VIT®
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

School of Computer Science and Engineering
Fall Semester-2024-25

Course Code : CBS3007

Course: Data Mining and Analytics

Alan Thomas

21BBS0115

Github link for the datasets and code-

https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA2

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.

LIBRARIES USED: Pandas, Numpy, Scikit Learn

Dataset : https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA2/Q1

SECTION 1

Sample Input

	A	B	C	D	E	F	G	H
1	Patient ID	Weight (kg)	Height (cm)	BMI	Age	Physical Activity	Needs_Diet	Gender_Male
2	1	101	151	44.3	58	0	1	TRUE
3	2	64	155	26.64	45	2	1	FALSE
4	3	110	191	30.15	24	3	1	TRUE
5	4	70	153	29.9	26	1	1	FALSE
6	5	73	178	23.04	25	1	0	TRUE
7	6	52	167	18.65	29	1	0	TRUE
8	7	71	175	23.18	51	0	0	TRUE
9	8	102	193	27.38	50	0	1	FALSE
10	9	51	183	15.23	65	0	0	FALSE
11	10	79	159	31.25	40	0	1	FALSE
12	11	87	185	25.42	41	1	1	FALSE
13	12	51	163	19.2	54	1	0	TRUE
14	13	113	180	34.88	52	2	1	FALSE
15	14	109	197	28.09	61	0	1	TRUE
16	15	70	164	26.03	57	2	1	TRUE
17	16	82	157	33.27	39	0	1	FALSE
18	17	107	163	40.27	44	0	1	FALSE
19	18	71	172	24	52	0	0	FALSE
20	19	98	189	27.43	18	1	1	FALSE
21	20	108	170	37.37	52	0	1	FALSE
22	21	91	165	33.43	54	2	1	FALSE
23	22	109	194	28.96	64	2	1	FALSE
24	23	64	167	22.95	31	1	0	FALSE
25	24	111	196	28.89	20	1	1	TRUE
26	25	111	173	37.09	18	1	1	FALSE
27	26	96	175	31.35	22	1	1	FALSE
28	27	111	174	36.66	43	2	1	TRUE
29	28	100	194	26.57	31	3	1	FALSE
30	29	104	190	28.81	56	0	1	TRUE
31	30	113	178	35.66	44	0	1	TRUE

patient_enrollment_diet

Code

```
import pandas as pd
from sklearn import naive_bayes
from sklearn import model_selection
from sklearn import metrics
from sklearn import preprocessing as pp
import numpy as np

data = pd.read_csv('patient_enrollment_diet.csv')
# encoding
le = pp.LabelEncoder()

data['Physical Activity'] = le.fit_transform(data['Physical Activity'])

temp = list(data.columns)
temp[7] = 'Physical Activity'
data.columns = temp
data = pd.get_dummies(data, columns=['Gender'], drop_first=True)

y = data['Needs_Diet']
X = data.drop(['Needs_Diet' , 'Patient ID'],axis = 1)

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
test_size=0.25, random_state=42)

nb = naive_bayes.GaussianNB()

nb.fit(X_train,y_train)
y_pred = nb.predict(X_test)

print('accuracy score:' , metrics.accuracy_score(y_test, y_pred))

print('precision score:' , metrics.precision_score(y_test, y_pred))

print('recall score:' , metrics.recall_score(y_test, y_pred))
```

print('F1 score:' , metrics.f1_score(y_test, y_pred))

Output

21BBS0115

```
data.head(10)
```

	Patient ID	Gender	Weight (kg)	Height (cm)	BMI	Age	Physical Activity	Needs_Diet
0	1	Male	101	151	44.30	58	Lightly active	1
1	2	Female	64	155	26.64	45	Sedentary	1
2	3	Male	110	191	30.15	24	Very active	1
3	4	Female	70	153	29.90	26	Moderately active	1
4	5	Male	73	178	23.04	25	Moderately active	0
5	6	Male	52	167	18.65	29	Moderately active	0
6	7	Male	71	175	23.18	51	Lightly active	0
7	8	Female	102	193	27.38	50	Lightly active	1
8	9	Female	51	183	15.23	65	Lightly active	0
9	10	Female	79	159	31.25	40	Lightly active	1

