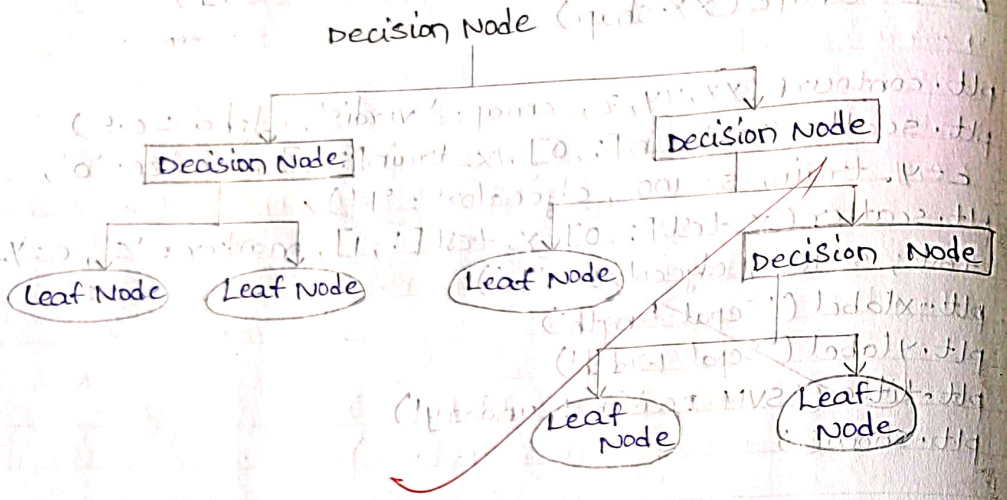


BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY

Experiment - 7

Aim: Write a program to demonstrate the working of decision tree based on ID3 algorithm. Use an appropriate data set for building the decision tree and apply knowledge to classify a new sample.

Description: Decision tree is a supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving classification problems. It is a tree structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

→ In a decision tree there are two nodes, which are the decision node and leaf node. Decision nodes are used to make any decision and have multiple branches. Whereas leaf nodes are the output of those decisions and do not contain any further branches.

→ In order to built a tree, we use the CART algorithm. which stands for classification and Regression Tree algorithm.

Algorithm:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step-3. Continue this process until a code stage is reached where you cannot further classify the nodes, and called the final node as a leaf node.

Attribute Selection Measures: (ASM)

While implementing a decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as Attribute selection measure or ASM. By this measure-ment, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are

- Information Gain
- Gini Index

Information Gain: Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

→ It calculates how much information a feature provides us about a class.

Information Gain =

Entropy (S) - [Cweighted Avg) * Entropy (each feature)]

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.

$$Entropy(S) = -p(Yes)\log_2 p(Yes) - P(no)\log_2 p(no)$$

Gini Index: Gini index is a measure of impurity or purity used while creating a decision tree in the CART.

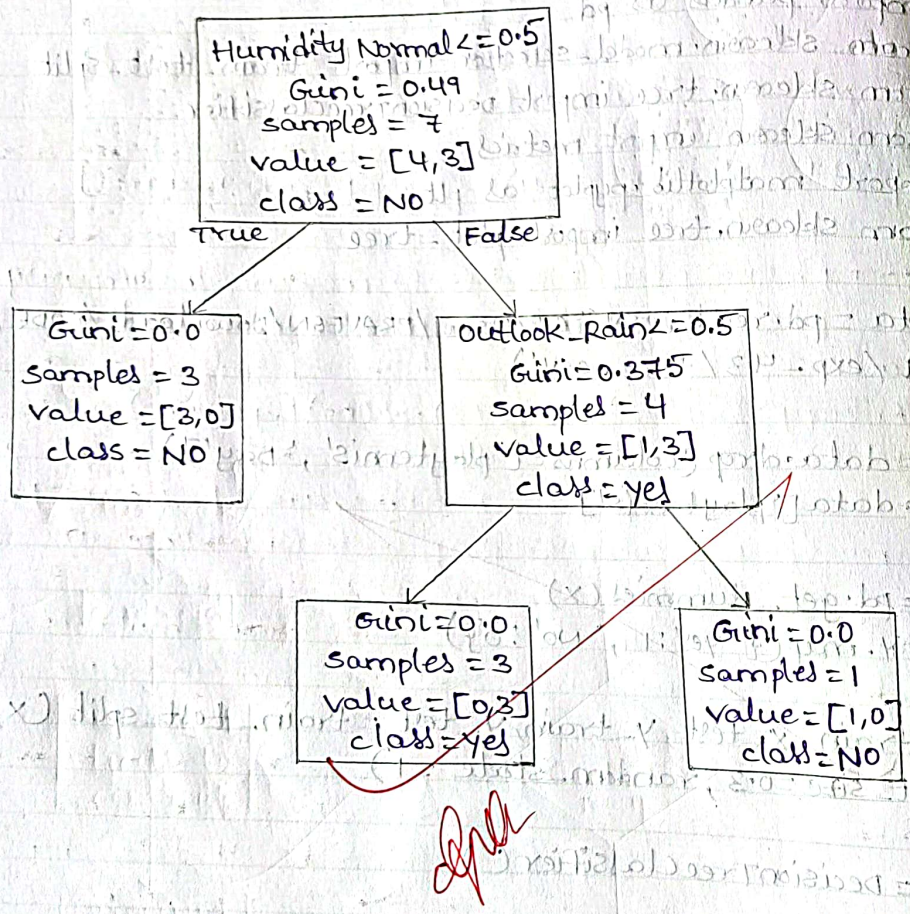
→ An attribute with the low Gini index should be preferred as compared to the high Gini index.

