

CSCI 5408

Assignment 6: Classification DM and Application

Submission date: 11th April 2017

Student ID: B00741912

Name: Yamuna Jayabalan

Student ID: B00762641

Name: Bharat Jain

Division of Task:

Yamuna Jayabalan – Worked on R Script with the relevant report & README

Bharat Jain – Worked on Weka Tool with relevant report & README

Objective:

- To gain an in-depth understanding of classification methods and applications.
- To perform classifications using two tools Weka and RStudio.
- To learn the ways to perform classification using Decision Tree and Naïve Bayes algorithms.
- To observe and compare the differences between the two algorithm implementations based on performance, flexibility and user interface.

Introduction:

Introduction on software tools Weka, RStudio and algorithms Decision Tree and Naïve Bayes algorithms.

WEKA Tool:

Waikato Environment for Knowledge Analysis (Weka) is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka contains a collection of visualization tools and algorithms for data analysis and predictive modelling, together with graphical user interfaces for easy access to these functions. [2]

R Programming Language:

R is an open source programming language and software environment for statistical computing and graphics that is supported by the R Foundation for Statistical Computing. R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. [3]

Decision Tree Algorithm:

A **decision tree** is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning. [4]

Naïve Bayes Algorithm:

In machine learning, **naive Bayes classifiers** are a family of simple classifiers based on applying Bayes with strong (naive) independence assumptions between the features.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a feature is independent of the value of any other feature, given the class variable. [5]

1. Application Scenario and Data Set:

1.1 Selected Dataset:

The Data set which we have selected is **Human Resources Analytics** [1]. Let's have a sneak peek into the data set which we have selected and discuss about the attributes.

Database Instance:

Below is the database instance of dataset under discussion. There are 15000 records in this dataset. A brief description of attributes is listed below.

time_spend_company ⓘ	Work_accident ⓘ	left ⓘ	promotion_last_5years ⓘ	sales ⓘ	salary ⓘ
3	0	1	0	sales	low
6	0	1	0	sales	medium
4	0	1	0	sales	medium
5	0	1	0	sales	low
3	0	1	0	sales	low
3	0	1	0	sales	low
4	0	1	0	sales	low
5	0	1	0	sales	low
5	0	1	0	sales	low
3	0	1	0	sales	low
3	0	1	0	sales	low
4	0	1	0	sales	low

Below is the list of all columns present in the dataset.

- **satisfaction_level** - Level of satisfaction (0-1)
- **last_evaluation** - Evaluation of employee performance (0-1)
- **number_project** - Number of projects completed while at work
- **average_monthly_hours** - Average monthly hours at workplace
- **time_spend_company** - Number of years spent in the company
- **Work_accident** - Whether the employee had a workplace accident
- **promotion_last_5years** - Whether the employee was promoted in the last five years
- **sales** - Department in which they work for
- **salary** - Relative level of salary (low med high)
- **Left** - Whether the employee left the workplace or not (Yes or No)

1.2 Application Scenario:

Based on the selected dataset, an interesting application scenario is devised for this task which is described below.

Why outstanding and most experienced employees are leaving the company? Based on certain attributes predict if an employee is part of a company or left.

Similarly, there can be many other application scenarios for any given dataset. But for this task the focus is on the above application scenario.

As the concept of decision tree is discussed above, when the decision tree is generated, target class attribute will be the leaf node by default. Using weka tool a specific target attribute can be selected.

2. Data processing and target dataset:

There are number of tools available for data processing & cleaning such as SSIS & Weka. But for the selected dataset there is no requirement for dataset cleaning.

- We have done the data processing where we have moved our target attribute to the last column using Weka. Weka provides the option to select which column is target attribute.
- Also, as there are numerical values in our dataset which is not accepted by WEKA tool. So, we have also applied the filter to make it work. The name of the filter is **NumericToNominal**.
- The Class to be predicted is taken as "left" which says if the employee has left the company or not (Yes/No). The value of this field was initially 1/0 but it is modified as Yes/No for ease of understanding.

So, our target data set is almost the same as source data set except that the target attribute is moved to the last column and the values are relatively modified.

3. Classification Data Mining Solution & Experimental Result:

In machine learning and statistics, **classification** is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, based on a training set of data containing observations (or instances) whose category membership is known. An example would be assigning a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition [6].

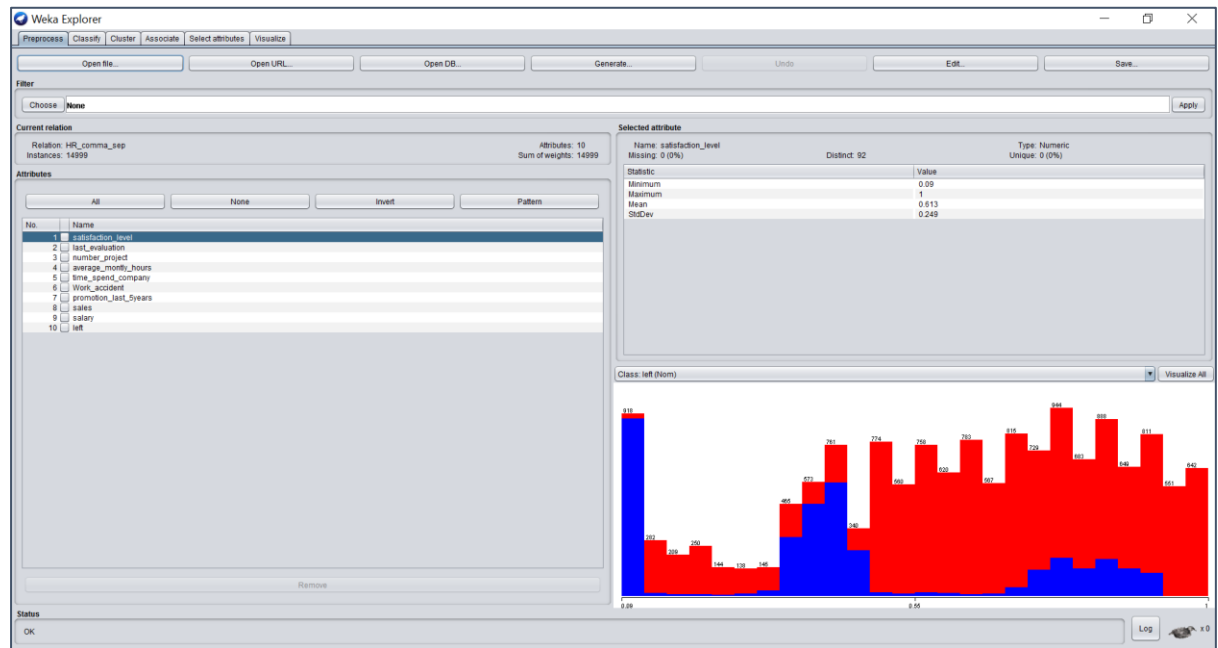
In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance [6].