Next steps: Generate code with data | View recommended plots | New interactive sheet

21BBS0115

```
print('accuracy score:' , metrics.accuracy_score(y_test, y_pred))

print('precision score:' , metrics.precision_score(y_test, y_pred))

print('recall score:' , metrics.recall_score(y_test, y_pred))

print('F1 score:' , metrics.f1_score(y_test, y_pred))
```

```
accuracy score: 0.6923076923076923
precision score: 0.7272727272727273
recall score: 0.8888888888888888
F1 score: 0.7999999999999999
```

So we see that the model was able to predict correctly 70% of the time whether a strict diet was needed or not.

```
[ ] y_pred

array([1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1])

[ ] np.array(y_test)

array([1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1])
```

SECTION 2

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

Libraries : Numpy, Pandas, sklearn, seaborn
Dataset : https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA2/Q2

Sample Input

	A	B	C	D	E	F	G	H	I	J	K
1	Name	Gender	Age	Weight (kg	Height (cm	BMI	Enrolled in	Diet Type	Activity Le	Creatine	
2	Aarav	Male	25	63.57121	155.5815	26.3	1	Vegetariar	High	1.94	
3	Ananya	Female	17	58.27933	159.3064	23	1	Non-Veget	High	0.65	
4	Neha	Female	31	75.6243	182.3772	22.7	1	Non-Veget	High	5.22	
5	Kavya	Female	25	60.44178	188.2885	17	0	Vegetariar	Low	1.05	
6	Priya	Female	28	84.6623	174.149	27.9	1	Keto	High	2.66	
7	Siddharth	Male	19	74.32142	183.1762	22.2	0	Non-Veget	High	0.83	
8	Aarav	Male	24	82.05739	189.4089	22.9	1	Vegetariar	Moderate	1.51	
9	Simran	Female	28	60.69289	164.2108	22.5	0	Vegetariar	High	1.18	
10	Harsh	Male	24	60.71909	165.1782	22.3	1	Vegan	Low	3.03	
11	Aarav	Male	27	69.04077	152.6475	29.6	0	Non-Veget	High	1.71	
12	Riya	Female	31	60.00051	155.5852	24.8	0	Keto	High	2.7	
13	Kavya	Female	24	89.9064	155.5333	37.2	0	Vegetariar	Low	4.29	
14	Aditya	Male	22	82.72278	166.8864	29.7	1	Keto	Moderate	2.73	
15	Pooja	Female	25	88.98774	184.4312	26.2	1	Vegetariar	High	3.04	
16	Kavya	Female	20	65.2725	156.3263	26.7	1	Vegan	High	0.9	
17	Harsh	Male	20	85.46815	161.9378	32.6	1	Non-Veget	Moderate	0.79	
18	Harsh	Male	33	87.14727	173.9578	28.8	0	Vegetariar	Low	2.7	
19	Aditi	Female	24	57.02738	185.1204	16.6	1	Vegetariar	High	0.93	
20	Priya	Female	34	81.80421	155.1357	34	0	Non-Veget	Moderate	0.65	
21	Neha	Female	34	81.43428	177.5947	25.8	0	Keto	High	1.54	
22	Simran	Female	33	70.80201	159.9162	27.7	1	Vegan	Low	4.04	
23	Aniket	Male	24	55.25168	178.3136	17.4	1	Non-Veget	Low	1.3	
24	Rahul	Male	19	72.99546	161.1391	28.1	0	Non-Veget	High	1.76	
25	Aniket	Male	35	71.54349	181.3848	21.7	0	Non-Veget	Low	1.31	
26	Siddharth	Male	28	69.82525	168.681	24.5	1	Vegetariar	Low	3.01	
27	Kavya	Female	20	63.70297	157.6084	25.6	0	Non-Veget	Moderate	0.65	
28	Rohan	Male	31	63.74323	186.9306	18.2	0	Vegetariar	Low	5.5	
29	Vivaan	Male	24	60.82089	169.4256	21.2	1	Keto	Low	3.4	
30	Siddharth	Male	17	89.48581	160.6081	34.7	0	Vegan	Moderate	4.14	
31	Sakshi	Female	23	65.38477	188.7484	18.4	1	Vegan	Low	1.11	

