Batch: H2-1 **Roll No:-** 16010122151

Experiment No. 6

Title: Classification using decision tree algorithm

Aim: To build a decision tree classifier with C4.5 algorithm using R libraries

Expected Outcome of Experiment:
CO3 : Understand the basic concept and techniques of Machine Learning regression and classification

Books/ Journals/ Websites referred:

1. Data Mining Concepts and Techniques Jiawei Han, Michelin Kamber, Jian Pie, 3rd edition

---

Step 1: Install and Load the packages

Step 2: Create cross validation folds

Step 3: Create the model

Step 4: Interpret the model results

Step 5: Visualize the tree

Step 6: Interpret the classification rules from the tree

Step 7: Predict the class for some random user input unseen data

Procedure for Implementation in lab :

1. Select a dataset suitable for classification from UCI data repository or Kaggle. Example:

   Titanic dataset (https://www.kaggle.com/datasets/yasserh/titanic-dataset) , where "Survived" can be used as a class label.

2. **Students should provide the following details of the dataset:**
   a. Title: Iris Flower Dataset
   b. Source:
   c. Number of instances:
   d. Number of attributes:
   e. Attribute information:

3. Handle the missing values appropriately
4. Create the decision tree, interpret the tree to extract all the rules and perform prediction on unseen data.

Code:

```
# Loading the required packages
install.packages("caret")
install.packages('rpart.plot')
library(caret)
library(rpart.plot)

# Loading the dataset
mydata <- iris
table(iris$Species)


# Summary of data
summary(mydata)


# Structure of data
str(mydata)


# Handling missing values (if any)
# Check for missing values
missing_values <- sum(is.na(mydata))
if (missing_values > 0) {
  # Handle missing values by imputation or removal
  mydata <- na.omit(mydata)  # Example: Remove rows with missing values
}
```

```
# Splitting the dataset into train and test sets (70% train, 30% test)
set.seed(3033)
intrain <- createDataPartition(y = mydata$Species, p = 0.7, list = FALSE)
training <- mydata[intrain,]
testing <- mydata[-intrain,]
dim(training); dim(testing)


# Classification by Information Gain
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(3333)
dtree_fit_info <- train(Species ~ ., data = training, method = "rpart",
                        parms = list(split = "information"),
                        trControl = trctrl,
                        tuneLength = 10)
prp(dtree_fit_info$finalModel, box.palette = "Reds", tweak = 1.2)


# Predictions of unseen data and model performance evaluation
test_pred_info <- predict(dtree_fit_info, newdata = testing)
confusionMatrix(test_pred_info, testing$Species)


# Extracting classification rules from the decision tree
tree_rules <- rpart.rules(dtree_fit_info$finalModel)


# Print the classification rules
print(tree_rules)
```

Dataset Details:

a. Title: Iris Flower Dataset

b. Source: The Iris Flower Dataset is a classic dataset used in machine learning and statistics. It was introduced by the British statistician and biologist Ronald Fisher in his 1936 paper "The use of multiple measurements in taxonomic problems" as an example of linear discriminant analysis.

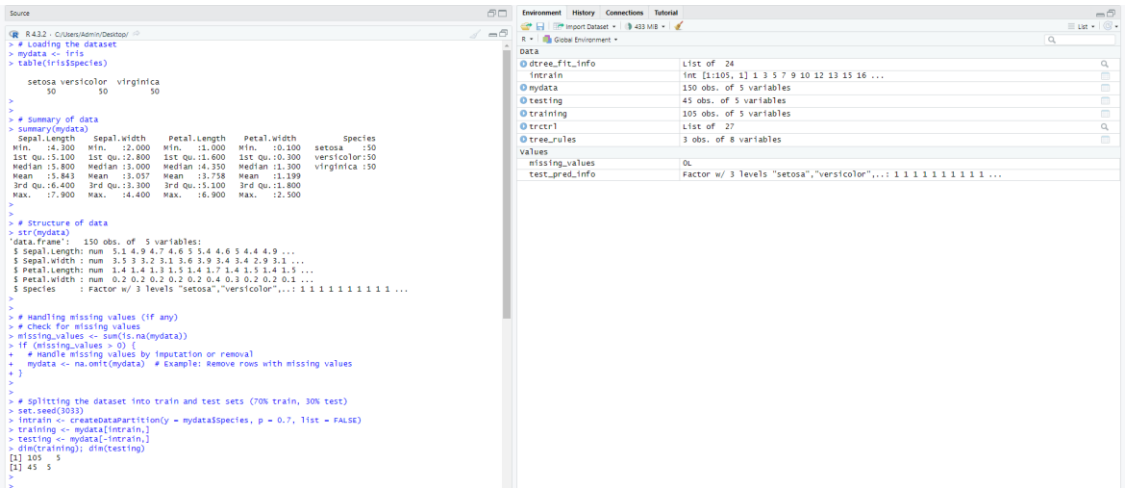
c. Number of instances: The dataset contains 150 instances.