An algorithm that implements classification, especially in a concrete implementation, is known as a **classifier**. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category [6]. We will be using J48 DT and Naïve Bayes in Weka. Along with that we will be using C5.0 DT and Naïve Bayes in RStudio.

3.1 Classification Data Mining using WEKA and Experiment Results

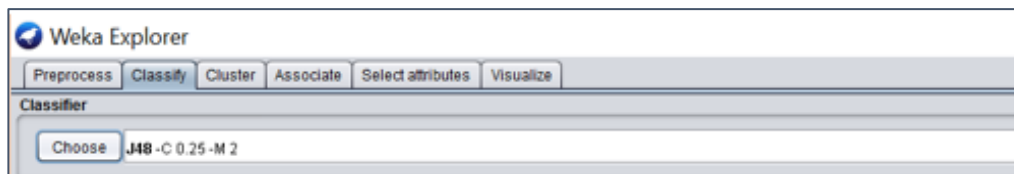
3.1.1 Decision Tree: Based on business scenario we have selected target attribute as Left.

Step 1: Import the data set in WEKA.

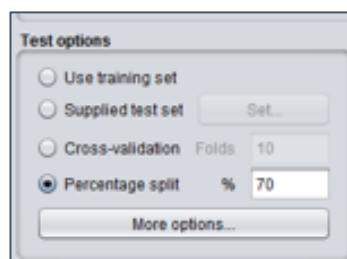


In the right bottom, we can also see that weka has accepted the data set and the data visualization is done. This is just the visualization based on “**Left Class**”. Similarly, visualization can be done for all other attributes based on the target attribute.

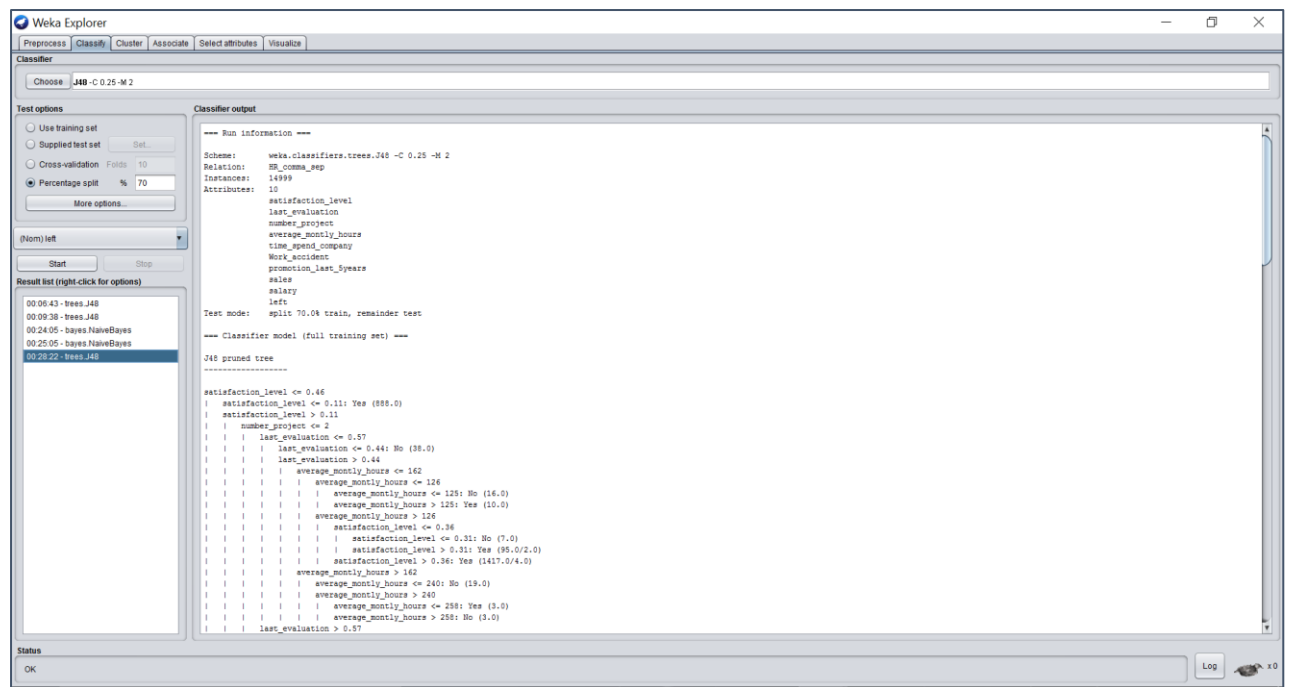
Step 2. Go to the Classify Tab and select Classifier as **J48**. It will look like below.



Step 3. Select percentage split as 70%. It will look like below.

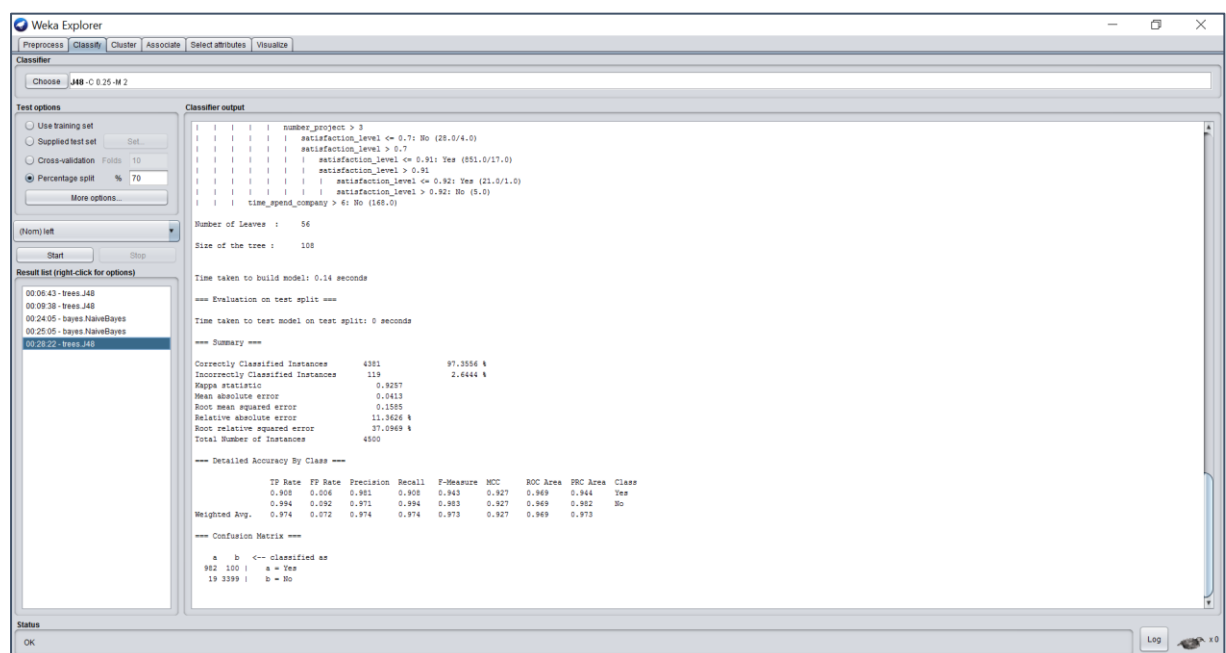


Step 4. Click on start to perform the J48 algorithm.



