```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
df=pd.read_csv('/content/ObesityDataSet (1).csv')
df
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCV
0	Female	21.000000	1.620000	64.000000	yes	no	2.
1	Female	21.000000	1.520000	56.000000	yes	no	3.
2	Male	23.000000	1.800000	77.000000	yes	no	2.
3	Male	27.000000	1.800000	87.000000	no	no	3.
4	Male	22.000000	1.780000	89.800000	no	no	2.
...	...	...	...	...	...	...	..
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.

2111 rows × 17 columns

Next steps: Generate code with df View recommended plots

```
df.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	C
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Someti
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Someti
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Someti
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Someti
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Someti

Next steps: Generate code with df View recommended plots

```
df.tail()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCV
2106	Female	20.976842	1.710730	131.408528	yes	yes	3.
2107	Female	21.982942	1.748584	133.742943	yes	yes	3.
2108	Female	22.524036	1.752206	133.689352	yes	yes	3.
2109	Female	24.361936	1.739450	133.346641	yes	yes	3.
2110	Female	23.664709	1.738836	133.472641	yes	yes	3.

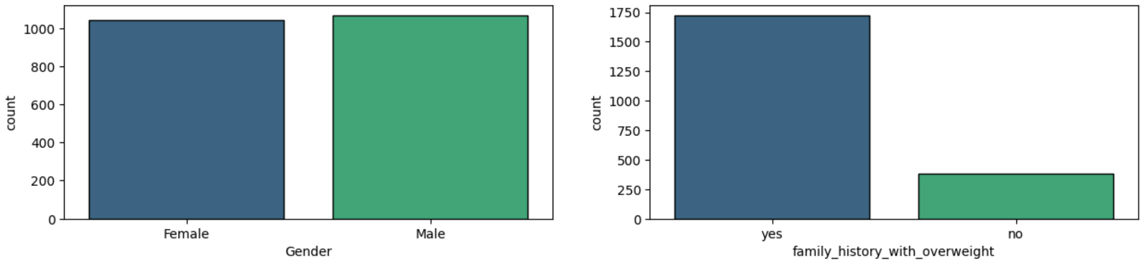
```
df.isna().sum()
```

```
Gender                            0
Age                               0
Height                            0
Weight                            0
family_history_with_overweight    0
FAVC                              0
FCVC                              0
NCP                               0
CAEC                              0
SMOKE                             0
CH2O                              0
SCC                               0
FAF                               0
TUE                               0
CALC                              0
MTRANS                            0
NObeyesdad                        0
dtype: int64
```

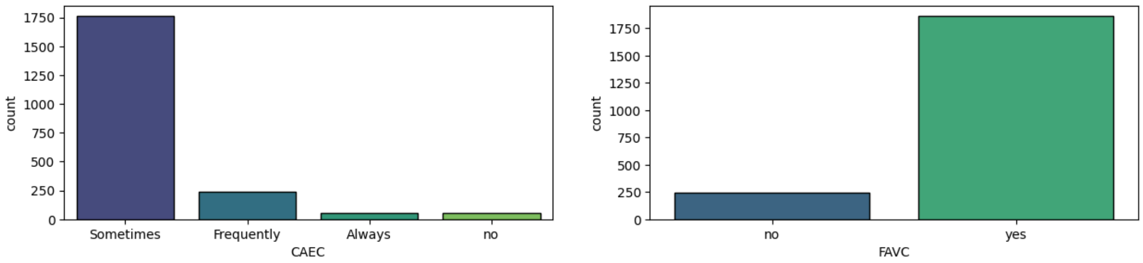
```
df.dtypes
```

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

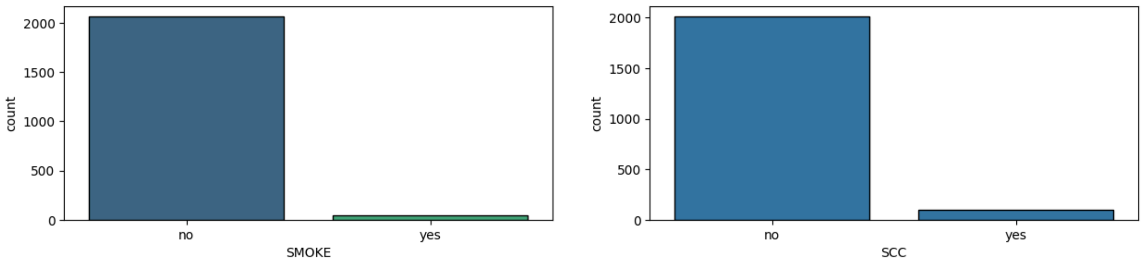
```
import seaborn as sns
plt.figure(figsize=(15,3))
plt.subplot(1,2,1)
sns.countplot(x='Gender',data=df,palette='viridis',edgecolor='k')
plt.subplot(1,2,2)
sns.countplot(x='family_history_with_overweight',data=df,palette='viridis',edgecolor='k')
plt.show()
```



