**Course Code : CBS3007**

**Course: Data Mining and Analytics**

**Class Number: VL2024250103849 / 50**

**Slot: L1+L2 / L49+L50**

**Mudit Sultania**

**21BBS0232**

**Q-1**

**Aim:** To collect the data set consists of 50 observations about patient enrolment in diet maintenance based on gender, weight, BMI etc (minimum 7 features). Implement a model that will recommend a strict diet is necessary or not for a patient using the naïve Bayes classification algorithm.

**Sample input :**

patient_diet_recommendation.csv > data

```
Gender,Weight,Height,BMI,Age,BloodPressure,Cholesterol,StrictDietNeeded
M,58.7,193.7,15.6,49,117.8,Normal,0
F,71.0,150.8,31.2,21,122.7,Borderline,1
M,90.6,156.0,37.2,28,134.5,High,1
M,68.9,159.4,27.1,34,104.0,Normal,1
M,66.2,145.8,31.1,55,121.6,High,1
F,59.8,170.4,20.6,41,121.6,Borderline,0
M,59.0,169.0,20.7,22,133.8,High,1
M,74.5,160.2,29.0,51,116.6,High,1
M,67.9,154.7,28.4,23,129.8,Borderline,1
F,58.6,145.0,27.9,39,136.5,High,1
M,54.8,152.1,23.7,28,92.3,High,1
M,63.4,167.4,22.6,33,137.5,Borderline,0
M,59.7,157.9,23.9,50,105.2,High,1
M,64.9,155.7,26.8,26,103.6,Borderline,1
F,62.7,165.0,23.0,23,76.8,High,1
M,68.9,171.6,23.4,33,108.0,Borderline,0
F,[illegible].8,46,119.9,Borderline,0
F,[illegible].5,20,114.3,High,1
F,92.8,173.7,30.8,37,113.1,High,1
M,76.9,161.8,29.4,53,103.3,High,1
F,67.2,156.9,27.3,36,133.0,Normal,1
M,73.8,166.7,26.6,43,134.4,Normal,1
F,54.9,157.4,22.2,20,126.3,Normal,0
F,49.2,156.3,20.1,36,91.7,Borderline,0
F,72.7,172.7,24.4,37,154.1,High,1
F,59.6,157.1,24.1,49,122.2,Borderline,0
```

Col 2: Weight

**CODE AND OUTPUT:**

# Importing libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

[ ]

# Load the dataset

```
df = pd.read_csv('patient_diet_recommendation.csv')
```

[3] ✓ 0.0s

Preprocess the data

```
df['Gender'] = df['Gender'].map({'M': 0, 'F': 1})  # Male: 0, Female: 1
df['Cholesterol'] = df['Cholesterol'].map({'Normal': 0, 'Borderline': 1, 'High': 2})
X = df[['Gender', 'Weight', 'Height', 'BMI', 'Age', 'BloodPressure', 'Cholesterol']]
y = df['StrictDietNeeded']
```

[4] ✓ 0.0s

Training

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[5] ✓ 0.0s

Naive bayes

```
nb = GaussianNB()

# Train the model
nb.fit(X_train, y_train)

# Make predictions on the test set
y_pred = nb.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

[6] ✓ 0.0s


**Conclusion :** We were able to achieve the required accuracy.

Link to the dataset: https://github.com/muditsultania/DataMining-Lab-2/blob/main/patient_diet_recommendation.csv

**Q-2**

**Aim:** To Implement K-means method of clustering. Use the patient details data set to classify into 3 clusters such as a person is normal, healthy and weak. A person/patient must be clustered as any one of normal/healthy or weak based on his/her input values.

**Sample input:**