It can be seen in right panel that the algorithm execution succeeded and the J48 pruned Tree is displayed. The result buffer is stored in the file named **DToutputWeka**.

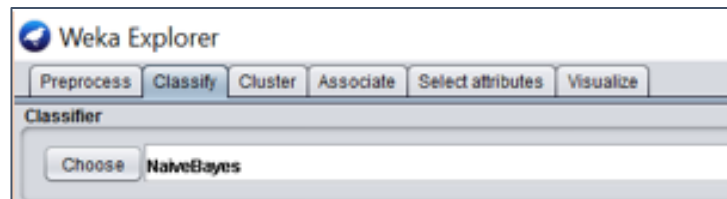
J48 Accuracy Metrics and Confusion Matrix:



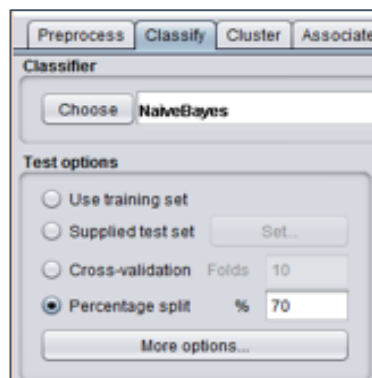
Above is the representation of metrics, Accuracy and Confusion Matrix based on the selected target attribute.

3.1.2 Naïve Bayes: Based on business scenario we have selected target attribute as” Left”.

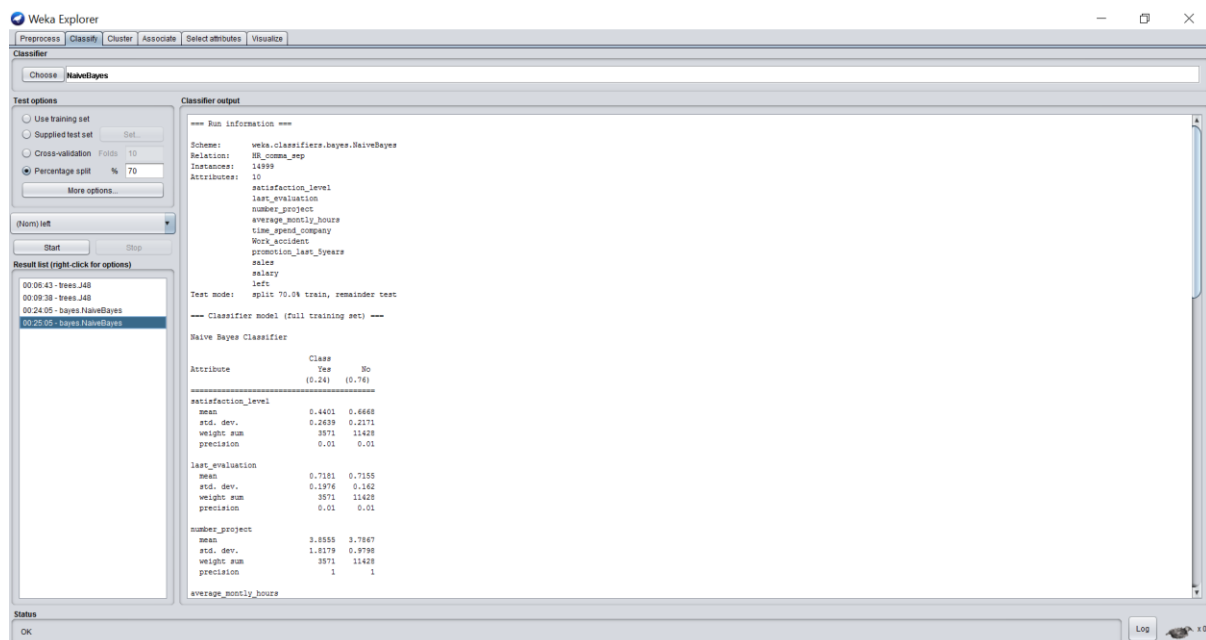
Step 1: Select the classifier as NaiveBayes.



Step 2: Select the percentage split as 70%.

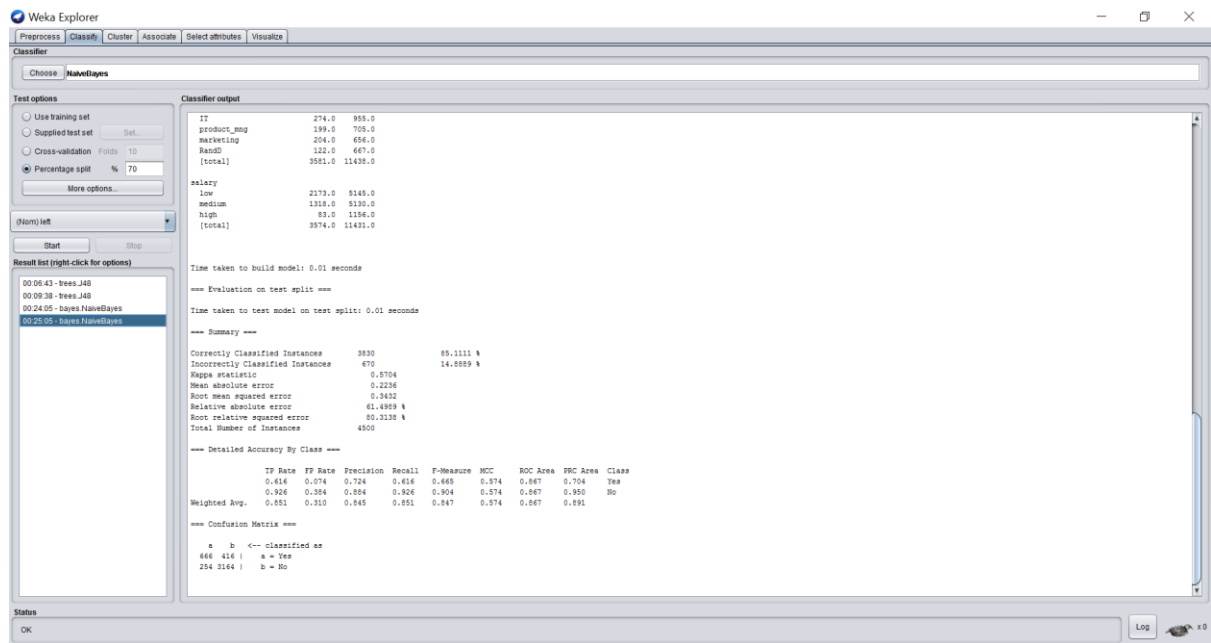


Step 3: Select target attribute as “Left” based on the business scenario. Click on start.