```
plt.figure(figsize=(15,3))
plt.subplot(1,2,1)
sns.countplot(x='CAEC',data=df,palette='viridis',edgecolor='k')
plt.subplot(1,2,2)
sns.countplot(x='FAVC',data=df,palette='viridis',edgecolor='k')
plt.show()
```

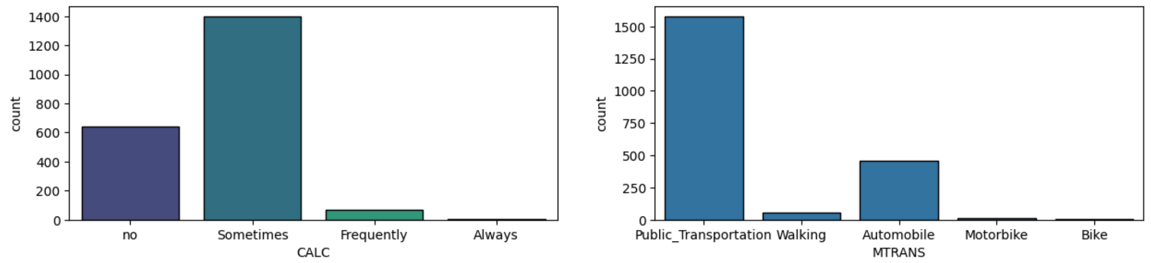


```
plt.figure(figsize=(15,3))
plt.subplot(1,2,1)
sns.countplot(x='SMOKE',data=df,palette='viridis',edgecolor='k')
plt.subplot(1,2,2)
sns.countplot(x='SCC',data=df,edgecolor='k')
plt.show()
```



```
plt.figure(figsize=(15,3))
plt.subplot(1,2,1)
sns.countplot(x='CALC',data=df,palette='viridis',edgecolor='k')
plt.subplot(1,2,2)
sns.countplot(x='MTRANS',data=df,edgecolor='k')
#plt.show()
```

```
<Axes: xlabel='MTRANS', ylabel='count'>
```



```
from sklearn.preprocessing import LabelEncoder
encode=LabelEncoder()
df['Gender']=encode.fit_transform(df['Gender'])
df['family_history_with_overweight']=encode.fit_transform(df['family_history_with_overweight'])
df['FAVC']=encode.fit_transform(df['FAVC'])
df['CAEC']=encode.fit_transform(df['CAEC'])
df['SMOKE']=encode.fit_transform(df['SMOKE'])
df['SCC']=encode.fit_transform(df['SCC'])
df['CALC']=encode.fit_transform(df['CALC'])
df['MTRANS']=encode.fit_transform(df['MTRANS'])
df.dtypes
```

```
Gender                              int64
Age                               float64
Height                            float64
Weight                            float64
family_history_with_overweight      int64
FAVC                                int64
FCVC                              float64
NCP                               float64
CAEC                                int64
SMOKE                               int64
CH2O                              float64
SCC                                 int64
FAF                               float64
TUE                               float64
CALC                                int64
MTRANS                              int64
NObeyesdad                         object
dtype: object
```

```
x=df.iloc[:,:-1].values
x
```

```
y=df.iloc[:,-1].values
y
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=0)
x_train
```