Code

```
import pandas as pd
from sklearn import naive_bayes
from sklearn import model_selection
from sklearn import metrics
from sklearn import preprocessing as pp
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.preprocessing import StandardScaler
import numpy as np
```

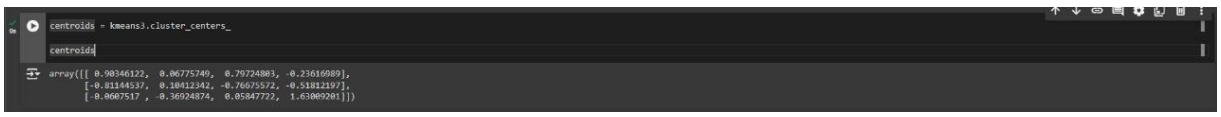
```
import seaborn as sns
data = pd.read_csv('patient_dataset.csv')
data.head(10)
data = data.drop(['Name' , 'Gender' , 'Age' , 'Enrolled in Diet Plan' , 'Diet Type' , 'Activity Level'] , axis = 1)
# check missing vlaues
data.any().isna().sum()
data.head()
num_clusters = [i for i in range(2, 7)]
```

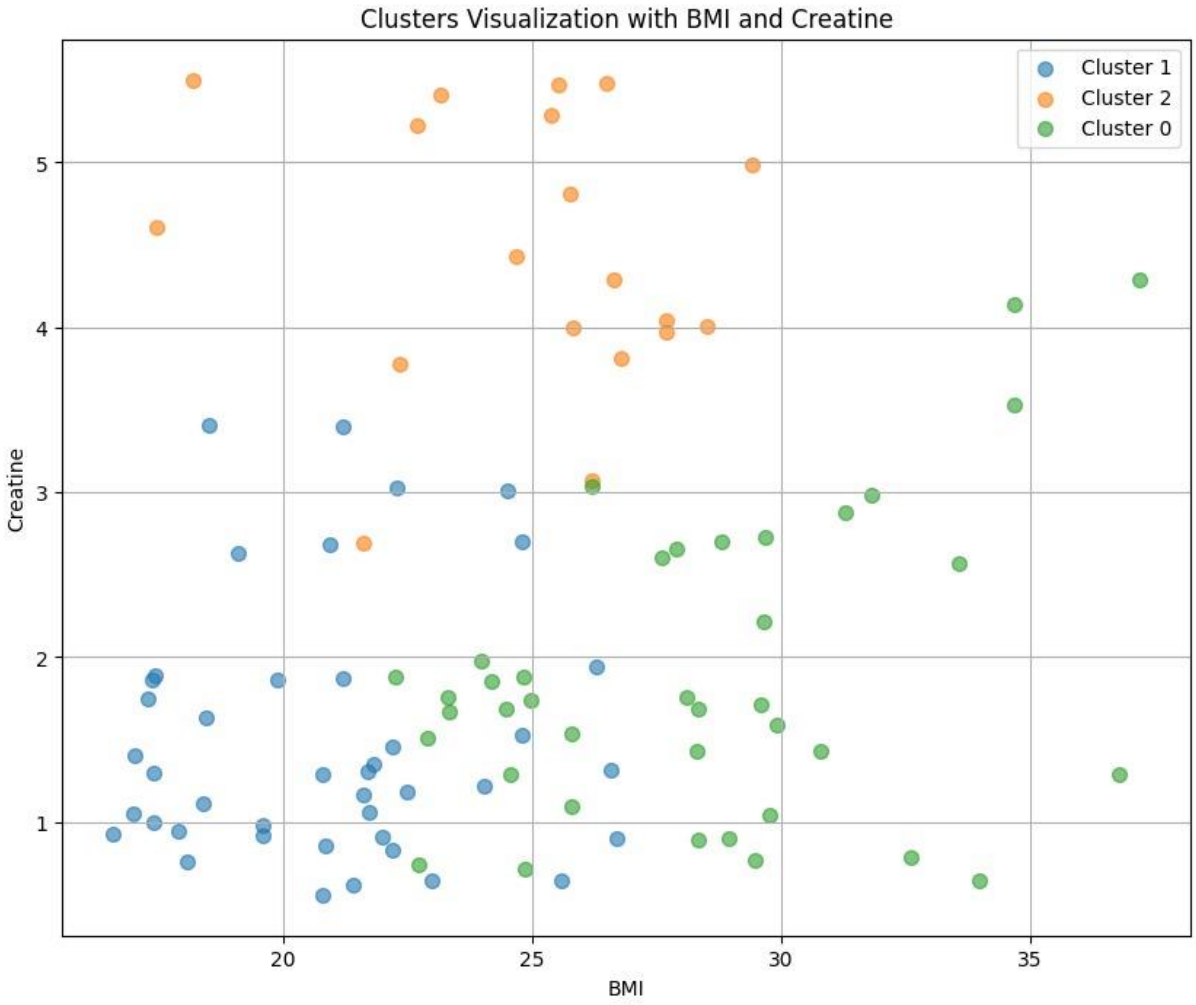
```
def kmeans_inertia(num_clusters, x_vals):
```

```
  inertia = []
  for num in num_clusters:
      kms = KMeans(n_clusters=num, random_state=42)
      kms.fit(x_vals)
      inertia.append(kms.inertia_)


  return inertia
X_scaled = StandardScaler().fit_transform(data)
inertia = kmeans_inertia(num_clusters,X_scaled)
inertia
kmeans3 = KMeans(n_clusters=3, random_state=42)
kmeans3.fit(X_scaled)


data['cluster'] = kmeans3.labels_
def give_label(cluster_num):
 if cluster_num == 0:
  return 'Weak'