It can be seen in right panel that the algorithm execution succeeded and the Naïve Bayes Classifiers are displayed. The result buffer is stored in the file named **NBoutputWeka**.

Naïve Bayes Accuracy Metrics and Confusion Matrix:

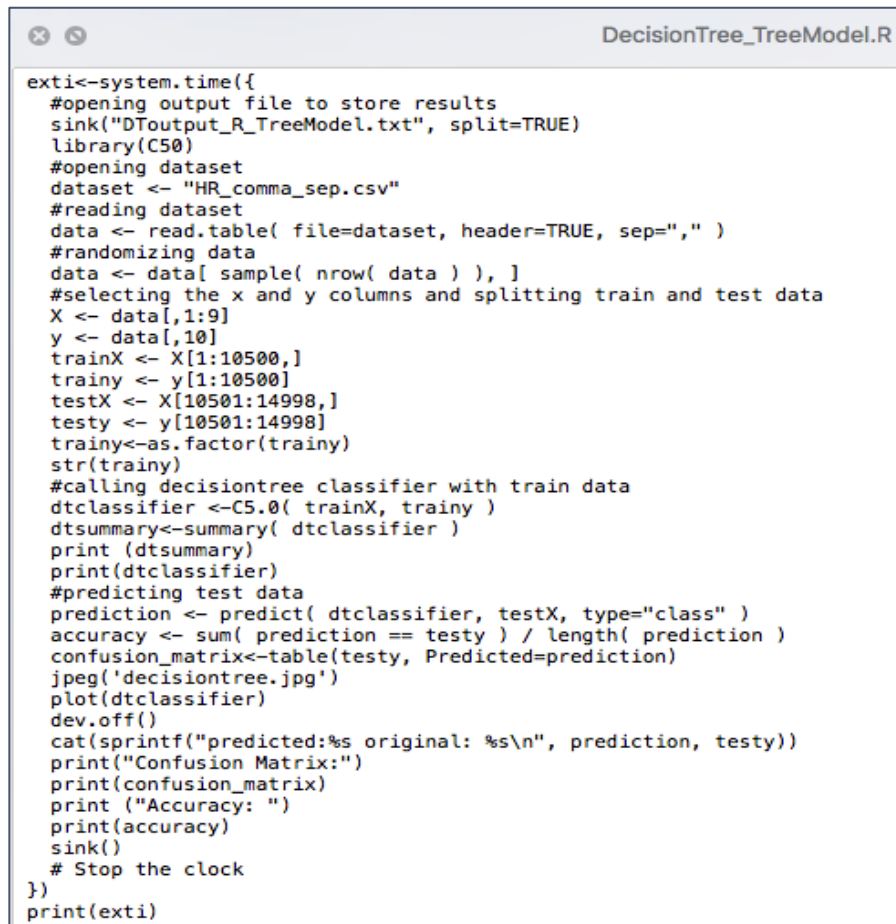


Above is the representation of metrics, Accuracy and Confusion Matrix based on the selected target attribute.

3.2 Classification Data Mining using R:

3.2.1 Tree based Decision Tree Model

Below is the R script to generate the Decision Tree.



```
exti<-system.time({
  #opening output file to store results
  sink("DToutput_R_TreeModel.txt", split=TRUE)
  library(C50)
  #opening dataset
  dataset <- "HR_comma_sep.csv"
  #reading dataset
  data <- read.table( file=dataset, header=TRUE, sep="," )
  #randomizing data
  data <- data[ sample( nrow( data ) ), ]
  #selecting the x and y columns and splitting train and test data
  X <- data[,1:9]
  y <- data[,10]
  trainX <- X[1:10500,]
  trainy <- y[1:10500]
  testX <- X[10501:14998,]
  testy <- y[10501:14998]
  trainy<-as.factor(trainy)
  str(trainy)
  #calling decisiontree classifier with train data
  dtclassifier <-C5.0( trainX, trainy )
  dtsummary<-summary( dtclassifier )
  print( dtsummary)
  print(dtclassifier)
  #predicting test data
  prediction <- predict( dtclassifier, testX, type="class" )
  accuracy <- sum( prediction == testy ) / length( prediction )
  confusion_matrix<-table(testy, Predicted=prediction)
  jpeg('decisiontree.jpg')
  plot(dtclassifier)
  dev.off()
  cat(sprintf("predicted:%s original: %s\n", prediction, testy))
  print("Confusion Matrix:")
  print(confusion_matrix)
  print ("Accuracy: ")
  print(accuracy)
  sink()
  # Stop the clock
})
print(exti)
```

Step by step detail of the above R Script:

- Opening output file to store the result of each step (DToutput_R_TreeModel.txt)
- Importing dataset
- Randomizing data
- Selecting target column Y and splitting dataset into the train(70%) and test(30%).
- Calling DecisionTree classifier with training dataset to train the model
- Predicting test dataset with the trained model
- Printing accuracy and confusion matrix.

Output is stored in the file **DToutput_R_TreeModel.txt**. In next section, output generated by the Decision Tree - Rule based model is described.

