CBS3007

DATA MINING AND ANALYSIS

DA 2

HARSHITH KUMAR

21BBS0163

Question 1

AIM:

Implement a model that will recommend a strict diet is necessary or not for a patient using the naïve Bayes classification algorithm.

LIBRARIES USED:

Pandas, Scikit Learn

SAMPLE DATASET:

https://github.com/harshith363/Lab-Mining/blob/main/DA2%20main/naive%20bayer/diet.csv

CODE:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy_score, classification_report


print("21BBS0163 HARSHITH KUMAR")


df = pd.read_csv('diet.csv')


print(df.head())


le = LabelEncoder()
```

```
df['Gender'] = le.fit_transform(df['Gender'])
df['Physical Activity Level'] = le.fit_transform(df['Physical Activity Level'])
df['Dietary Habit'] = le.fit_transform(df['Dietary Habit'])
df['Strict Diet'] = le.fit_transform(df['Strict Diet'])

X = df[['Gender', 'Age', 'Weight', 'Height', 'BMI',
       'Physical Activity Level']]
y = df['Strict Diet']

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

model = GaussianNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT:

```
PS C:\Users\harsh\OneDrive\Documents\VIT\FALLSEM 24-25\DATA MINING\Lab Mining\DA2 main\naive bayer> py .\naiveBayer.py
21BBS0163 HARSHITH KUMAR
   Gender  Age  Weight  Height   BMI Physical Activity Level Dietary Habit Strict Diet
0    Male   25      75     175  24.5       Moderately Active      Moderate          No
1  Female   30      65     160  25.4              Sedentary     Unhealthy         Yes
2    Male   45      90     170  31.1          Lightly Active     Unhealthy         Yes
3  Female   28      55     165  20.2       Moderately Active       Healthy          No
4    Male   35      82     180  25.3          Lightly Active      Moderate          No
Accuracy: 0.9166666666666666
Classification Report:
              precision    recall  f1-score   support

           0       0.90      1.00      0.95         9
           1       1.00      0.67      0.80         3

    accuracy                           0.92        12
   macro avg       0.95      0.83      0.87        12
weighted avg       0.92      0.92      0.91        12

PS C:\Users\harsh\OneDrive\Documents\VIT\FALLSEM 24-25\DATA MINING\Lab Mining\DA2 main\naive bayer>
```

RESULT:

Successfully implemented Naïve Bayer classification

Question 2

AIM:

Implement K-means method of clustering

LIBRARIES USED:

Pandas, Scikit Learn

SAMPLE DATASET:

https://github.com/harshith363/Lab-Mining/blob/main/DA2%20main/K%20means/diet_2.csv

CODE:

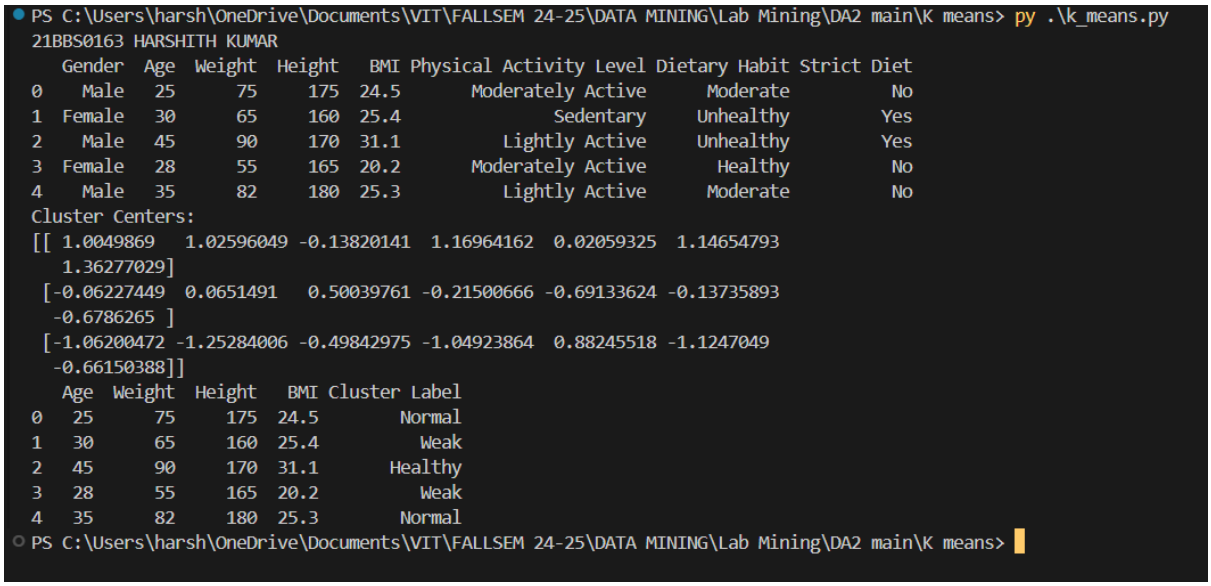
```
import pandas as pd

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt


print("21BBS0163 HARSHITH KUMAR")


file_path = 'diet_2.csv'

data = pd.read_csv(file_path)
```

```
print(data.head())

label_encoders = {}
for column in ['Gender', 'Physical Activity Level', 'Dietary Habit', 'Strict Diet']:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le

features = ['Age', 'Weight', 'Height', 'BMI',
            'Physical Activity Level', 'Dietary Habit', 'Strict Diet']
X = data[features]

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

kmeans = KMeans(n_clusters=3, random_state=42)
data['Cluster'] = kmeans.fit_predict(X_scaled)

cluster_labels = {0: 'Healthy', 1: 'Normal', 2: 'Weak'}
data['Cluster Label'] = data['Cluster'].map(cluster_labels)

print("Cluster Centers:")
print(kmeans.cluster_centers_)

print(data[['Age', 'Weight', 'Height', 'BMI', 'Cluster Label']].head())

plt.figure(figsize=(8, 6))
plt.scatter(data['Age'], data['BMI'], c=data['Cluster'],
```

