MSPA-3

(EXPERIMENT 7 - EXPERIMENT 8)

Name – Vinay Partap Singh

Course Name – MLT Lab

Section – B

Course Code – CSE-2424

Roll No. – 177

Reg No. – 20010232

EXPERIMENT NO. 7

Name of experiment –

Decision Trees using Scikit-learn

Problem Statement –

Predicting whether a person is infected from COVID-19 or not.

Theory –

Decision trees are a fundamental machine learning technique for both classification and regression tasks. This method represents a predictive model in a tree-like structure, where each internal node corresponds to a feature or attribute, and each leaf node represents a class label or a numerical value. Decision trees make predictions by recursively partitioning the data into subsets based on the most informative features, leading to a hierarchy of decision rules.

The key advantage of decision trees lies in their interpretability, as they offer a transparent visualization of the decision-making process. They are also robust to outliers and can handle both categorical and numerical data. However, decision trees are susceptible to overfitting, which can be mitigated by techniques like pruning. Ensembles, such as Random Forests and Gradient Boosting, leverage multiple decision trees to improve predictive accuracy and generalization.

In summary, decision trees are a versatile and valuable tool in machine learning, offering a simple yet effective approach to decision-making and predictive modeling.

Data Description–

The dataset reports given a person has/has not one of the three symptoms: 'Fever', 'Cough', 'Breathing Issue' and if he is suffering from COVID-19 or not.

Class	2
Samples Total	13
Dimensionality -	3
Features	'Fever', 'Cough', 'Breathing Issue'

Problem Analysis-

Upon critical Analysis of the given Problem, it can be seen that there's a Relationship between the person being COVID-19 infected and if he has symptoms like Fever, Cough, or Breathing Issue as given in the dataset description. This type of problem can easily be dealt with by using Decision Trees for Classification.

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node.

Decision trees can handle both categorical and numerical data. The core algorithm for building decision trees called ID3 which uses Entropy and Information Gain to construct a decision tree. A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

Machine Learning Task- Classification

Predictors-

If a person has or has not

symptoms like: Fever, Cough, Breathing Issue

Response Variable:

Whether a person is infected by COVID-19 or not.

Pseudocode-

Data: X<-Input, Y<-Target

model = DecisionTreeClassifier()

clf = model.fit(X,Y)

prediction = clf.predict(sample)

print(prediction)

plot_decision_tree()

Algorithm for Decision Tree-

1. Place the best attribute of the dataset at the root of the tree.

2. Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.