Output file

```

Factor w/ 2 levels "No","Yes": 2 2 1 2 1 2 1 1 1 2 ...

Call:
C5.0.default(x = trainX, y = trainy)

C5.0 [Release 2.07 GPL Edition]          Mon Apr 10 20:00:50 2017
-----

Class specified by attribute 'outcome'

Read 10500 cases (10 attributes) from undefined.data

Decision tree:

average_monthly_hours > 287: Yes (230)
average_monthly_hours <= 287:
: ...number_project <= 2:
:   : ...satisfaction_level > 0.46: No (445/22)
:   :   : satisfaction_level <= 0.46:
:   :       : ...last_evaluation > 0.57: No (82/3)
:   :       :   : last_evaluation <= 0.57:
:   :       :       : ...last_evaluation <= 0.44: No (30)
:   :       :       :   : last_evaluation > 0.44:
:   :       :       :       : ...average_monthly_hours <= 160:
:   :       :       :           : ...satisfaction_level <= 0.36:
:   :       :       :               : ...satisfaction_level <= 0.35: No (15/4)
:   :       :       :               :   : satisfaction_level > 0.35: Yes (67/1)
:   :       :       :               :       : satisfaction_level > 0.36:
:   :       :       :               :           : ...average_monthly_hours > 127: Yes (947/3)
:   :       :       :               :               : average_monthly_hours <= 127:
:   :       :       :               :                   : ...average_monthly_hours <= 125: No (6)
:   :       :       :               :                   :   : average_monthly_hours > 125: Yes (48)
:   :       :       :               :                       : average_monthly_hours > 160:
:   :       :       :                   : ...average_monthly_hours <= 162: Yes (10)
:   :       :       :                       :   : average_monthly_hours > 162:
:   :       :       :                           : ...average_monthly_hours <= 240: No (14)
:   :       :       :                           :   : average_monthly_hours > 240:
:   :       :       :                               : ...average_monthly_hours <= 258: Yes (3)
:   :       :       :                               :   : average_monthly_hours > 258: No (3)

```

Output Execution Screenshot

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for loading data and performing a classification task.


```

1 #xtici<-system.time({
2   #opening output file to store results
3   sink("D:\\output_R_TreeModel.txt", split=TRUE)
4   library(C50)
5   #opening dataset
6   dataset <- "HR_comma_sep.csv"
7   #reading dataset
8   data <- read.table(file=dataset, header=TRUE, sep=",")
9   #randomizing data
10  #}

```
- Console:** Shows the output of the code execution, including a confusion matrix and system time.


```

11 Predicted
testy No Yes
      No 3429 13
      Yes 89 967
[1] "Accuracy: "
[1] 0.9773233
user system elapsed
1.86 0.06 1.95

```
- Environment:** Lists the objects in the workspace.
 - `accuracy`: 0.977323254779902
 - `confusion_matrix`: 'table' int [1:2, 1:2] 3429 89 13 967
 - `dataset`: "HR_comma_sep.csv"
 - `dtclassifier`: List of 16
 - `dtsummary`: List of 2
 - `extci`: Class 'proc_time' Named num [1:5] 1.86 0.06 1.95 NA NA
 - `prediction`: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 1 ...
 - `testy`: Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 1 1 ...
 - `trainy`: Factor w/ 2 levels "No","Yes": 2 2 1 2 1 1 1 1 2 ...
 - `y`: Factor w/ 2 levels "No","Yes": 2 2 1 2 1 2 1 1 1 2 ...
- Files:** Shows the project files, including `NaiveBayes.R`, `Decision_TreeModel.R`, and `Decision_Tree_RuleModel.R`.

The above figure demonstrates the output generated by the R Script for the Decision tree.

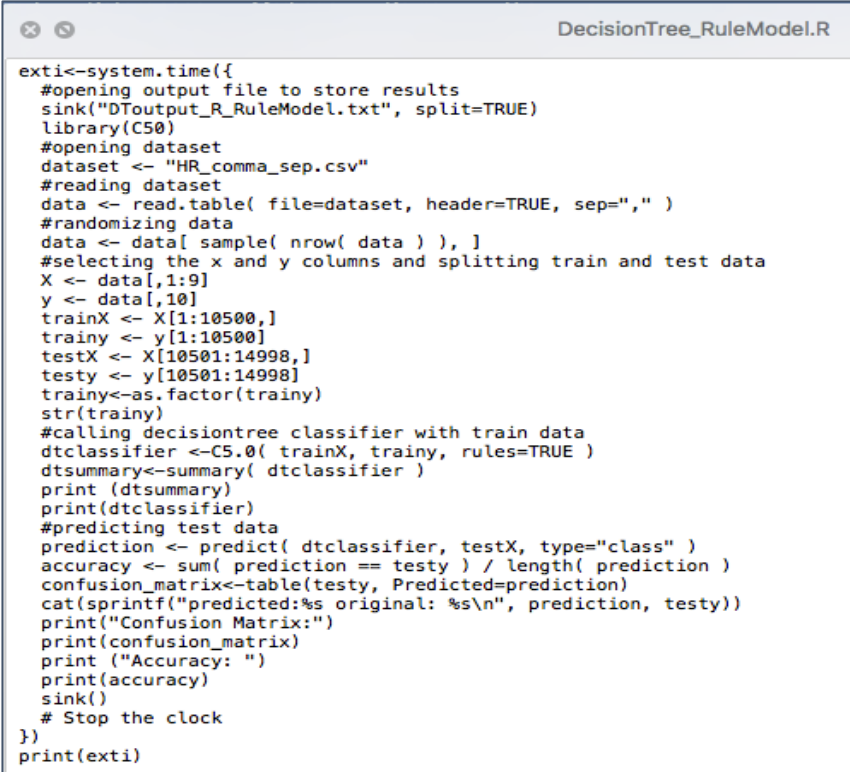
Understanding Decision Tree Model

```
-> number_project
    -> satisfaction_level
        -> last_evaluation
            -> average_monthly_hours
                -> Left
```

Basically, a **decision tree** is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm.

3.2.2 Rule based Decision Tree Model

Below is the R script used to generate the Decision Tree.



```
DecisionTree_RuleModel.R

exti<-system.time({
  #opening output file to store results
  sink("DToutput_R_RuleModel.txt", split=TRUE)
  library(C50)
  #opening dataset
  dataset <- "HR_comma_sep.csv"
  #reading dataset
  data <- read.table( file=dataset, header=TRUE, sep="," )
  #randomizing data
  data <- data[ sample( nrow( data ) ), ]
  #selecting the x and y columns and splitting train and test data
  X <- data[,1:9]
  y <- data[,10]
  trainX <- X[1:10500,]
  trainy <- y[1:10500]
  testX <- X[10501:14998,]
  testy <- y[10501:14998]
  trainy<-as.factor(trainy)
  str(trainy)
  #calling decisiontree classifier with train data
  dtclassifier <-C5.0( trainX, trainy, rules=TRUE )
  dtsummary<-summary( dtclassifier )
  print( dtsummary)
  print(dtclassifier)
  #predicting test data
  prediction <- predict( dtclassifier, testX, type="class" )
  accuracy <- sum( prediction == testy ) / length( prediction )
  confusion_matrix<-table(testy, Predicted=prediction)
  cat(sprintf("predicted:%s original: %s\n", prediction, testy))
  print("Confusion Matrix:")
  print(confusion_matrix)
  print ("Accuracy: ")
  print(accuracy)
  sink()
  # Stop the clock
})
print(exti)
```