```
Gender,Weight,Height,BMI,Age,BloodPressure,ClusterLabel
M,40.4,146.5,33.4,38,123.5,Normal
F,64.2,153.6,31.4,36,123.5,Healthy
F,59.1,153.4,14.3,36,112.8,Weak
F,62.0,152.2,27.7,53,154.5,Healthy
F,60.9,151.3,24.4,46,101.0,Normal
M,74.8,153.3,23.2,35,100.1,Healthy
F,66.0,163.4,20.1,19,144.8,Healthy
M,67.7,164.9,26.6,18,109.4,Healthy
F,51.6,160.0,18.2,22,105.8,Healthy
F,31.4,155.7,30.6,37,154.5,Normal
F,76.2,172.8,20.8,28,134.8,Healthy
M,86.4,159.1,24.9,59,111.1,Healthy
F,61.6,169.4,31.1,19,123.3,Weak
F,67.5,154.3,26.7,20,149.2,Healthy
F,68.2,164.1,24.1,40,130.4,Weak
F,81.8,173.4,24.0,29,139.3,Normal
M,73.9,133.9,21.9,37,132.1,Normal
F,70.0,153.4,25.0,22,134.0,Normal
M,65.6,165.9,31.0,54,120.6,Normal
M,65.6,172.1,28.2,55,120.2,Weak
M,50.6,162.0,27.6,47,124.0,Healthy
M,61.2,160.5,34.3,26,131.0,Normal
M,73.9,176.1,31.8,51,115.8,Healthy
M,62.7,172.7,26.5,52,121.8,Weak
F,65.4,151.3,23.2,39,115.3,Weak
M,53.2,161.2,24.8,41,121.7,Weak
M,69.5,163.6,30.1,34,101.2,Normal
M,50.4,177.2,35.2,27,159.1,Healthy
F,67.9,168.7,36.0,47,109.0,Healthy
F,63.1,159.0,24.0,40,114.6,Healthy
M,47.6,162.2,28.2,49,107.7,Weak
F,75.7,141.2,20.3,34,129.2,Normal
M,63.4,169.7,23.2,34,117.1,Weak
```

**CODE AND OUTPUT:**

# Importing necessary libraries

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder, StandardScaler
import matplotlib.pyplot as plt
```

[1] ✓ 3.0s

# Load the dataset

```
df = pd.read_csv('patient_clustering.csv')
```

[2] ✓ 0.0s

## Pre-processing

```
df['Gender'] = df['Gender'].map({'M': 0, 'F': 1})  # Male: 0, Female: 1

X = df[['Gender', 'Weight', 'Height', 'BMI', 'Age', 'BloodPressure']]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
kmeans = KMeans(n_clusters=3, random_state=42)

# Fit the model to the data
kmeans.fit(X_scaled)

# Get the cluster labels for each data point
df['ClusterLabel'] = kmeans.labels_

# Mapping the numeric cluster labels to meaningful names
# Assuming we map cluster 0 -> Normal, 1 -> Healthy, 2 -> Weak
cluster_mapping = {0: 'Normal', 1: 'Healthy', 2: 'Weak'}
df['ClusterLabel'] = df['ClusterLabel'].map(cluster_mapping)
```

✓ 0.1s Python

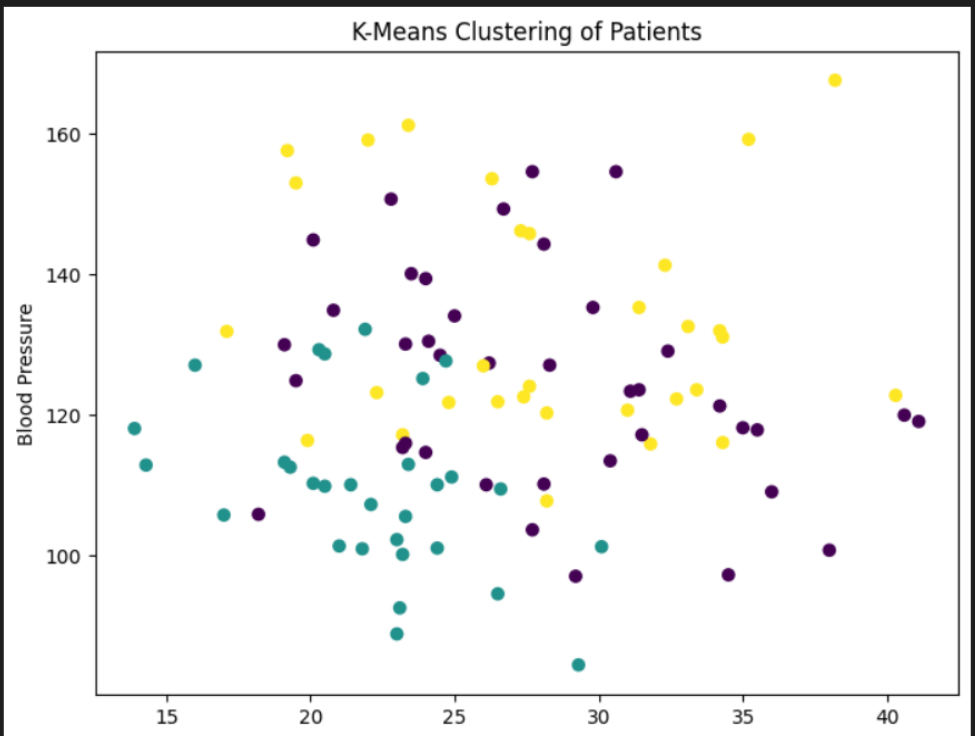
```
c:\Users\mudit\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