d. Number of attributes: There are 4 attributes in the dataset.

e. Attribute information:
- Sepal Length (in cm)
- Sepal Width (in cm)

- Petal Length (in cm)
- Petal Width (in cm)
- Species: Class label indicating the species of iris plant (setosa, versicolor, or virginica)

Implementation:

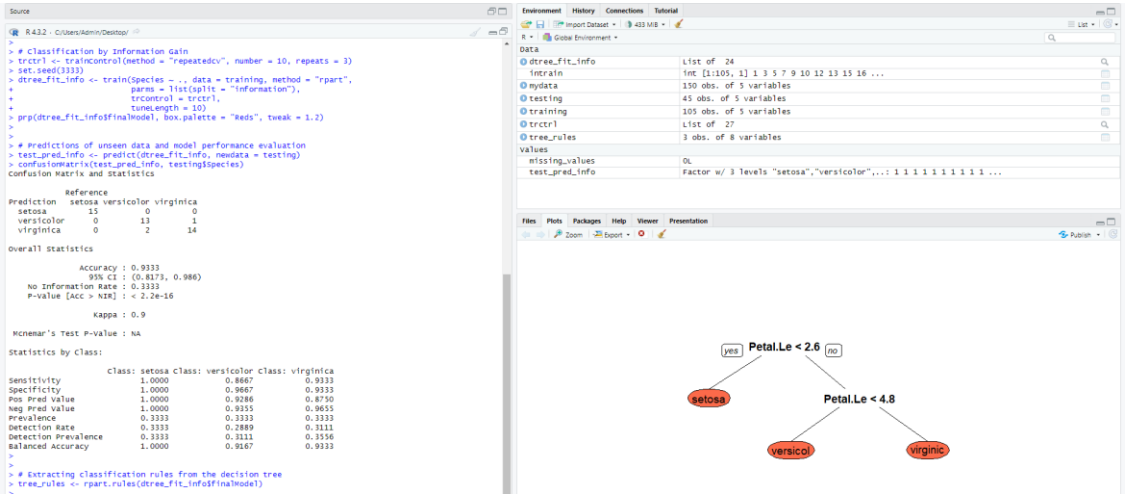


```
> # Loading the dataset
> mydata <- iris
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50

>
>
> # Summary of data
> summary(mydata)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
>
>
> # Structure of data
> str(mydata)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
>
>
> # Handling missing values (if any)
> # Check for missing values
> missing_values <- sum(is.na(mydata))
> if (missing_values > 0) {
+   # Handle missing values by imputation or removal
+   mydata <- na.omit(mydata)  # Example: Remove rows with missing values
+ }
>
>
> # Splitting the dataset into train and test sets (70% train, 30% test)
> set.seed(3033)
> intrain <- createDataPartition(y = mydata$Species, p = 0.7, list = FALSE)
> training <- mydata[intrain,]
> testing <- mydata[-intrain,]
> dim(training); dim(testing)
[1] 105   5
[1] 45  5
>
>
```