 elif cluster_num == 1:
  return 'Normal'
 else:
  return 'Healthy'


data['Class'] = data['cluster'].apply(give_label)
```

```
import matplotlib.pyplot as plt


df = data.copy()


features = ['BMI', 'Creatine']
X = df[features]


# Plotting the clusters
plt.figure(figsize=(10, 8))
for cluster in df['cluster'].unique():
    cluster_data = df[df['cluster'] == cluster]
    plt.scatter(cluster_data['BMI'], cluster_data['Creatine'], label=f'Cluster {cluster}', s=50, alpha=0.6)


plt.xlabel('BMI')
plt.ylabel('Creatine')
plt.title('Clusters Visualization with BMI and Creatine')
plt.legend()
plt.grid(True)
plt.show()
centroids = kmeans3.cluster_centers_


print(centroids)
```

Output :






```
data.head(10)
```

	Weight (kg)	Height (cm)	BMI	Creatine	cluster	Class
0	63.571215	155.581517	26.3	1.94	1	Normal
1	58.279333	159.306436	23.0	0.65	1	Normal
2	75.624299	182.377218	22.7	5.22	2	Healthy
3	60.441782	188.288523	17.0	1.05	1	Normal
4	84.662303	174.149041	27.9	2.66	0	Weak
5	74.321422	183.176187	22.2	0.83	1	Normal
6	82.057392	189.408861	22.9	1.51	0	Weak
7	60.692893	164.210828	22.5	1.18	1	Normal
8	60.719087	165.178218	22.3	3.03	1	Normal
9	69.040768	152.647544	29.6	1.71	0	Weak