$$\text{Gini index} = 1 - \Sigma_j P_j^2$$

Program:

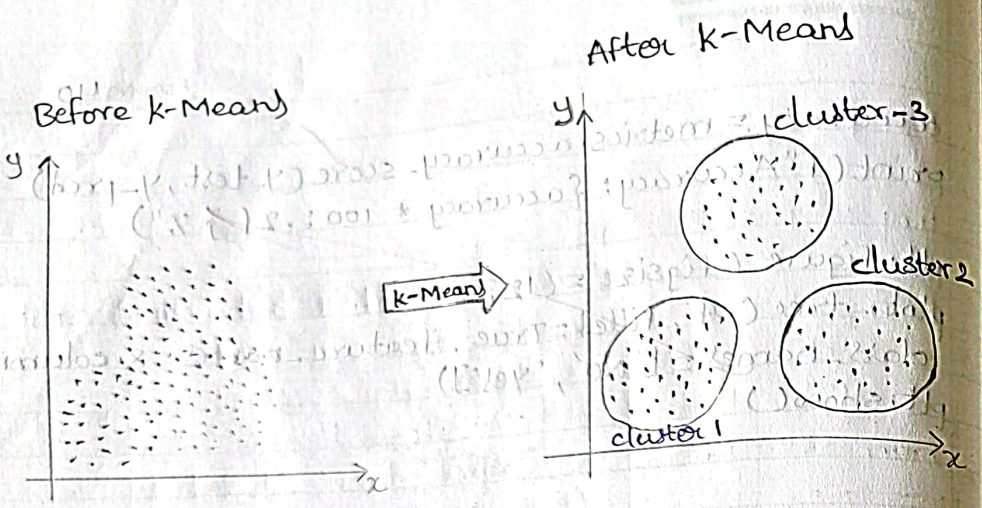
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

data = pd.read_csv('C:/Users/PRAVEEN/Downloads/CODE/
ML/exp-4.3/tree.csv')

x = data.drop(columns=['playtennis','Day'])
y = data['playtennis']

x = pd.get_dummies(x)
y = y.map({'Yes':1,'No':0})

x_train, x_test, y_train, y_test = train_test_split(x,y,
test_size=0.3, random_state=1)

clf = DecisionTreeClassifier().

clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
```

Accuracy = 90.00%

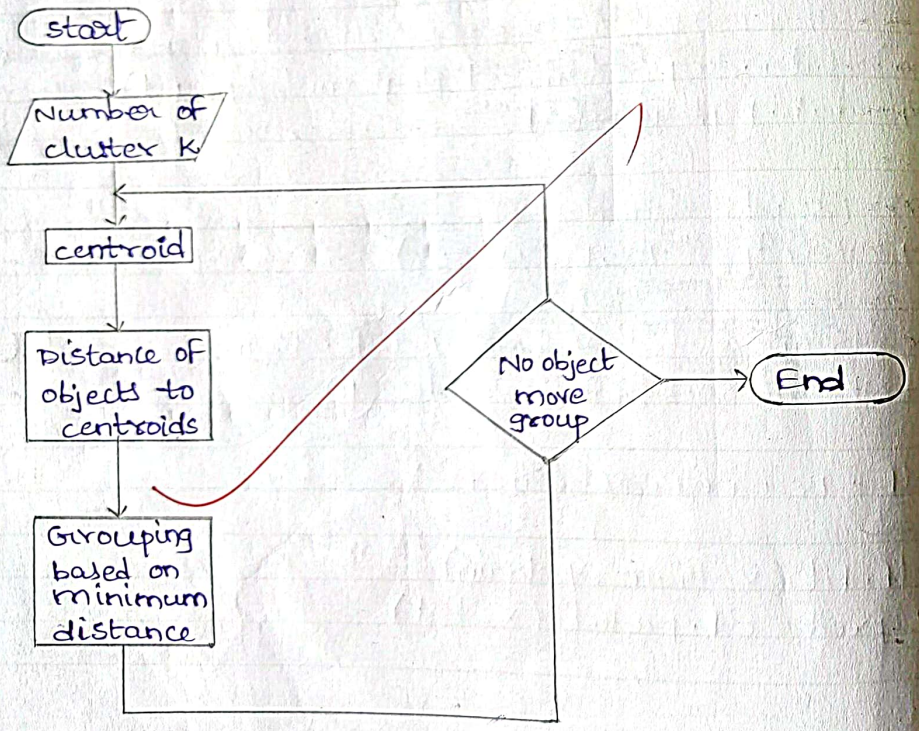


BVC GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY


```
accuracy = metrics.accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

plt.figure(figsize=(12,8))
plot_tree(clf, filled=True, feature_names=X.columns,
class_names=['NO', 'yes'])
plt.show()
```



K-Means Algorithm for clustering;



BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY

Experiment-8

Aim: Apply EM algorithm to cluster a Heart Disease Data set. Use the same data set for clustering using k-means algorithm. Compare the results of these two algorithms and comment on the quality of clustering. You can add python ML library classes / API in the program.

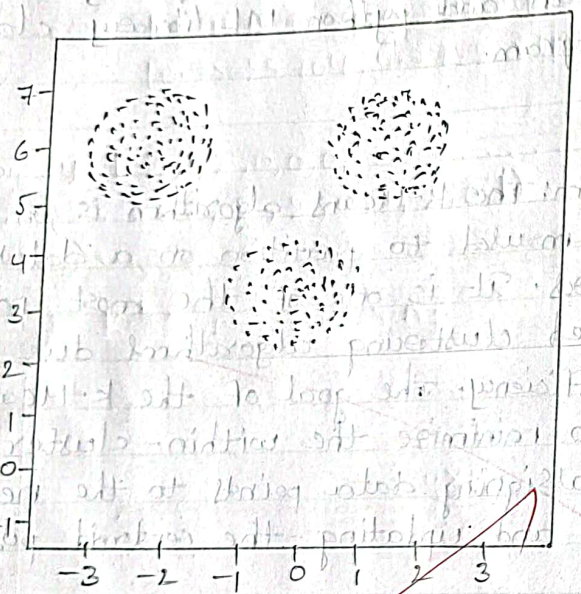
Description:

K-Means Algorithm: The K-Means algorithm is an iterative clustering algorithm used to partition on a dataset into K distinct clusters. It is one of the most popular and widely used clustering algorithms due to its simplicity and efficiency. The goal of the K-Means algorithm is to minimize the within-cluster variance by iteratively assigning data points to the nearest cluster centroid and updating the centroid positions.

Algorithm steps:

Step-1: Select the number k, to decide the no.of clusters.
Step-2: Select Random k points (or) centroids.
Step-3: Assign each data point to their closest centroid which will form the predefined k-clusters.
Step-4: Calculate the variance and place a new centroid of each cluster.
Step-5: Repeat the 3rd step which means reassign each data point to the new closest centroid of each cluster.

Generated Data



▼ KMeans
KMeans(n_clusters=3)

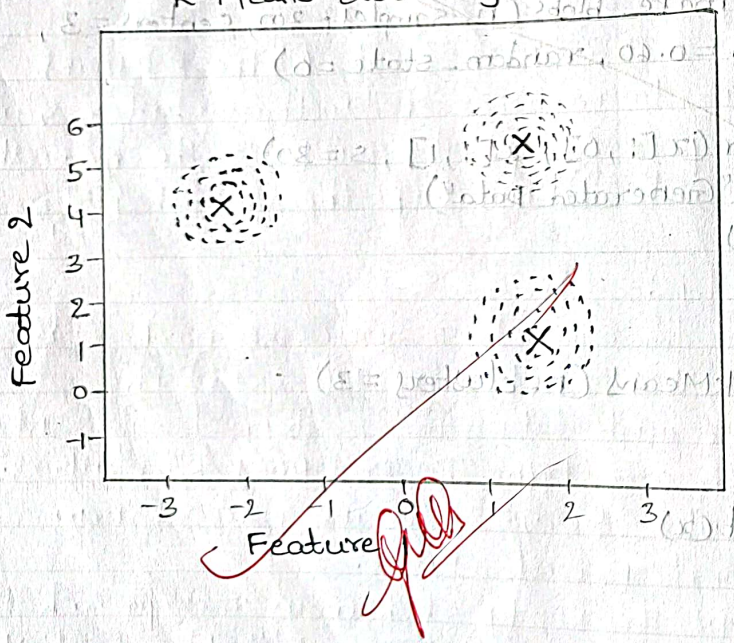
step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
step-7: The model is ready

program:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import kMeans

x,y_true = make_blobs(n_samples=300, centers=3,
cluster_std=0.60, random_state=0)

plt.scatter(x[:,0], x[:,1], s=30)
plt.title("Generated Data")
plt.show()


kmeans = kMeans(n_clusters=3)


kmeans.fit(x)
```

```
centers = kmeans.cluster_centers_
labels = kmeans.labels_

centers

labels

plt.scatter(x[:,0], x[:,1], c=labels, s=50, cmap='viridis')
plt.scatter(centers[:,0], centers[:,1], c='red', s=200, alpha=0.75, marker='x')
plt.title('kmeans clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