3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

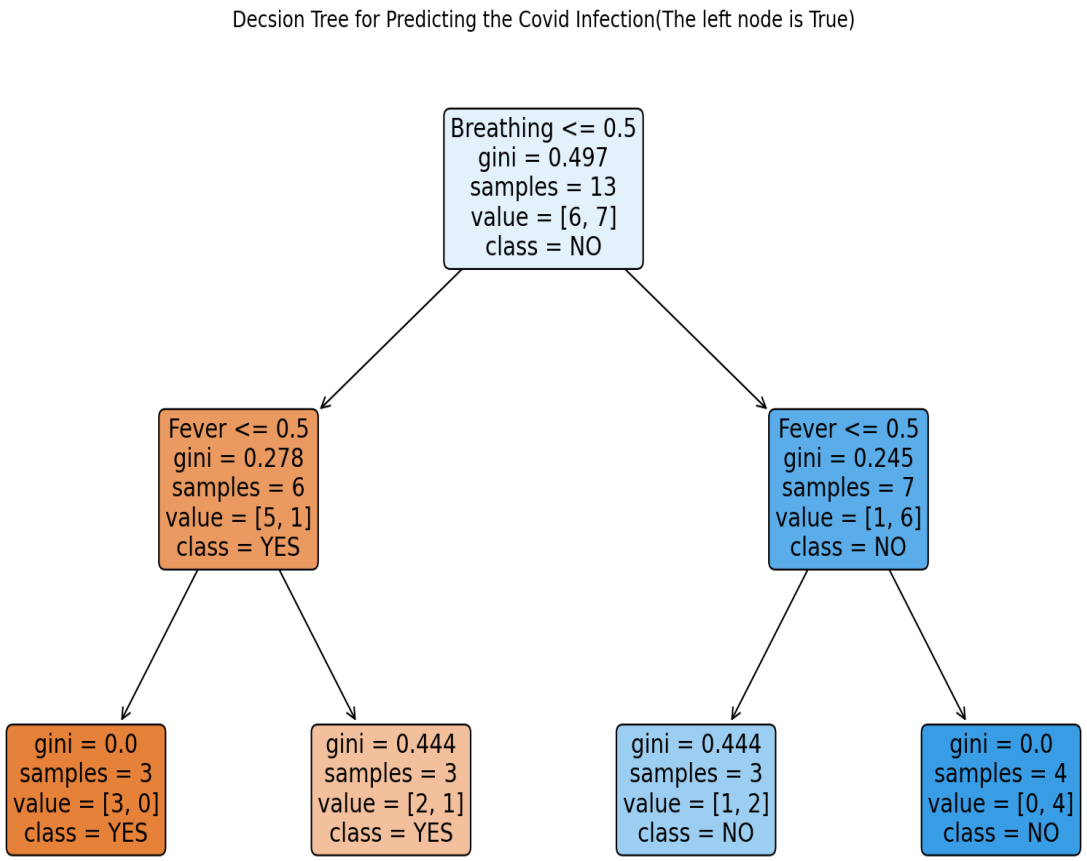
Program –

```
feature_names = ['Fever', 'Cough', 'Breathing','Issue']
X = [ [0,0,0], [1,1,1], [1,1,0],
[1,0,1], [1,1,1], [0,1,0],
[1,0,1], [0,1,1], [1,1,0],
[0,1,0],[0,1,1],[0,1,1],[1,1,0] ]

# Labels for the training data
Y = ['NO', 'YES', 'NO', 'YES',
'YES', 'NO', 'YES', 'YES',
'YES', 'NO', 'YES', 'NO', 'NO']
# Finding Unique Labels
labels = list(set(Y))

# Create a decision tree classier model
from sklearn.tree import DecisionTreeClassifier as DTClf
clf = DTClf(max_depth=4).fit(X,Y)
# Now predict and print the class for the unknown patient(example)
pred = clf.predict([[1,1,0]])
print(f"Person infected? {pred[0]}")

# plotting tree for COVID-19 classication
import matplotlib.pyplot as plt
# plt the figure,
plt.figure(figsize=(30,20))
from sklearn.tree import plot_tree as PT
PT(clf, class_names=labels,feature_names=feature_names,
   rounded = True,filled = True,fontsize=14)
plt.title('Decsion Tree for Predicting the Covid'+' Infection(The
left node is True)')
plt.show()
```

Output –

```
PS D:\Machine Learning Lab> python -
Person infected? NO
```

Graphs-



Conclusion – We have Successfully Predicted if the person is infected from COVID-19 or not given features by Decision Trees for Classification.

EXPERIMENT NO. 8

Name of experiment –

k-means clustering using Scikit-learn

Problem Statement –

BB supermarket mall maintains the customer data using membership cards. They have some basic data about the customers like Customer ID, name, age, gender, number of visits to the supermarket in a year, and the average spending per visit.

They want to understand their customers so that the sense can be given to the marketing team so that they can plan the strategy accordingly. As a data scientist your task is to identify the customer groups and interpret them appropriately after analysing the clusters.

This can be later given to the marketing team to increase the customer base and the profit of the supermarket.

Theory –
K-means clustering is a fundamental machine learning technique used for unsupervised data partitioning and cluster identification. The method starts by selecting a predefined number of clusters (K) and randomly initializing cluster centroids. It then iteratively assigns data points to the nearest centroid and recalculates the centroids based on the mean of the assigned points. This process continues until convergence or a specified number of iterations.

The underlying theory is rooted in optimizing the within-cluster variance, seeking to minimize the sum of squared distances between data points and their assigned cluster centroids. K-means assumes that clusters are spherical and equally sized, making it sensitive to initial centroid placements and not suitable for irregularly shaped or varying-sized clusters.

Despite its simplicity, K-means is widely used for data segmentation, feature engineering, and anomaly detection. Its efficiency makes it suitable for large datasets, but its limitations require careful consideration, often necessitating the use of more advanced clustering techniques when dealing with complex data structures.

Data Description– The BB Supermarket dataset consists of customers like Customer ID, name, age, gender, number of visits to the supermarket in a year(Nvisits), and the average spending per visit(purchase Amount).