SECTION 3

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

Libraries Used : Numpy, Pandas, sklearn, matplotlib, seaborn

Dataset : https://github.com/ALANT535/DATA-MINING-RESOURCES/tree/main/DA2/Q3

Sample Input

road_transport_records

Road ID	Length (km)	Number of Bends	Traffic Volume	Accident Risk
SH12	50	25	18000	High
NH48	250	26	5000	High
NH27	140	30	6000	High
NH31	75	33	22000	Very High
NH48	80	8	22000	Very High
NH37	60	55	28000	Extreme
NH31	290	33	3200	Very High
SH2	300	15	12000	Moderate
SH38	600	33	27000	Extreme
NH31	260	10	20000	High
NH61	120	12	9000	Moderate
SH50	250	25	13000	High
NH27	600	31	30000	Extreme
SH17	80	38	16000	Very High
NH16	350	50	28000	Extreme
NH75	275	22	40000	Extreme
NH58	300	28	16000	High
SH10	210	18	35000	Extreme
NH75	260	12	20000	High
SH10	320	19	5000	Moderate
NH31	500	26	23000	Very High
SH38	75	17	2000	Moderate
NH1	190	17	15000	Moderate
SH1	75	38	9000	Very High
NH9	350	14	17000	High
SH25	130	38	40000	Extreme

Code and Output Code

Code :

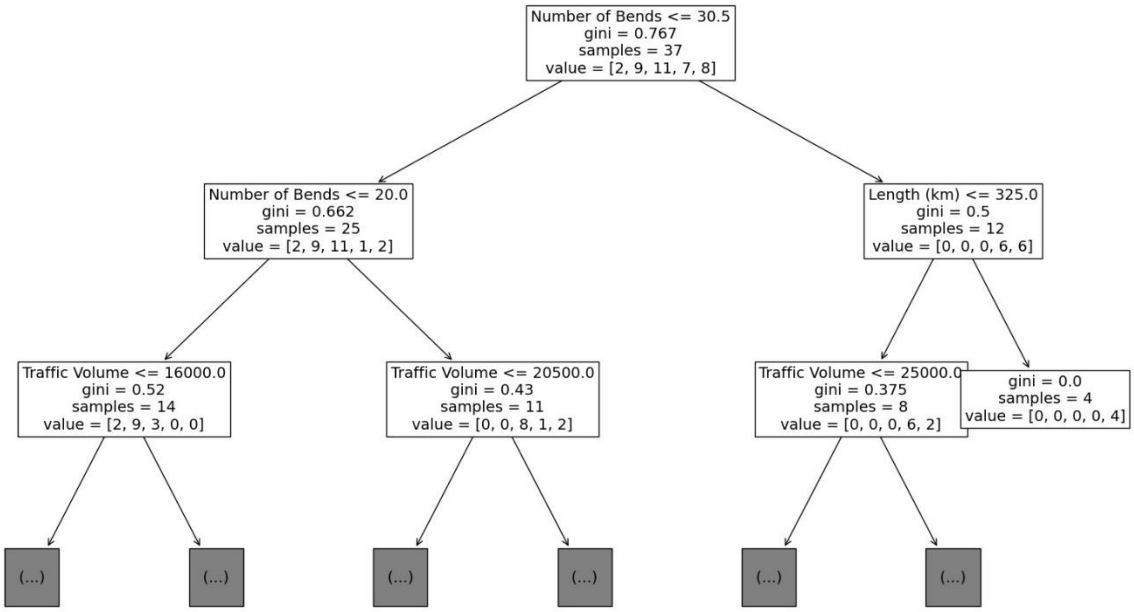
```
import pandas as pd
import math
import matplotlib.pyplot as plt
import networkx as nx
from networkx.drawing.nx_agraph import
graphviz_layout



df =
pd.read_csv(r"DA2\Q3\road_transport_record
s.csv")



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



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



def id3(data, target_column, features):
  if len(data[target_column].unique()) == 1:
      return data[target_column].iloc[0]
  if len(features) == 0:
      return data[target_column].mode().iloc[0]
  entropy_outcome = calculate_entropy(data,
target_column)
  best_feature = max(features, key=lambda x:
calculate_information_gain(data, x,
target_column, entropy_outcome))
  tree = {best_feature: {}}
  features = [f for f in features if f !=
best_feature]
  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
```

```
def plot_tree(tree, parent_name, graph,
depth, max_depth):
  if depth > max_depth:
    return
  if isinstance(tree, dict):
    feature = list(tree.keys())[0]
    for value, subtree in tree[feature].items():
      node_name = f"{feature} = {value}"
      graph.add_node(node_name)
      graph.add_edge(parent_name,
node_name)
      plot_tree(subtree, node_name, graph,
depth + 1, max_depth)
  else:
    leaf_name = f"Accident Risk: {tree}"
    graph.add_node(leaf_name)
    graph.add_edge(parent_name,
leaf_name)


def visualize_decision_tree(decision_tree,
max_depth=3):
  graph = nx.DiGraph()
  root_name = list(decision_tree.keys())[0]
  graph.add_node(root_name)
  plot_tree(decision_tree, root_name, graph,
0, max_depth)
  plt.figure(figsize=(20, 15))
  pos = nx.spring_layout(graph, seed=42,
k=0.5, iterations=50)
```

```
    nx.draw(graph, pos, with_labels=True,
node_size=3500, node_color="lightblue",
font_size=12, font_weight="bold",
arrows=True, connectionstyle='arc3,rad=0.1')
    plt.title("Decision Tree Visualization")
    plt.show()



features = ['Length (km)', 'Number of Bends',
'Traffic Volume']
decision_tree = id3(df, 'Accident Risk',
features)



print("\nGenerated Decision Tree using ID3
algorithm:")
print(decision_tree)



print("\n\nNow printing it-")
visualize_decision_tree(decision_tree)
```