```
y_train
```

```
len(y_train)
```

```
print(y_train[0:1000])
print(y_train[1000:1477])
```

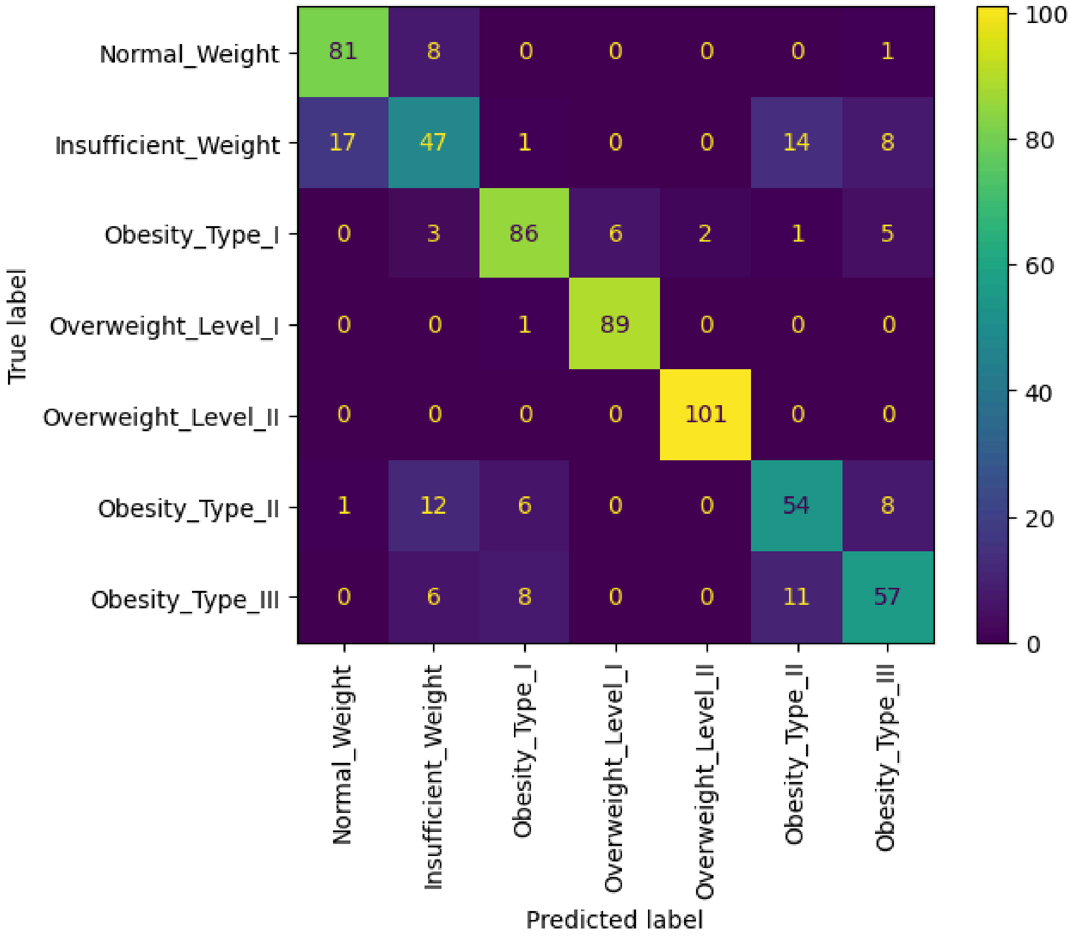
```
y_test
```

```
from sklearn.preprocessing import MinMaxScaler
norm=MinMaxScaler()
norm.fit(x_train)
x_train=norm.transform(x_train)
x_test=norm.transform(x_test)
x_train
```

