Machine learning

Q-2. Imagine you have a dataset where you have different features like Age , Gender , Height , Weight , BMI , and Blood Pressure and you have to classify the people into different classes like Normal , Overweight , Obesity , Underweight , and Extreme Obesity by using any 4 different classification algorithms. Now you have to build a model which can classify people into different classes. Dataset This is the Dataset You can use this dataset for this question.

In [1]:

```
## Import the necessary libraries:-
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
# Load the dataset
data = pd.read_csv('Downloads/archive (2)/ObesityDataSet_raw_and_data_sinthetic.csv
```

In [4]:

```
## Checking top 5 rows
data.head()
```

Out[4]:

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CA
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

In [5]:

```
## Checking Rows & Columns Availabale in Dataset
data.shape
```

Out[5]:

```
(2111, 17)
```

In [6]:

```
## Checking Details Information related with Dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Gender                          2111 non-null   object 
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object 
 5   FAVC                            2111 non-null   object 
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object 
 9   SMOKE                           2111 non-null   object 
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object 
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            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
```

In [7]:

```
## Checking All Columns name present in dataset
data.columns
```

Out[7]:

```
Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweig
ht',
       'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TU
E',
       'CALC', 'MTRANS', 'NObeyesdad'],
      dtype='object')
```

In [8]:

```
## Checking Statistical Analysis of Dataset
data.describe()
```

Out[8]:

	Age	Height	Weight	FCVC	NCP	CH2O	
count	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.000000	2111.00
mean	24.312600	1.701677	86.586058	2.419043	2.685628	2.008011	1.01
std	6.345968	0.093305	26.191172	0.533927	0.778039	0.612953	0.85
min	14.000000	1.450000	39.000000	1.000000	1.000000	1.000000	0.00
25%	19.947192	1.630000	65.473343	2.000000	2.658738	1.584812	0.12
50%	22.777890	1.700499	83.000000	2.385502	3.000000	2.000000	1.00
75%	26.000000	1.768464	107.430682	3.000000	3.000000	2.477420	1.66
max	61.000000	1.980000	173.000000	3.000000	4.000000	3.000000	3.00

In [9]:

```
## Checking Information Related with Dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Gender                          2111 non-null   object 
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   object 
 5   FAVC                            2111 non-null   object 
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   object 
 9   SMOKE                           2111 non-null   object 
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   object 
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            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
```

In [10]:

```
## Checking All Columns Available in dataset
data.columns
```

Out[10]:

```
Index(['Gender', 'Age', 'Height', 'Weight', 'family_history_with_overweig
ht',
       'FAVC', 'FCVC', 'NCP', 'CAEC', 'SMOKE', 'CH2O', 'SCC', 'FAF', 'TU
E',
       'CALC', 'MTRANS', 'NObeyesdad'],
      dtype='object')
```

In [11]:

```
# Preprocess the dataset
encoder = LabelEncoder()
data['Gender'] = encoder.fit_transform(data['Gender'])
data['family_history_with_overweight'] = encoder.fit_transform(data['family_history_
data['FAVC'] = encoder.fit_transform(data['FAVC'])
data['CAEC'] = encoder.fit_transform(data['CAEC'])
data['SMOKE'] = encoder.fit_transform(data['SMOKE'])
data['SCC'] = encoder.fit_transform(data['SCC'])
data['CALC'] = encoder.fit_transform(data['CALC'])
data['MTRANS'] = encoder.fit_transform(data['MTRANS'])
data['NObeyesdad'] = encoder.fit_transform(data['NObeyesdad'])
```