Output file

```

D\Toutput_R_RuleModel.txt
Factor w/ 2 levels "No","Yes": 1 1 2 1 2 1 1 1 1 1 ...

Call:
C5.0.default(x = trainX, y = trainy, rules = TRUE)

C5.0 [Release 2.07 GPL Edition]          Mon Apr 10 19:59:27 2017
-----

Class specified by attribute 'outcome'

Read 10500 cases (10 attributes) from undefined.data

Rules:

Rule 1: (312, lift 1.3)
      last_evaluation <= 0.44
      -> class No [0.997]

Rule 2: (357, lift 1.3)
      average_monthly_hours <= 125
      -> class No [0.997]

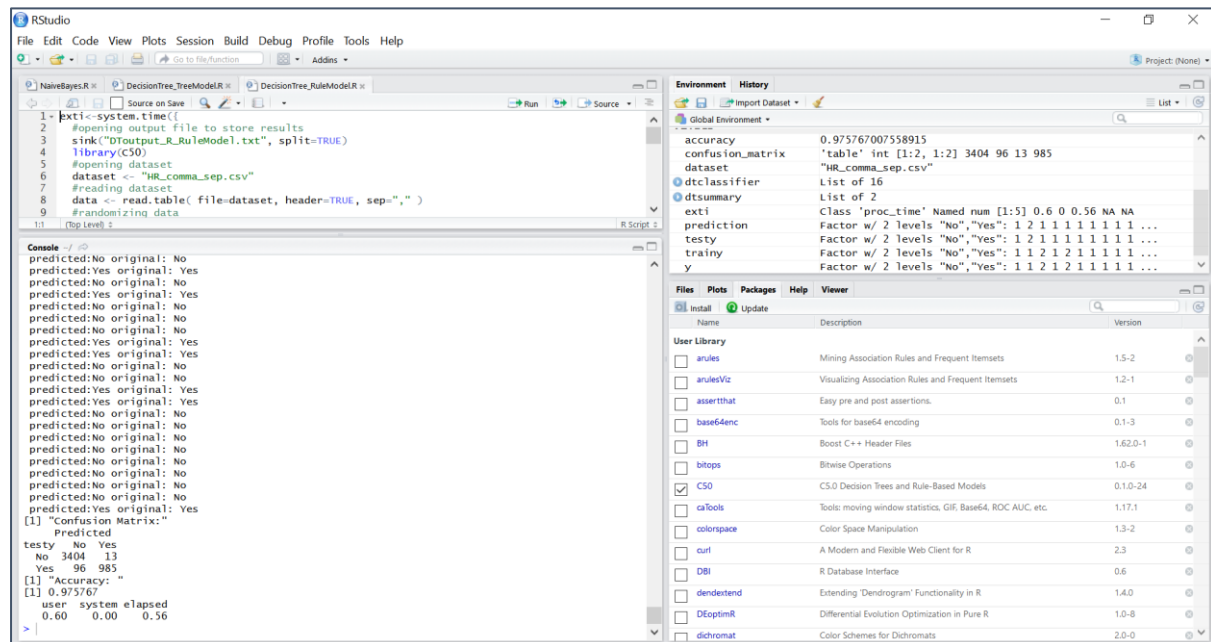
Rule 3: (397, lift 1.3)
      time_spend_company > 6
      -> class No [0.997]

Rule 4: (2358/23, lift 1.3)
      satisfaction_level > 0.71
      last_evaluation <= 0.8
      average_monthly_hours <= 287
      -> class No [0.990]

```

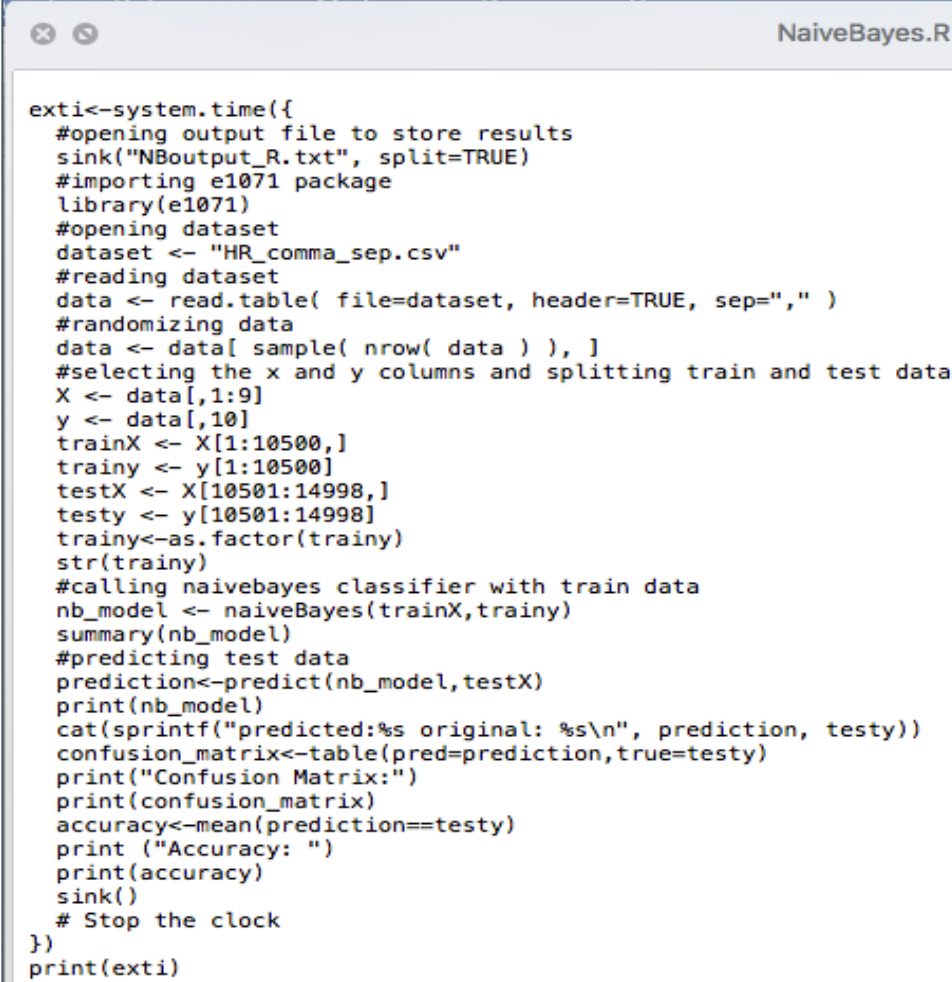
Output is stored in file **DToutput_R_TreeModel.txt**