```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay,classification_report
model=SVC()
model1=RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=42)
model2=DecisionTreeClassifier(criterion='entropy')
lst=[model,model1,model2]
```

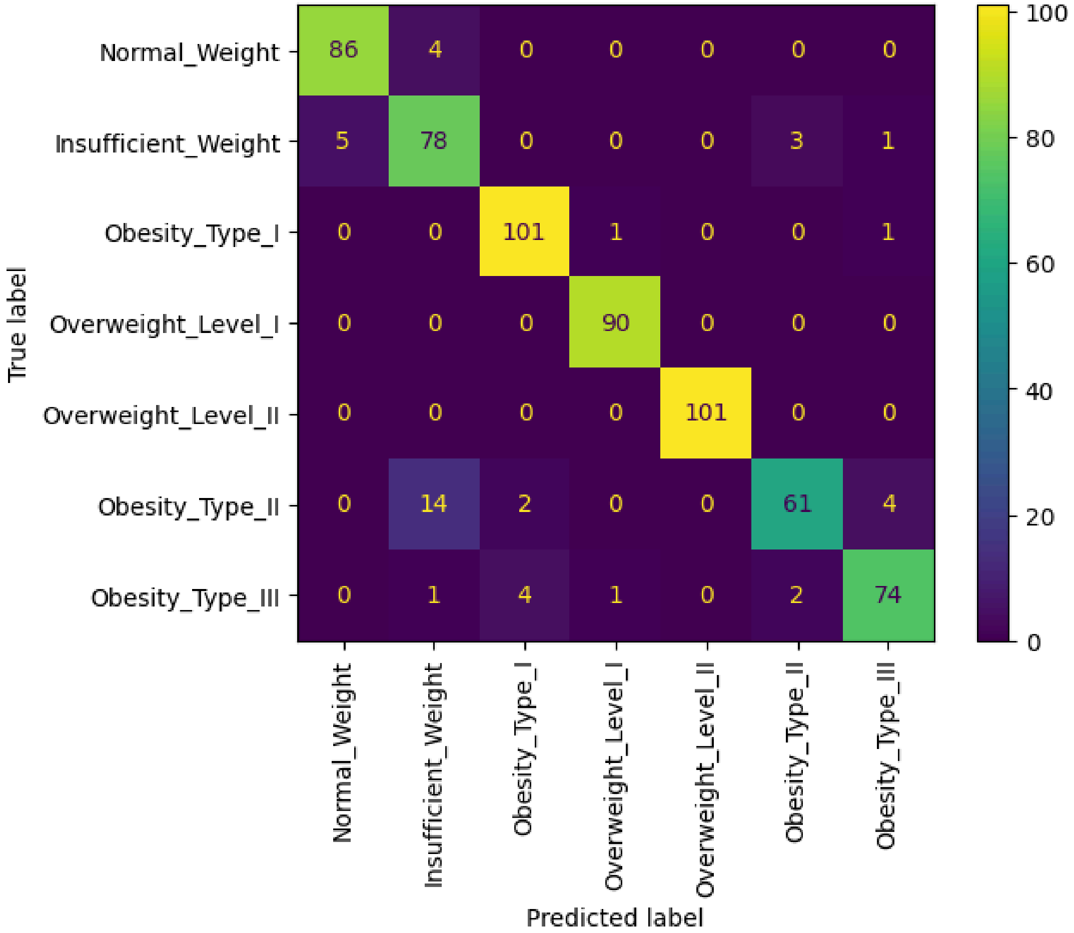
```
for i in lst:
  i.fit(x_train,y_train)
  y_pred=i.predict(x_test)
  print("Model is", i )
  #print(y_pred)
  cm=confusion_matrix(y_test,y_pred)
  print(cm)
  print("Score is", accuracy_score(y_test,y_pred))
  print(classification_report(y_test,y_pred))
  label=['Normal_Weight','Insufficient_Weight','Obesity_Type_I','Overweight_Level_I','Overweight_Level_II','(
  cmd=ConfusionMatrixDisplay(cm,display_labels=label)
  cmd.plot(xticks_rotation = 'vertical')
  plt.show()
  print("\n")
```

```
Model is SVC()
[[ 81   8   0   0   0   0   1]
 [ 17  47   1   0   0  14   8]
 [  0   3  86   6   2   1   5]
 [  0   0   1  89   0   0   0]
 [  0   0   0   0 101   0   0]
 [  1  12   6   0   0  54   8]
 [  0   6   8   0   0  11  57]]
Score is 0.8123028391167192
                     precision    recall  f1-score   support

 Insufficient_Weight       0.82      0.90      0.86        90
       Normal_Weight       0.62      0.54      0.58        87
      Obesity_Type_I       0.84      0.83      0.84       103
     Obesity_Type_II       0.94      0.99      0.96        90
    Obesity_Type_III       0.98      1.00      0.99       101
  Overweight_Level_I       0.68      0.67      0.67        81
 Overweight_Level_II       0.72      0.70      0.71        82