Out of these Nvisits and purchase Amount can be used for categorization of customers, rest do not play a very important role.

Class	4
Samples Total	28

Dimensionality	2
Available Features	ID, name, age, gender, Nvisits, purchaseAmount
Selected Features	Nvisits, purchaseAmount

Problem Analysis-

Upon critical Analysis of the given Problem, it can be seen that number of visits and purchase amount play an important role in determining the category of customer.

This type of problem can easily be dealt with by using K Means Clustering. K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science.

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster.

Machine Learning Task- Clustering.

Predictor Variables-

- Nvisits
- Purchase Amount

Response Variable-

Class of Customers:

['Moderate', 'Basic', 'Prime',

'Window Shopper']

Model Parameters- Cluster Centroids learned from training data

Hyper Parameters-

- Number of Clusters (k)
- Number of iterations

Pseudo Code for K-Means Clustering -

1. Choose the number of clusters(k) into which the training instances **X** are to be grouped

2. Select the centroids c_1, c_2, ... c_k randomly

3. Repeat steps 4 and 5 until convergence or until the end of a fixed number of iterations

4. for each data point x_i in **X**: - find the nearest centroid (c_1, c_2, ... c_k) - assign the point to this nearest cluster centroid.

5. for each cluster j = 1..k - new centroid = mean of all points assigned to that cluster

6. End

Program –

```
# Expt. 8 Implementing K-means clustering for BB Supermarket
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import matplotlib.patches as mpatches
from sklearn import preprocessing
# X contains training data samples from a synthetic dataset
# X = Dataset of attributes [Nvisits,purchaseAmount]
X = [ [2,10],[9, 0], [6, 5], [11, 0], [9, 699], [4, 220], [6, 550],[5, 553],
[8, 450],\
[4, 150], [5, 120], [7, 100], [5, 200], [12, 0], [8, 0],[6, 120], \
[7, 404], [5, 388], [6, 225], [8, 350], [8, 236], [4, 167], [9, 400],\
[9, 817], [10, 1010], [7, 25],[11, 825], [11, 700],[10, 20],[8,921]]
```