Output Execution Screen shot



3.2.3 Naïve Bayes

Below is the R script used to generate the Naïve Bayes.



```
exti<-system.time({
  #opening output file to store results
  sink("NBoutput_R.txt", split=TRUE)
  #importing e1071 package
  library(e1071)
  #opening dataset
  dataset <- "HR_comma_sep.csv"
  #reading dataset
  data <- read.table( file=dataset, header=TRUE, sep="," )
  #randomizing data
  data <- data[ sample( nrow( data ) ), ]
  #selecting the x and y columns and splitting train and test data
  X <- data[,1:9]
  y <- data[,10]
  trainX <- X[1:10500,]
  trainy <- y[1:10500]
  testX <- X[10501:14998,]
  testy <- y[10501:14998]
  trainy<-as.factor(trainy)
  str(trainy)
  #calling naiveBayes classifier with train data
  nb_model <- naiveBayes(trainX,trainy)
  summary(nb_model)
  #predicting test data
  prediction<-predict(nb_model,testX)
  print(nb_model)
  cat(sprintf("predicted:%s original: %s\n", prediction, testy))
  confusion_matrix<-table(pred=prediction,true=testy)
  print("Confusion Matrix:")
  print(confusion_matrix)
  accuracy<-mean(prediction==testy)
  print ("Accuracy: ")
  print(accuracy)
  sink()
  # Stop the clock
})
print(exti)
```

Step by step detail of the above R Script:

- Opening output file to store the result of each step (DToutput_R_TreeModel.txt)
- Importing package e1071
- Importing source dataset
- Randomizing data
- Selecting target column Y and splitting dataset into the train(70%) and test(30%).
- Calling NaiveBayes classifier with training dataset to train the model
- Predicting test dataset with the trained model
- Printing accuracy and confusion matrix.

The output file is stored in the file named **NBoutput_R.txt**. In next section, we will describe about the output generated by the Weka.

Output file

```

Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 2 ...

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = trainX, y = trainy)

A-priori probabilities:
trainy
      No      Yes
0.7650476 0.2349524

Conditional probabilities:
      satisfaction_level
trainy      [,1]      [,2]
      No  0.6631682 0.2175565
      Yes 0.4402270 0.2639574

      last_evaluation
trainy      [,1]      [,2]
      No  0.7159417 0.1627246
      Yes 0.7173247 0.1975143

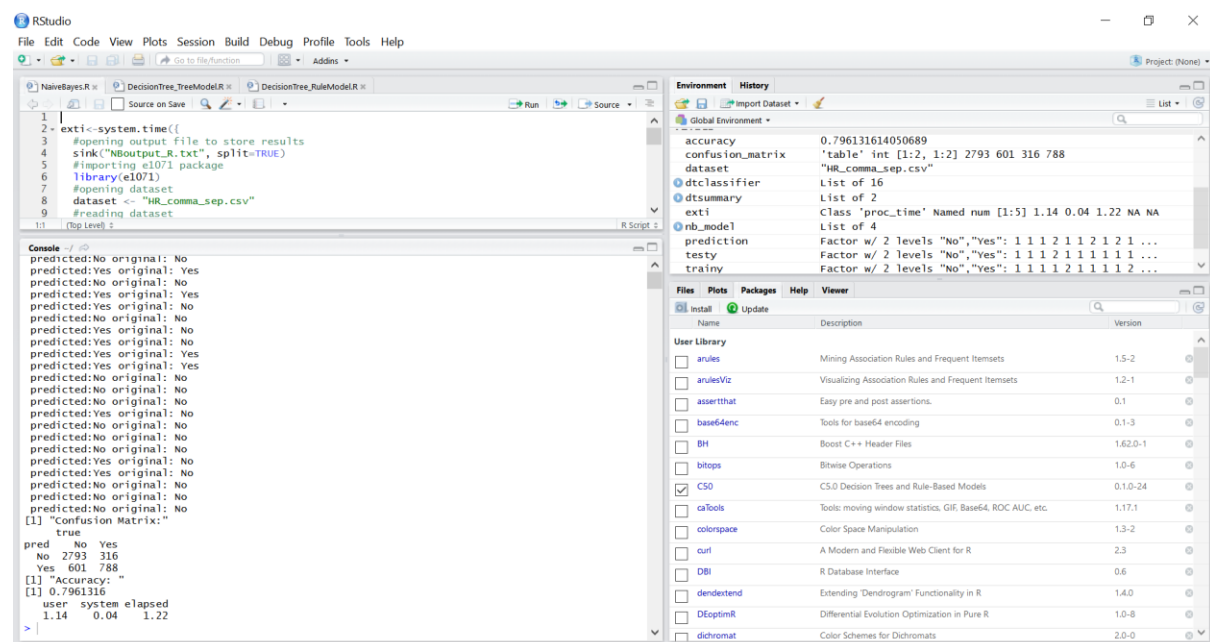
      number_project
trainy      [,1]      [,2]
      No  3.784389 0.9759663
      Yes 3.843535 1.8225857

      average_monthly_hours
trainy      [,1]      [,2]
      No 199.0381 45.83321
      Yes 206.9712 61.19958

```

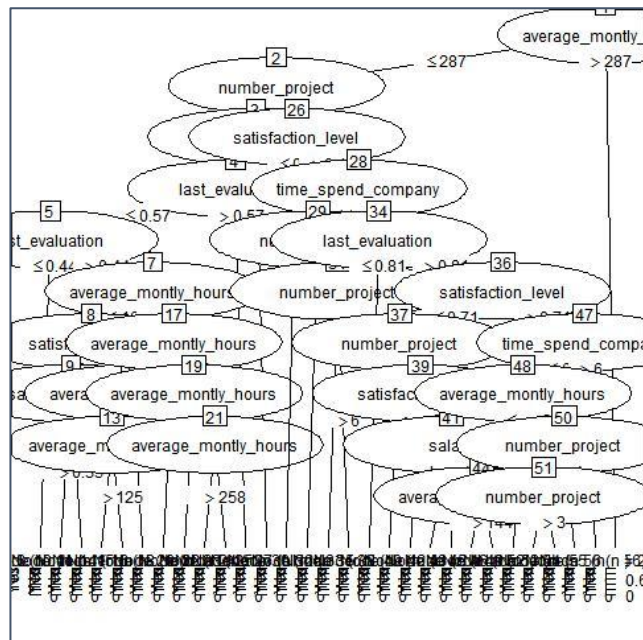
Image above shows part of NaiveBayes model.