Environment	
Data	
dtree_fit_info	List of 24
intrain	int [1:105, 1] 1 3 5 7 9 10 12 13 15 16 ...
mydata	150 obs. of 5 variables
testing	45 obs. of 5 variables
training	105 obs. of 5 variables
trctrl	List of 27
tree_rules	3 obs. of 8 variables
Values	
missing_values	0L
test_pred_info	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...



```
>
> # Classification by Information Gain
> trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
> set.seed(3333)
> dtree_fit_info <- train(Species ~ ., data = training, method = "rpart",
+                         parms = list(split = "information"),
+                         trControl = trctrl,
+                         tuneLength = 10)
> prp(dtree_fit_info$finalModel, box.palette = "Reds", tweak = 1.2)
>
>
> # Predictions of unseen data and model performance evaluation
> test_pred_info <- predict(dtree_fit_info, newdata = testing)
> confusionMatrix(test_pred_info, testing$Species)
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         15          0         0
  versicolor      0         13         1
  virginica       0          2        14

Overall Statistics

               Accuracy : 0.9333
                 95% CI : (0.8173, 0.986)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.8667           0.9333
Specificity                 1.0000            0.9667           0.9333
Pos Pred Value              1.0000            0.9286           0.8750
Neg Pred Value              1.0000            0.9355           0.9655
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.2889           0.3111
Detection Prevalence        0.3333            0.3111           0.3556
Balanced Accuracy           1.0000            0.9167           0.9333
>
>
> # Extracting classification rules from the decision tree
> tree_rules <- rpart.rules(dtree_fit_info$finalModel)
>
```

Environment	
Data	
dtree_fit_info	List of 24
intrain	int [1:105, 1] 1 3 5 7 9 10 12 13 15 16 ...
mydata	150 obs. of 5 variables
testing	45 obs. of 5 variables
training	105 obs. of 5 variables
trctrl	List of 27
tree_rules	3 obs. of 8 variables
Values	
missing_values	0L
test_pred_info	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...


```
> # Print the classification rules
> print(tree_rules)
   .outcome  seto vers virg
     setosa [1.00  .00  .00] when Petal.Length <  2.6
 versicolor [ .00 1.00  .00] when Petal.Length is 2.6 to 4.8
  virginica [ .00  .10  .90] when Petal.Length >=        4.8
>
```

Conclusion:
Hence, we learned classification using decision tree algorithm.

Post lab Questions:

Q1. Explain in detail with mathematical formulae wherever necessary, the following decision tree algorithms with emphasis on the criteria they use, the type of data that they're most suitable for and their pros and cons:

- CART
- C4.5
- C5.0
- CHAID
- ID3

ANSWER:

Decision tree algorithms are popular machine learning methods used for both classification and regression tasks. They construct a tree-like structure where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a numerical value. Here's an explanation of the mentioned decision tree algorithms:

- CART (Classification and Regression Trees):

Criteria: CART uses binary recursive partitioning to split the data into homogenous subsets. The splits are determined based on impurity measures such as Gini impurity or entropy.

Type of Data: CART can handle both numerical and categorical data. It is versatile and can be applied to various types of datasets.

Pros:

- Simple to understand and interpret.
- Can handle both classification and regression tasks.
- Handles missing values well.

Cons:

- Prone to overfitting, especially with large and complex trees.
- May not capture complex relationships in the data.
- Greedy nature may lead to locally optimal solutions.

- C4.5:

Criteria: C4.5, developed by Ross Quinlan, uses information gain or gain ratio to determine the best attribute to split the data at each node. Information gain measures the reduction in entropy, while gain ratio normalizes the information gain by the intrinsic information of the attribute.

Type of Data: C4.5 can handle both discrete and continuous attributes. It works well with datasets that have categorical attributes.

Pros:

- Handles both discrete and continuous attributes.
- Prunes the tree to reduce overfitting.
- Deals with missing attribute values through surrogate splits.

Cons:

- Prone to bias towards attributes with many values.
- Less efficient with large datasets due to its computational complexity.

- C5.0:

Criteria: C5.0 is an improved version of C4.5, addressing some of its limitations. It uses the same criteria for splitting as C4.5 but incorporates various optimizations for faster and more accurate tree generation.

Type of Data: Similar to C4.5, C5.0 can handle both discrete and continuous attributes and works well with categorical datasets.

Pros:

- Faster and more memory-efficient than C4.5.
- Incorporates boosting techniques to improve accuracy.
- Handles missing values and noisy data effectively.

Cons:

- May produce large and complex trees, leading to reduced interpretability.
- Can still be computationally intensive, especially with large datasets.

- CHAID (Chi-squared Automatic Interaction Detector):

Criteria: CHAID is primarily used for classification and employs chi-squared tests to determine the best attribute splits. It identifies significant associations between variables to partition the data.

Type of Data: CHAID works well with categorical and ordinal data. It is commonly used in marketing and social sciences for segmentation analysis.

Pros:

- Suitable for categorical and ordinal data.
- Provides easy-to-interpret results in the form of a tree.
- Handles large datasets efficiently.