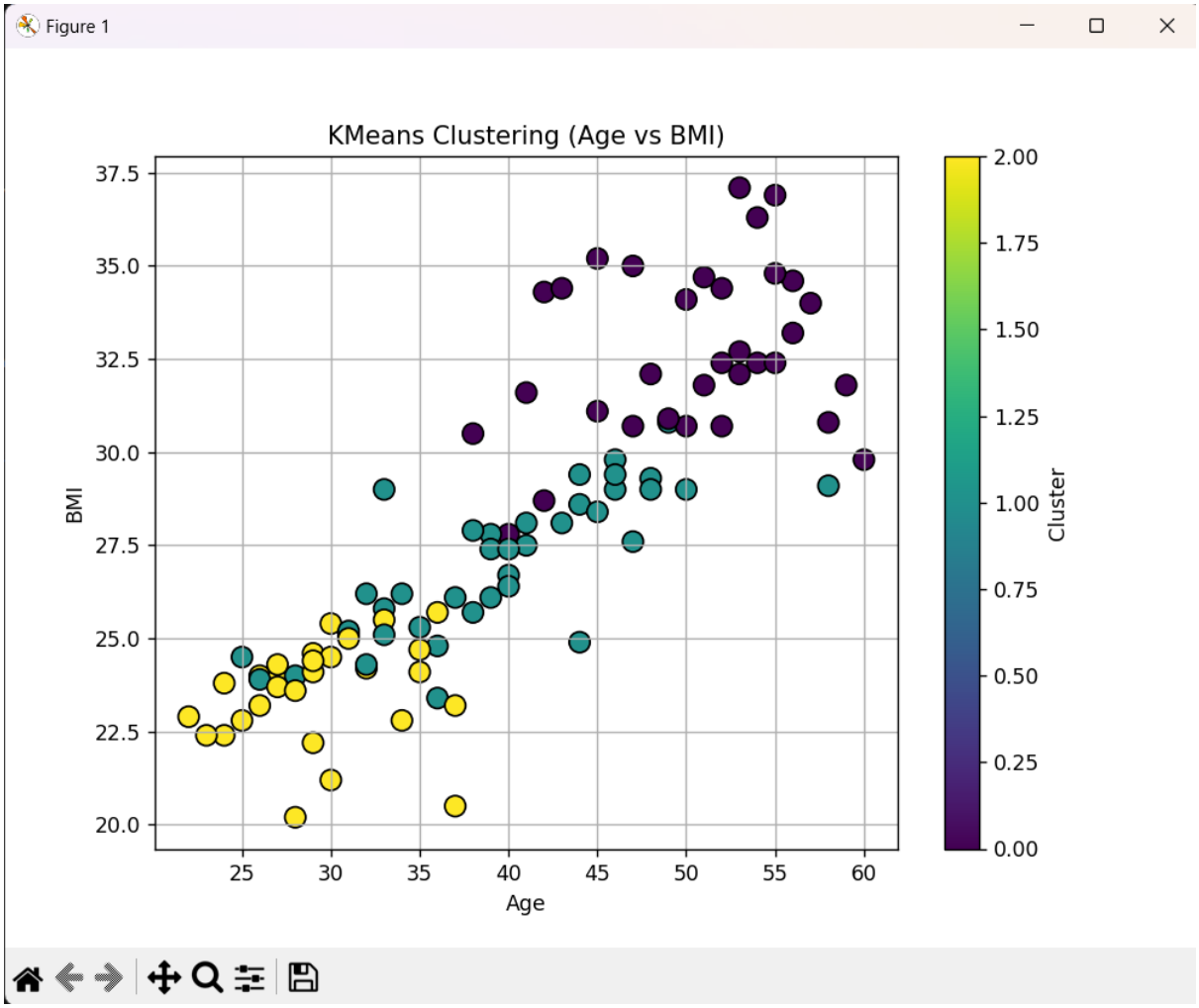
```
        cmap='viridis', marker='o', edgecolor='k', s=100)
plt.xlabel('Age')
plt.ylabel('BMI')
plt.title('KMeans Clustering (Age vs BMI)')
plt.colorbar(label='Cluster')
plt.grid(True)
plt.show()
```