```
plt.figure(figsize=(8, 6))
plt.scatter(df['BMI'], df['BloodPressure'], c=kmeans.labels_, cmap='viridis')
plt.title('K-Means Clustering of Patients')
plt.xlabel('BMI')
plt.ylabel('Blood Pressure')
plt.show()
```

✓ 0.2s Python

```
plt.figure(figsize=(8, 6))
plt.scatter(df['BMI'], df['BloodPressure'], c=kmeans.labels_, cmap='viridis')
plt.title('K-Means Clustering of Patients')
plt.xlabel('BMI')
plt.ylabel('Blood Pressure')
plt.show()
```

✓ 0.2s




# Result

```
print(df[['Gender', 'Weight', 'Height', 'BMI', 'Age', 'BloodPressure', 'ClusterLabel']].head())
```

[6] ✓ 0.0s

| | Gender | Weight | Height | BMI | Age | BloodPressure | ClusterLabel |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 40.4 | 146.5 | 33.4 | 38 | 123.5 | Weak |
| 1 | 1 | 64.2 | 153.6 | 31.4 | 36 | 123.5 | Normal |
| 2 | 1 | 59.1 | 153.4 | 14.3 | 36 | 112.8 | Healthy |
| 3 | 1 | 62.0 | 152.2 | 27.7 | 53 | 154.5 | Normal |
| 4 | 1 | 60.9 | 151.3 | 24.4 | 46 | 101.0 | Healthy |

**Conclusion :** We were able to get the desired result.

Link to the dataset: https://github.com/muditsultania/DataMining-Lab-2/blob/main/patient_clustering.csv

**Q-3**

**Aim:** The ID3 algorithm builds decision trees using a top-down greedy search approach through the space of possible branches with no backtracking. Consider a dataset of 50 rows “Road transport records” with the attributes “Road ID”, “Length”, Numberof_Bends”, “Trafficvolume” and “AccidentRisk”. The aim is to Implement the same to the dataset to recommend the decision tree to classify the data.

**Sample Input :**

road_transport_records.csv > data

```
RoadID,Length,NumberOfBends,TrafficVolume,AccidentRisk
R001,37.6,7,4099,Moderate
R002,2.7,18,3094,Moderate
R003,44.9,4,2606,High
R004,43.1,2,4765,High
R005,23.4,14,3709,Low
R006,20.2,13,3653,Low
R007,14.4,12,1553,Moderate
R008,24.4,7,4933,High
R009,3.2,2,3456,Moderate
R010,42.8,13,615,High
R011,2.7,14,138,Low
R012,17.1,17,3442,Moderate
R013,46.6,9,986,Moderate
R014,37.0,17,2944,High
R015,14.8,8,2304,Low
R016,13.4,2,1198,Moderate
R017,1.2,13,1923,Low
R018,35.1,3,3301,Moderate
R019,3.4,16,3255,Low
R020,27.1,6,3193,High
R021,48.1,19,3630,Moderate
R022,6.4,7,3878,Moderate
R023,44.8,10,2996,Low
R024,49.6,9,3553,Low
R025,4.0,2,3611,Moderate
R026,44.3,15,1088,High
R027,26.3,5,4216,Low
R028,45.4,4,4126,Low
R029,29.0,4,4279,Low
R030,33.4,18,1782,Moderate
R031,27.6,4,4765,Moderate
R032,43.4,17,4063,Moderate
R033,37.0,12,1033,Moderate
```

**CODE and Result:**

## Importing Necessary libraries

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

[1] ✓ 2.2s

## Load the dataset

+ Code + Markdown

```
df = pd.read_csv('road_transport_records.csv')
```

[2] ✓ 0.0s

### Preprocessing