```
array([[-1.60811992, 2.58816585],
       [1.95159369, 0.83467497],
       [0.95625704, 4.37226546]])

array([2,1,2,0,0,0,...,2,2,0,1,2,0])
```



Experiment - 9

Aim: Assuming a set of documents that need to be classified, use the naive Bayesian classifier model to perform this task. Built-in Java classes/API can be used to write the program. calculate the accuracy, precision and recall for your data set.

Description:

Naive Bayes: Naive Bayes algorithm is a supervised learning algorithms, which is based on Bayes theorem and used for solving classification problems.

→ It is mainly used in text classification that includes a high dimensional training dataset.

→ It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

Example:

- Spam filtration
- sentimental analysis
- classifying articles

Naive: It is called Naive because it assumes that the occurance of a certain feature is independent of the occurance of other features.

Bayes: It is called Bayes because it depends on the principle of Bayes Theorem.

Bayes Theorem: Bayes Theorem is also known as Bayes Rule or Bayes law which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

→ The formula for Bayes theorem is given as

$$P(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$$

where, P(A) is prior probability
P(B) is Marginal probability
P(A/B) is posterior probability
P(B/A) is likelihood probability

Working of Naive Bayes classifier:

→ Convert the given dataset into frequency tables.
→ Generate likelihood table by finding the probabilities of given features.
→ Now, use Bayes theorem to calculate the posterior probability.

Program:

```
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

iris = sns.load_dataset('iris')
```

	Sepal_length	Sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
...	...	...	...	...
149	5.9	3.0	5.1	1.8

150 rows x 4 columns

```
0         setosa
1         setosa
...          ...
149    virginica
Name: species, Length:150, dtype: object
```

▼ GaussianNB
GaussianNB()

Accuracy: 100.00%
Classification Report:

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	9
virginica	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY


```
x = iris.drop(columns='species')
x
```

```
y = iris['species']
y
```

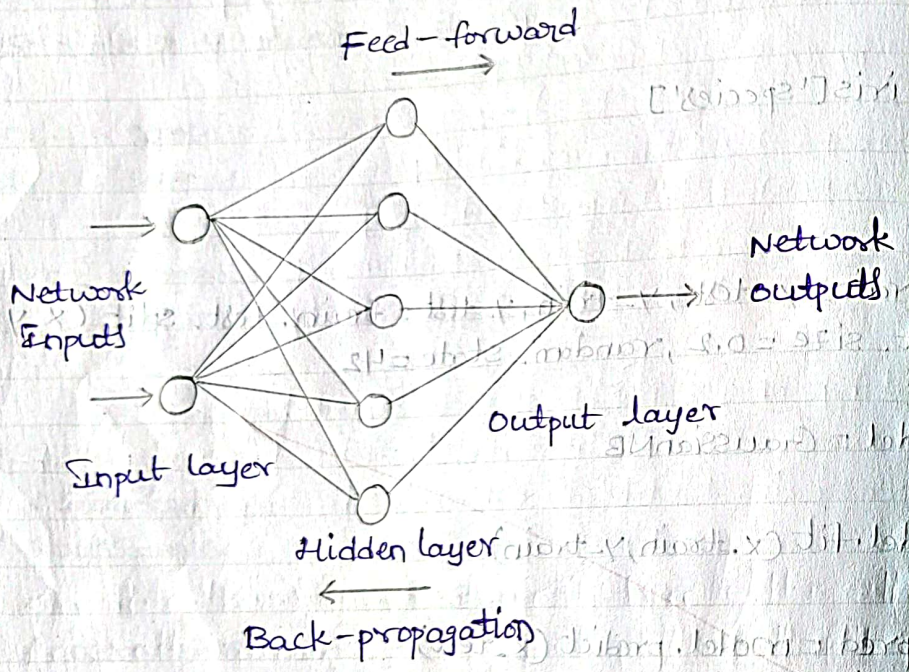
```
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42
```

```
model = GaussianNB
```

```
model.fit(x_train, y_train)
```

```
y_pred = model.predict(x_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
print('Classification Report:')
print(classification_report(y_test, y_pred))
```



BVC
GROUP OF INSTITUTIONS
ODALAREVU · AMALAPURAM · RAJAHMUNDRY

Experiment-10

Page No. 47

Aim: Build an Artificial Neural Network by implementing the Back propagation Algorithm and test the same using appropriate data sets.

Description:

Artificial Neural Network: (ANN)

ANN are algorithm based on brain function and are used to model complicated patterns and forecast issues. The ANN is a deep learning method comes from the concept of the human brain Biological Neural Networks. ANN algorithm accepts only numeric and structured data.

→ Convolutional Neural Networks (CNN) and Recursive Neural Networks (RNN) are used to accept unstructured and non-numeric data forms such as Image, Text and Speech.

→ ANNs are at the very core of deep learning being used in

* Google images
* Apple's Siri
* YouTube
* DeepMind's AlphaGo

There are 3 layers in the network Architecture.

* Input layer
* Hidden layer (More than one)
* Output layer

→ Hidden layer accelerates and improves the efficiency of the network by recognizing just the most important information from the inputs and discarding the redundant information.
→ The activation function is important for two reasons
→ The model captures the presence of non linear relationships between the inputs.
→ It contributes to the conversion of the input into a more usable output.

program:

```
import tensorflow as tf

from tensorflow import keras

dataframe = tf.keras.datasets.mnist

dataframe

(x_train,y_train), (x_test ,y_test) = dataframe.load_data()

x_train.shape

x_test.shape
```

```
<module 'keras.api.datasets.mnist' from 'C:\\Users\\
praveen\\anaconda3\\Lib\\site-packages\\keras\\api\\
datasets\\mnist\\__init__.py' >
```

(60000, 28, 28)

(10000, 28, 28)

```
x_train, x_test = x_train/255, x_test/255
x_train.shape
y_train.shape
x_test.shape
y_test.shape

model = tf.keras.models.Sequential(
        [tf.keras.layers.Flatten(input_shape=(28,28)),
         tf.keras.layers.Dense(128, activation='relu'),
         tf.keras.layers.Dropout(0.2),
         tf.keras.layers.Dense(10, activation='softmax')
        ])

model

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
```

(60000, 28, 28)

(60000)

(10000, 28, 28)

(10000)

<Sequential name=sequential_1, built=True>

Epoch 1/5
1875/1875 ____ 135 6ms/step - accuracy: 0.8601 - loss: 0.48

Epoch 2/5
1875/1875 ____ 205 5ms/step - accuracy: 0.9548 - loss: 0.152

Epoch 3/5
1875/1875 ——— 11s 6ms/step - accuracy: 0.9667 - loss: 0.1091
Epoch 4/5
1875/1875 ——— 10s 5ms/step - accuracy: 0.9745 - loss: 0.0837
Epoch 5/5
1875/1875 ——— 10s 5ms/step - accuracy: 0.9770 - loss: 0.0741

313/313 ——— 0s 881us/step - accuracy: 0.9744 - loss: 0.0864
[0.07309079915285 11, 0.9779002822 87]

model.evaluate(x_test, y_test)