s

OUTPUT:



```
PS C:\Users\harsh\OneDrive\Documents\VIT\FALLSEM 24-25\DATA MINING\Lab Mining\DA2 main\K means> py .\k_means.py
21BBS0163 HARSHITH KUMAR
   Gender  Age  Weight  Height   BMI Physical Activity Level Dietary Habit Strict Diet
0    Male   25      75     175  24.5       Moderately Active      Moderate          No
1  Female   30      65     160  25.4              Sedentary     Unhealthy         Yes
2    Male   45      90     170  31.1          Lightly Active     Unhealthy         Yes
3  Female   28      55     165  20.2       Moderately Active       Healthy          No
4    Male   35      82     180  25.3          Lightly Active      Moderate          No
Cluster Centers:
[[ 1.0049869   1.02596049 -0.13820141  1.16964162  0.02059325  1.14654793
   1.36277029]
 [-0.06227449  0.0651491   0.50039761 -0.21500666 -0.69133624 -0.13735893
  -0.6786265 ]
 [-1.06200472 -1.25284006 -0.49842975 -1.04923864  0.88245518 -1.1247049
  -0.66150388]]
   Age  Weight  Height   BMI Cluster Label
0   25      75     175  24.5        Normal
1   30      65     160  25.4          Weak
2   45      90     170  31.1       Healthy
3   28      55     165  20.2          Weak
4   35      82     180  25.3        Normal
PS C:\Users\harsh\OneDrive\Documents\VIT\FALLSEM 24-25\DATA MINING\Lab Mining\DA2 main\K means>
```



RESULT:

Successfully performed K means clustering

Question 3

AIM:

Implement the ID3 algorithm on the dataset to recommend the decision tree to classify the data.

LIBRARIES USED:

Pandas, Scikit Learn, Matplotlib

SAMPLE DATASET:

https://github.com/harshith363/Lab-Mining/blob/main/DA2%20main/ID/data.csv

CODE:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
import math
from sklearn.preprocessing import LabelEncoder

print("21BBS0163 HARSHITH KUMAR")

# Load the dataset
df = pd.read_csv('data.csv')
df.head()

# Encode the target variable
label_encoder = LabelEncoder()
df['AccidentRisk'] = label_encoder.fit_transform(df['AccidentRisk'])

# Function to calculate entropy

def calculate_entropy(data, target_column):
    total_rows = len(data)
    target_values = data[target_column].unique()

    entropy = 0
    for value in target_values:
        value_count = len(data[data[target_column] == value])
        proportion = value_count / total_rows
        entropy -= proportion * math.log2(proportion) if proportion != 0 else 0
```

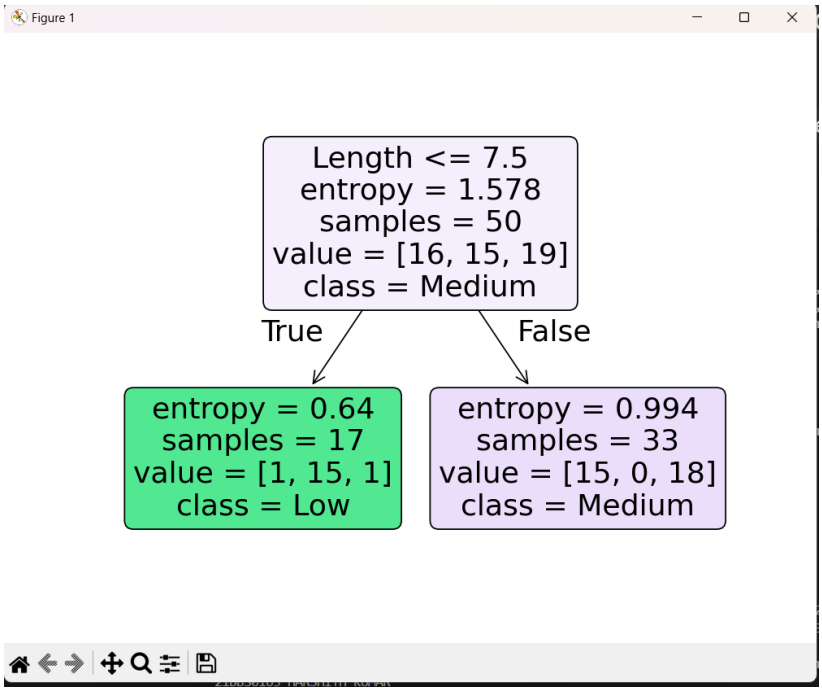
```
    return entropy