```
X = df[['Length', 'NumberOfBends', 'TrafficVolume']]
y = df['AccidentRisk'].map({'Low': 0, 'Moderate': 1, 'High': 2})  # Encode the target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[3] ✓ 0.0s

### Train

```
clf = DecisionTreeClassifier(criterion='entropy', random_state=42)

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

[4] ✓ 0.0s

### Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

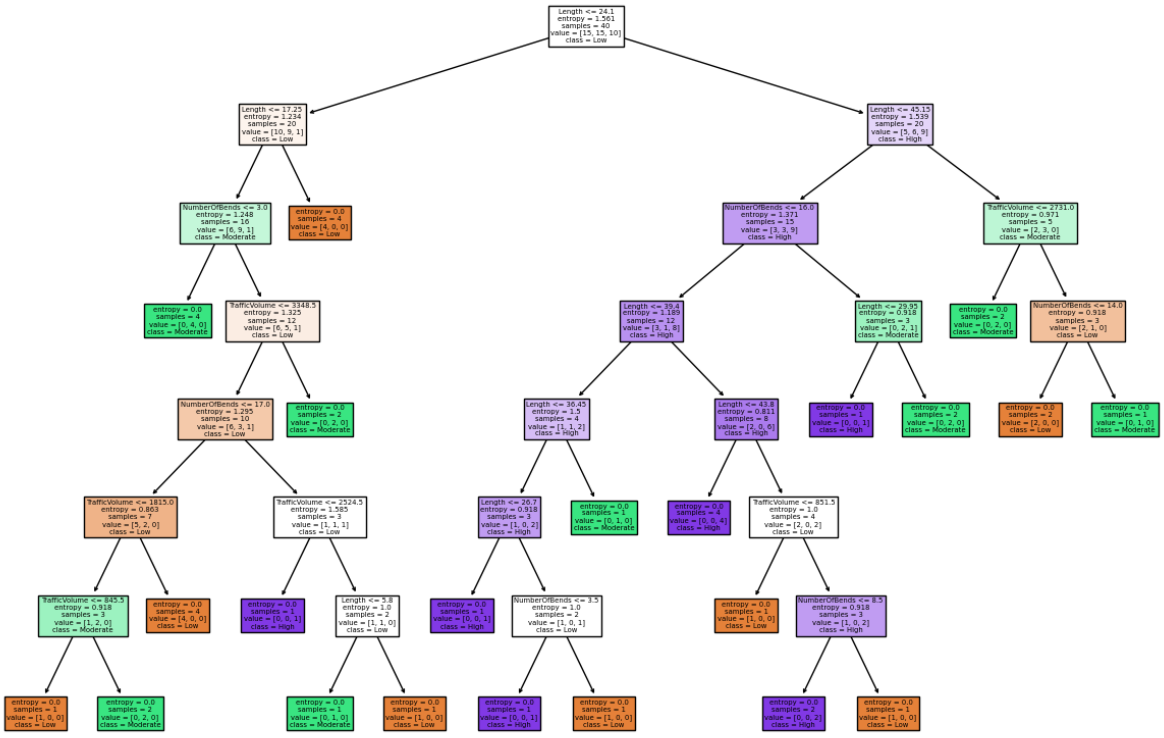
[5] ✓ 0.0s

## Result

```
print(f'Accuracy: {accuracy * 100:.2f}%')
print('Confusion Matrix:')
print(conf_matrix)
print('Classification Report:')
print(class_report)

plt.figure(figsize=(15,10))
plot_tree(clf, feature_names=['Length', 'NumberOfBends', 'TrafficVolume'], class_names=['Low', 'Moderate', 'High'], filled=True)
plt.show()
```

[6] ✓ 1.1s

```
Accuracy: 20.00%
Confusion Matrix:
[[1 1 1]
 [2 1 1]
 [2 1 0]]
Classification Report:
              precision    recall  f1-score   support

           0       0.20      0.33      0.25         3
           1       0.33      0.25      0.29         4
           2       0.00      0.00      0.00         3

    accuracy                           0.20        10
   macro avg       0.18      0.19      0.18        10
weighted avg       0.19      0.20      0.19        10
```



**CONCLUSION :** We were able to get the desired result.

Link to dataset: https://github.com/muditsultania/DataMining-Lab-2/blob/main/road_transport_records.csv