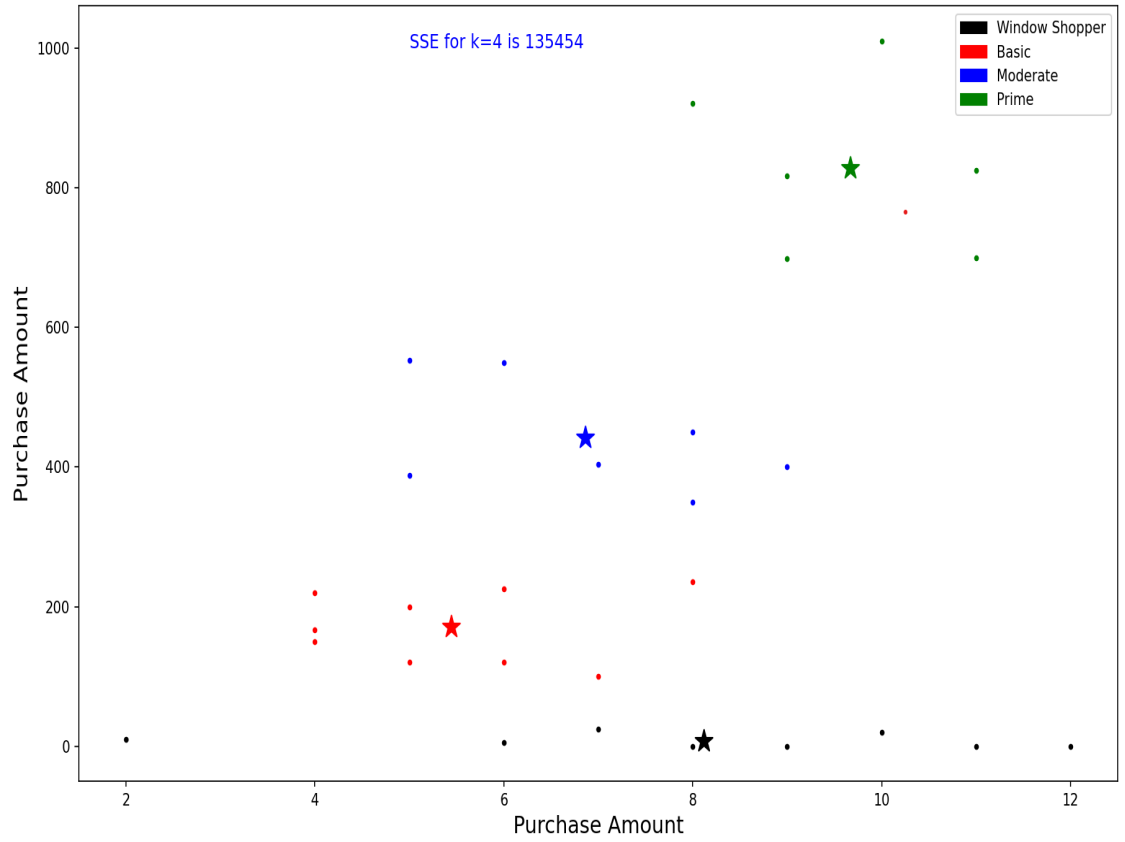
```
# Initializing Hyperparameters
# Number of clusters k (you may find best k by using elbow point)
k = 4
# n_init: Number of times the k-means algorithm is run with different centroid
seeds
model = KMeans(n_clusters=k, n_init=5,max_iter=100)
# Training
# Fitting the input data
kmeans = model.fit(X) # model is trained and stored in kmeans
# Centroid values as the output of the clustering process
Xarr = np.array(X)
centroids = np.around(kmeans.cluster_centers_, decimals=2)
print(f"Cluster Centers are: {centroids}")
# Predicting the cluster labels and Computing SSE
labels = kmeans.predict(X)
print(f"Cluster Labels are: {labels}")
sse=round(kmeans.inertia_)
print(f"The SSE for k= {k} is {sse}")
colors = ['red', 'green', 'blue', 'black','c','m','y']
fig, ax = plt.subplots()
plt.xlabel('Number of Visits')
plt.ylabel('Purchase Amount')
t="SSE for k=" + str(k)+ " is " + str(sse)
plt.text(5, 1000, t, fontsize=12,color="blue")
# Ploting the points within the same cluster with the unique color
save_cluster_point_color={}
for i in range(k):
    points = np.array([X[j] for j in range(len(X)) if labels[j] == i])
    ax.scatter(points[:, 0], points[:, 1], s=7, c=colors[i])
    # plot centroid of that cluster
    ax.scatter(centroids[i, 0], centroids[i, 1], marker='*',
s=200,c=colors[i])
    save_cluster_point_color.update({centroids[i,0]*centroids[i,1]:colors[i]})
ax.set_xlabel('Purchase Amount',fontdict={'size':15})
ax.set_ylabel('Purchase Amount',fontdict={'size':15})
# Gettin the colors for legends of cluster labels
myKeys = list(save_cluster_point_color.keys())
myKeys.sort()
save_cluster_point_color = {i: save_cluster_point_color[i] for i in myKeys}
# predicting class of input pattern
xnew = np.array([[4, 20], [9, 500]])
newLabels = kmeans.predict(xnew)
print("New instances",xnew)
print("Labels:",newLabels)
# categories and cluster labels extraction
classes=['Window Shopper','Basic', 'Moderate', 'Prime']
colors_for_legends=list(save_cluster_point_color.values())
# Cluster Labelling in colors
green_patch = mpatches.Patch(color=colors_for_legends[0], label=classes[0])
blue_patch = mpatches.Patch(color=colors_for_legends[1], label=classes[1])
yellow_patch = mpatches.Patch(color=colors_for_legends[2], label=classes[2])
red_patch = mpatches.Patch(color=colors_for_legends[3], label=classes[3])
```

```
plt.legend(handles=[green_patch, blue_patch, yellow_patch, red_patch, ])
plt.show()
```

Output –

```
Cluster Centers are: [[  5.44 170.89]
 [  9.67 828.67]
 [  6.86 442.14]
 [  8.12   7.5 ]]
Cluster Labels are: [3 3 3 3 1 0 2 2 2 0 0 0 0 3 3 0 2 2 0 2 0 0 2 1 1 3 1 1 3 1]
The SSE for k= 4 is 135454
New instances [[  4  20]
 [  9 500]]
Labels: [3 2]
```

Graph –



Conclusion – In this experiment we learnt about the K-mean clustering and how can we use this data to create the clusters related to BB supermarket customer type segmentation.