Obesity_Type_I' 'Obesity_Type_III'
'Overweight_Level_II' 'Overweight_Level_II' 'Obesity_Type_III'
'Obesity_Type_II' 'Obesity_Type_III' 'Obesity_Type_I' 'Obesity_Type_II'
'Obesity_Type_I' 'Insufficient_Weight' 'Obesity_Type_II' 'Obesity_Type_I'
'Obesity_Type_I' 'Insufficient_Weight' 'Overweight_Level_I'
'Obesity_Type_I' 'Obesity_Type_III' 'Overweight_Level_I'
'Overweight_Level_I' 'Normal_Weight' 'Insufficient_Weight'
'Normal_Weight' 'Obesity_Type_I' 'Insufficient_Weight' 'Normal_Weight'
'Overweight_Level_II' 'Overweight_Level_I' 'Obesity_Type_III'
'Obesity_Type_I' 'Normal_Weight' 'Obesity_Type_II' 'Obesity_Type_I'
```

```
 'Obesity_Type_I' 'Overweight_Level_II' 'Overweight_Level_II'
 'Insufficient_Weight' 'Insufficient_Weight' 'Overweight_Level_I'
 'Obesity_Type_I' 'Obesity_Type_I' 'Overweight_Level_I'
 'Overweight_Level_I' 'Obesity_Type_I' 'Obesity_Type_III' 'Obesity_Type_I'
 'Overweight_Level_II' 'Obesity_Type_II' 'Normal_Weight'
 'Overweight_Level_I' 'Obesity_Type_II' 'Obesity_Type_III'
 'Overweight_Level_I' 'Insufficient_Weight' 'Overweight_Level_I'
 'Obesity_Type_I' 'Obesity_Type_II' 'Obesity_Type_III'
 'Overweight_Level_I' 'Overweight_Level_II' 'Obesity_Type_III'
 'Overweight_Level_II' 'Obesity_Type_III' 'Normal_Weight'
 'Insufficient_Weight' 'Obesity_Type_II' 'Insufficient_Weight'
 'Overweight_Level_II' 'Overweight_Level_I' 'Obesity_Type_I'
 'Insufficient_Weight' 'Insufficient_Weight' 'Obesity_Type_I'
 'Overweight_Level_I' 'Overweight_Level_I' 'Obesity_Type_I'
 'Insufficient_Weight' 'Insufficient_Weight' 'Overweight_Level_I'
 'Insufficient_Weight' 'Obesity_Type_III' 'Obesity_Type_III'
 'Obesity_Type_III' 'Obesity_Type_II' 'Obesity_Type_III' 'Obesity_Type_II'
 'Overweight_Level_II' 'Overweight_Level_I' 'Insufficient_Weight'
 'Obesity_Type_I' 'Obesity_Type_II' 'Overweight_Level_II' 'Obesity_Type_I'
 'Obesity_Type_II' 'Obesity_Type_I' 'Insufficient_Weight'
 'Insufficient_Weight' 'Overweight_Level_II' 'Obesity_Type_I'
 'Overweight_Level_I' 'Overweight_Level_II' 'Obesity_Type_III'
 'Overweight_Level_I' 'Obesity_Type_III' 'Obesity_Type_II'
 'Obesity_Type_I' 'Obesity_Type_III' 'Insufficient_Weight'
 'Overweight_Level_II' 'Insufficient_Weight' 'Insufficient_Weight'
 'Overweight_Level_II' 'Overweight_Level_II' 'Obesity_Type_II'
 'Insufficient_Weight' 'Obesity_Type_II' 'Insufficient_Weight'
 'Overweight_Level_I' 'Insufficient_Weight' 'Obesity_Type_I'
 'Insufficient_Weight' 'Obesity_Type_I' 'Overweight_Level_II'
 'Obesity_Type_II' 'Obesity_Type_I' 'Overweight_Level_II'
 'Obesity_Type_II' 'Obesity_Type_II']
Accuracy: 0.8983451536643026
```

In this code we use the library sklearn.

Change the value we use map.

In this dataset we use knn classification .

THANK YOU!