            accuracy                           0.81       634
           macro avg       0.80      0.80      0.80       634
        weighted avg       0.81      0.81      0.81       634
```

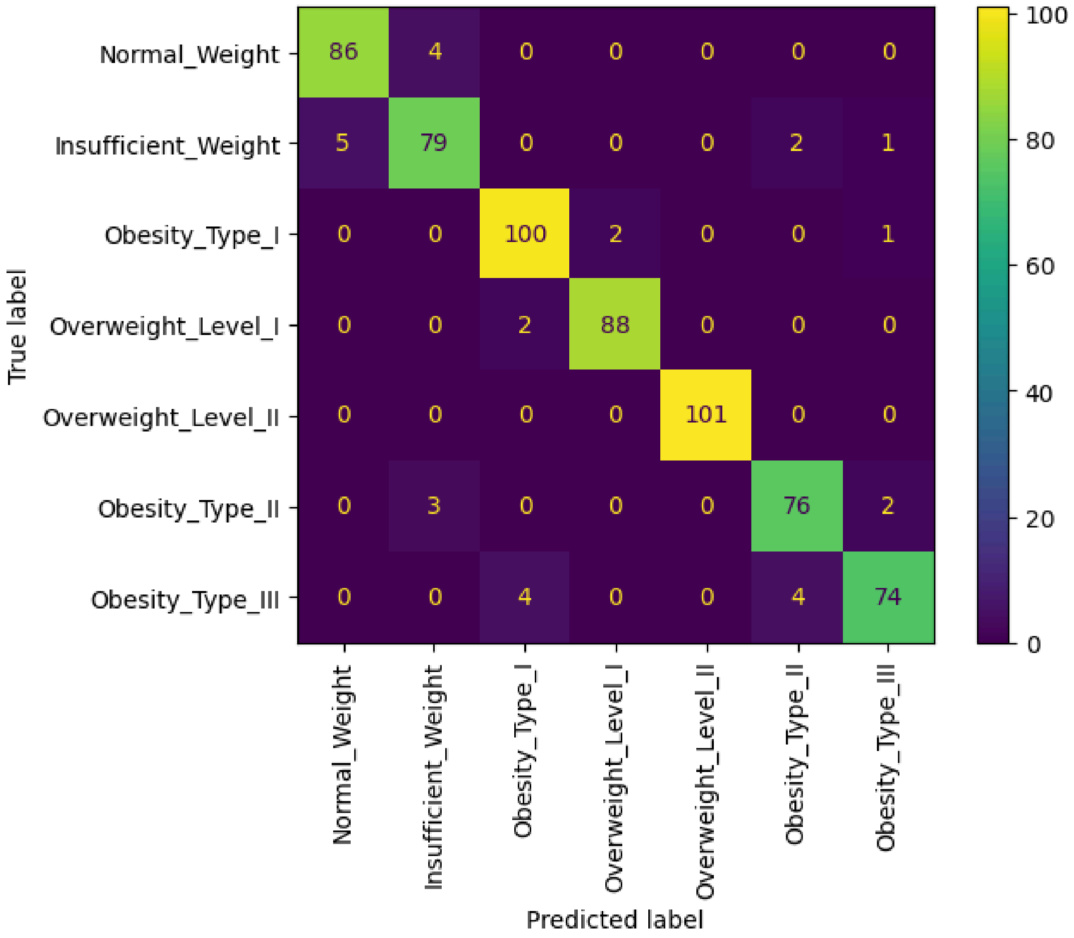


```
Model is RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=42)
[[ 86   4   0   0   0   0   0]
 [  5  78   0   0   0   3   1]
 [  0   0 101   1   0   0   1]
 [  0   0   0  90   0   0   0]
 [  0   0   0   0 101   0   0]
 [  0  14   2   0   0  61   4]
 [  0   1   4   1   0   2  74]]
```

```
Score is 0.9321766561514195
                     precision    recall  f1-score   support

Insufficient_Weight       0.95      0.96      0.95        90
      Normal_Weight       0.80      0.90      0.85        87
     Obesity_Type_I       0.94      0.98      0.96       103
    Obesity_Type_II       0.98      1.00      0.99        90
   Obesity_Type_III       1.00      1.00      1.00       101
 Overweight_Level_I       0.92      0.75      0.83        81
Overweight_Level_II       0.93      0.90      0.91        82

           accuracy                           0.93       634
          macro avg       0.93      0.93      0.93       634
       weighted avg       0.93      0.93      0.93       634
```



```
Model is DecisionTreeClassifier(criterion='entropy')
[[ 86   4   0   0   0   0   0]
 [  5  79   0   0   0   2   1]
 [  0   0 100   2   0   0   1]
 [  0   0   2  88   0   0   0]
 [  0   0   0   0 101   0   0]
 [  0   3   0   0   0  76   2]
 [  0   0   4   0   0   4  74]]
Score is 0.9526813880126183
                     precision    recall  f1-score   support

Insufficient_Weight       0.95      0.96      0.95        90
      Normal_Weight       0.92      0.91      0.91        87
     Obesity_Type_I       0.94      0.97      0.96       103
    Obesity_Type_II       0.98      0.98      0.98        90
   Obesity_Type_III       1.00      1.00      1.00       101
 Overweight_Level_I       0.93      0.94      0.93        81
 Overweight_Level_I       0.93      0.94      0.93        81
Overweight_Level_II       0.95      0.90      0.92        82

           accuracy                           0.95       634
          macro avg       0.95      0.95      0.95       634
       weighted avg       0.95      0.95      0.95       634
```


Normal_Weight
Insufficient_Weight
Obesity_Type_I
Overweight_Level_I
Overweight_Level_II
Obesity_Type_II
Obesity_Type_III
True label
Normal_Weight
Insufficient_Weight
Obesity_Type_I
Overweight_Level_I
Overweight_Level_II
Obesity_Type_II
Obesity_Type_III
Predicted label