Output :




```
PS C:\Users\LENOVO\Documents\Important_documents\VIT\Semesters\sem7\DATA MINING\DA> python -u "c:\Users\LENOVO\Documents\Important_do
cuments\VIT\Semesters\sem7\DATA MINING\DA\DA2\Q3\ID3_algo_tree.py"

Generated Decision Tree using ID3 algorithm:
{'Traffic Volume': {18000: 'High', 5000: {'Length (km)': {250: 'High', 320: 'Moderate', 180: 'Moderate'}}, 6000: 'High', 22000: 'Very
 High', 28000: 'Extreme', 3200: 'Very High', 12000: 'Moderate', 27000: 'Extreme', 20000: 'High', 9000: {'Length (km)': {120: 'Moderat
e', 75: 'Very High', 320: 'Low', 150: 'Very High'}}, 13000: 'High', 30000: 'Extreme', 16000: {'Length (km)': {80: 'Very High', 300: '
High'}}, 40000: 'Extreme', 35000: 'Extreme', 23000: 'Very High', 2000: 'Moderate', 15000: {'Length (km)': {190: 'Moderate', 600: 'Ext
reme', 150: 'High'}}, 17000: 'High', 19000: 'High', 10000: {'Length (km)': {400: 'Low', 190: 'High', 500: 'Extreme'}}, 4500: 'Very Hi
gh', 3000: {'Number of Bends': {40: 'Very High', 29: 'High', 28: 'High'}}, 8000: 'High', 11000: 'Moderate', 25000: 'Very High', 14000
: 'Moderate'}}
```

RESULT:

Created the Decision Tree based on the concept of ID3 algorithm.

XXXXXXXXXXXXXXX