# Function to calculate information gain

def calculate_information_gain(data, feature, target_column, entropy_outcome):
    unique_values = data[feature].unique()
    weighted_entropy = 0

    for value in unique_values:
        subset = data[data[feature] == value]
        proportion = len(subset) / len(data)
        weighted_entropy += proportion * \
            calculate_entropy(subset, target_column)

    information_gain = entropy_outcome - weighted_entropy
    return information_gain

# Calculate the entropy of the target variable
entropy_outcome = calculate_entropy(df, 'AccidentRisk')

# Calculate and print entropy and information gain for each feature
print("\nEntropy and Information Gain for each feature:")
for column in df.columns[:-1]:
    entropy = calculate_entropy(df, column)
    information_gain = calculate_information_gain(
        df, column, 'AccidentRisk', entropy_outcome)
    print(f"{column} - Entropy: {entropy:.3f}, Information Gain: {information_gain:.3f}")

# Feature selection for the first step in making decision tree
```

```
selected_feature = 'Length'  # Example feature

# Ensure the selected feature is numeric
if df[selected_feature].dtype == 'object':
    df[selected_feature] = label_encoder.fit_transform(df[selected_feature])

# Prepare the data for training
X = df[[selected_feature]]
y = df['AccidentRisk']

# Create and train a decision tree
clf = DecisionTreeClassifier(criterion='entropy', max_depth=1)
clf.fit(X, y)

# Plot the decision tree
plt.figure(figsize=(8, 6))
plot_tree(clf, feature_names=[
        selected_feature], class_names=label_encoder.classes_, filled=True,
rounded=True)
plt.show()

# Implement the ID3 algorithm

def id3(data, target_column, features):
    # If all target values are the same, return that value (leaf node)
    if len(data[target_column].unique()) == 1:
        return data[target_column].iloc[0]

    # If no more features, return the most common target value
    if len(features) == 0:
```

```
        return data[target_column].mode().iloc[0]

    # Find the feature with the highest information gain
    best_feature = max(features, key=lambda x: calculate_information_gain(
        data, x, target_column, entropy_outcome))

    # Create a new decision tree with the best feature as a node
    tree = {best_feature: {}}

    # Remove the best feature from the remaining features
    features = [f for f in features if f != best_feature]

    # Recursively split the data by the feature values
    for value in data[best_feature].unique():
        subset = data[data[best_feature] == value]
        subtree = id3(subset, target_column, features)
        tree[best_feature][value] = subtree

    return tree

# List of features (excluding the target column)
features = list(df.columns[:-1])

# Build the decision tree using the ID3 algorithm
decision_tree = id3(df, 'AccidentRisk', features)

# Print the resulting decision tree
print("\nGenerated Decision Tree using ID3 algorithm:")
print(decision_tree)
```

OUTPUT:

```
PS C:\Users\harsh\OneDrive\Documents\VIT\FALLSEM 24-25\DATA MINING\Lab Mining\DA2 main\ID> py .\ID_main.py
21BBS0163 HARSHITH KUMAR

Entropy and Information Gain for each feature:
Road ID - Entropy: 5.644, Information Gain: 1.578
Length - Entropy: 3.064, Information Gain: 1.043
Numberof_Bends - Entropy: 2.717, Information Gain: 0.950
Trafficvolume - Entropy: 4.521, Information Gain: 1.425

Generated Decision Tree using ID3 algorithm:
{'Road ID': {1: 0, 2: 1, 3: 0, 4: 1, 5: 2, 6: 2, 7: 1, 8: 0, 9: 0, 10: 1, 11: 2, 12: 2, 13: 1, 14: 2, 15: 0, 16: 2, 17: 1, 18: 2, 19: 0, 20: 1, 21: 0, 22: 2, 23: 1, 24
: 2, 25: 0, 26: 2, 27: 1, 28: 2, 29: 0, 30: 1, 31: 2, 32: 0, 33: 0, 34: 1, 35: 2, 36: 2, 37: 2, 38: 0, 39: 0, 40: 1, 41: 2, 42: 0, 43: 1, 44: 2, 45: 2, 46: 0, 47: 1, 4
8: 2, 49: 0, 50: 1}}
PS C:\Users\harsh\OneDrive\Documents\VIT\FALLSEM 24-25\DATA MINING\Lab Mining\DA2 main\ID>
```



RESULT:

Successfully printed the decision tree