Cons:

- Limited to categorical and ordinal predictors.
- May not capture complex interactions between variables.
- Less flexible compared to other algorithms like C4.5 and CART.

- ID3 (Iterative Dichotomiser 3):

Criteria: ID3 uses entropy and information gain to select the best attribute for splitting the data. It maximizes the information gain at each node to create a tree that best classifies the data.

Type of Data: ID3 is suitable for categorical data and works well with datasets that have a limited number of discrete attributes.

Pros:

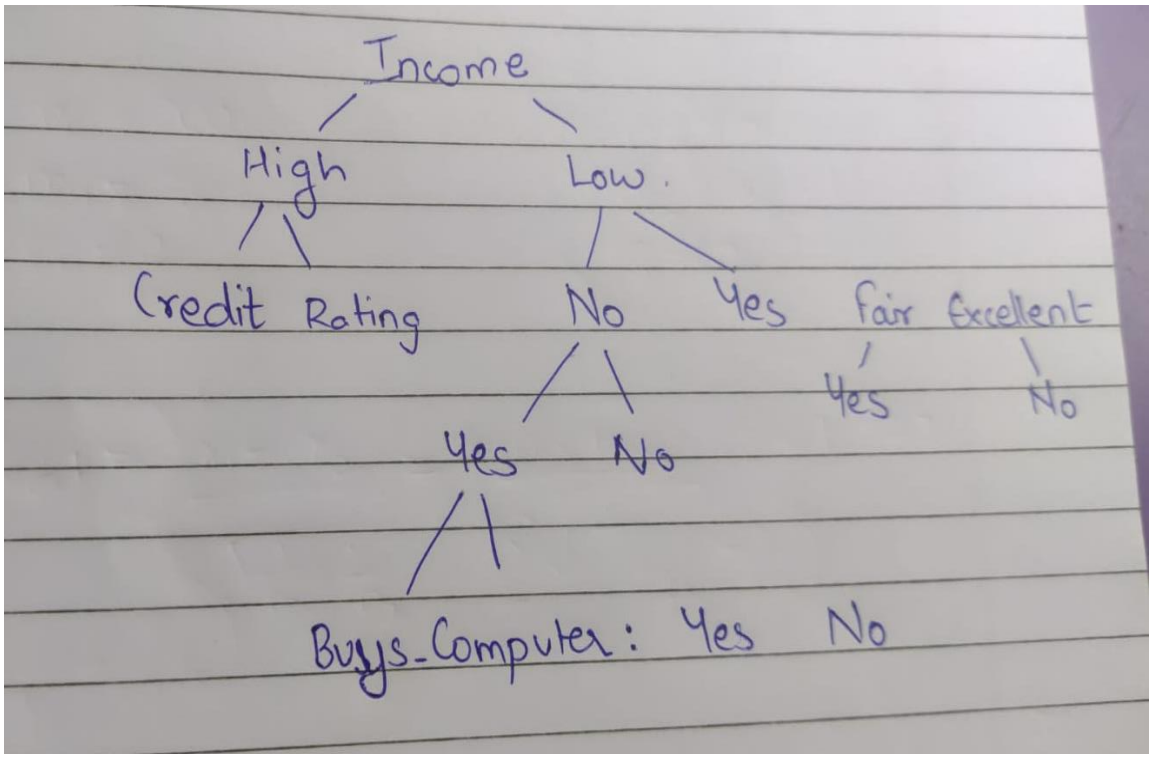
- Simple and easy to implement.
- Generates small and interpretable trees.
- Handles noisy data and missing values using majority voting.

Cons:

- Limited to categorical attributes.
- Prone to overfitting, especially with noisy data.
- Does not support pruning, which can lead to complex trees.

Q2. Apply the C4.5 decision tree algorithm on the following dataset and build the decision tree. Extract all the classification rules from the resulting tree. Students are supposed to solve this in their notebook by hand and then upload the scan/photograph of the same here.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



- If Income is High:

- If Credit Rating is Excellent, then Buys_Computer = Yes.
- If Credit Rating is Fair, then Buys_Computer = No.
- If Credit Rating is No or missing, then C4.5 decision tree handles the value.

- If Income is Low:

- If Credit Rating is Yes, then Buys_Computer = Yes.
- If Credit Rating is Fair or No (or missing), then Buys_Computer = No.