In [12]:

```
## Checking Details Information related with Dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2111 entries, 0 to 2110
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Gender                          2111 non-null   int32
 1   Age                             2111 non-null   float64
 2   Height                          2111 non-null   float64
 3   Weight                          2111 non-null   float64
 4   family_history_with_overweight  2111 non-null   int32
 5   FAVC                            2111 non-null   int32
 6   FCVC                            2111 non-null   float64
 7   NCP                             2111 non-null   float64
 8   CAEC                            2111 non-null   int32
 9   SMOKE                           2111 non-null   int32
 10  CH2O                            2111 non-null   float64
 11  SCC                             2111 non-null   int32
 12  FAF                             2111 non-null   float64
 13  TUE                             2111 non-null   float64
 14  CALC                            2111 non-null   int32
 15  MTRANS                          2111 non-null   int32
 16  NObeyesdad                      2111 non-null   int32
dtypes: float64(8), int32(9)
memory usage: 206.3 KB
```

In [13]:

```
# Split the dataset into features (X) and target (y)
X = data.drop('NObeyesdad', axis=1)
y = data['NObeyesdad']
```

In [14]:

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

```

In [15]:

```
# Decision Tree Classifier
dt_clf = DecisionTreeClassifier()
dt_clf.fit(X_train, y_train)
dt_predictions = dt_clf.predict(X_test)
```

In [16]:

```
# Logistic Regression Classifier
lr_clf = LogisticRegression()
lr_clf.fit(X_train, y_train)
lr_predictions = lr_clf.predict(X_test)
```

In [17]:

```
# Random Forest Classifier
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, y_train)
rf_predictions = rf_clf.predict(X_test)
```

In [18]:

```
# Support Vector Machine (SVM) Classifier
svm_clf = SVC()
svm_clf.fit(X_train, y_train)
svm_predictions = svm_clf.predict(X_test)
```

In [19]:

```
# Print classification reports for each classifier
print("Decision Tree Classifier:")
print(classification_report(y_test, dt_predictions))
```

```
Decision Tree Classifier:
              precision    recall  f1-score   support

           0       0.92      0.96      0.94        56
           1       0.87      0.87      0.87        62
           2       0.97      0.92      0.95        78
           3       0.93      0.95      0.94        58
           4       1.00      1.00      1.00        63
           5       0.91      0.89      0.90        56
           6       0.94      0.96      0.95        50

    accuracy                           0.94       423
   macro avg       0.93      0.94      0.94       423
weighted avg       0.94      0.94      0.94       423
```

In [20]:

```
print("Logistic Regression Classifier:")
print(classification_report(y_test, lr_predictions))
```

```
Logistic Regression Classifier:
              precision    recall  f1-score   support

           0       0.74      0.93      0.83        56
           1       0.53      0.42      0.47        62
           2       0.58      0.60      0.59        78
           3       0.82      0.84      0.83        58
           4       0.90      1.00      0.95        63
           5       0.54      0.38      0.44        56
           6       0.35      0.38      0.37        50

    accuracy                           0.65       423
   macro avg       0.64      0.65      0.64       423
weighted avg       0.64      0.65      0.64       423
```

In [21]:

```
print("Random Forest Classifier:")
print(classification_report(y_test, rf_predictions))
```

```
Random Forest Classifier:
              precision    recall  f1-score   support

           0       1.00      0.96      0.98        56
           1       0.88      0.94      0.91        62
           2       0.99      0.96      0.97        78
           3       0.97      0.98      0.97        58
           4       1.00      1.00      1.00        63
           5       0.89      0.88      0.88        56
           6       0.96      0.96      0.96        50

    accuracy                           0.96       423
   macro avg       0.95      0.95      0.95       423
weighted avg       0.96      0.96      0.96       423
```

In [22]:

```
print("SVM Classifier:")
print(classification_report(y_test, svm_predictions))
```

```
SVM Classifier:
              precision    recall  f1-score   support

           0       0.71      0.88      0.78        56
           1       0.48      0.34      0.40        62
           2       0.65      0.33      0.44        78
           3       0.77      0.41      0.54        58
           4       0.56      1.00      0.72        63
           5       0.47      0.48      0.47        56
           6       0.43      0.58      0.49        50

    accuracy                           0.57       423
   macro avg       0.58      0.57      0.55       423
weighted avg       0.59      0.57      0.54       423
```

In [ ]:

```

```