Output execution Screen shot



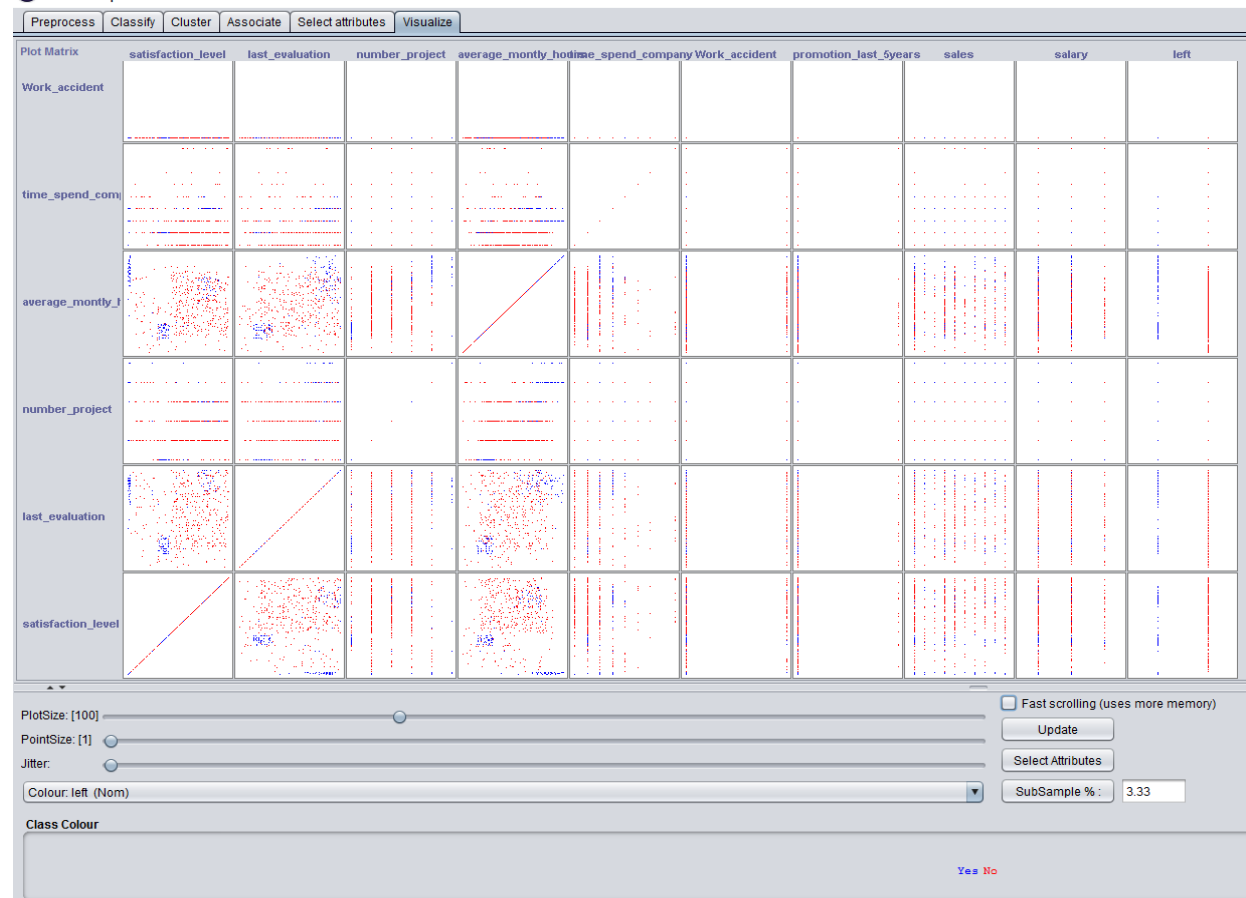
Output Visualization

Decision Tree Visualization in RStudio

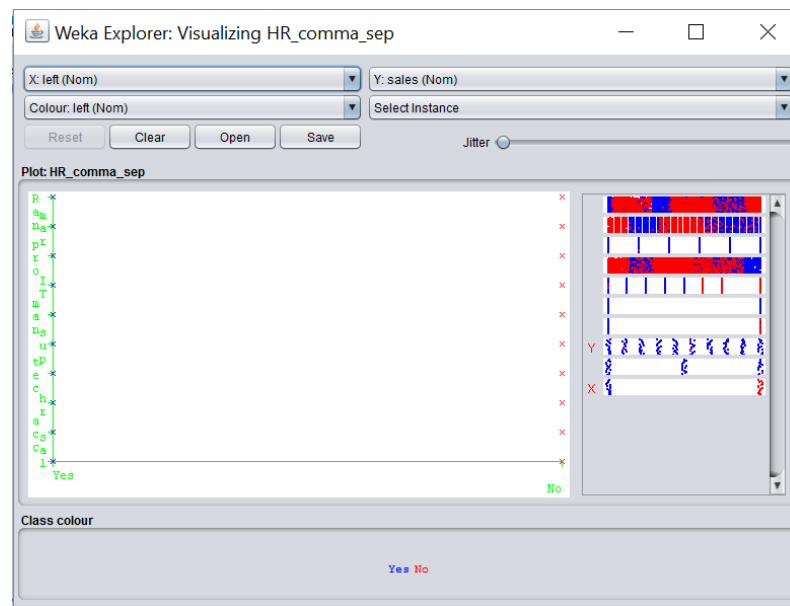


Visualizing relationship between each attribute using Weka

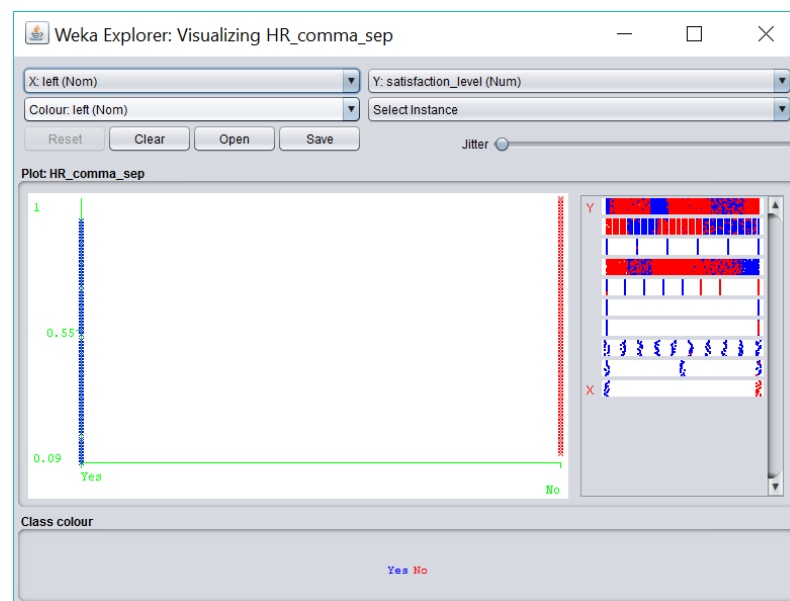
Weka Explorer



Below image depicts the visualization between attribute “left” in X-axis and “sales” in Y axis



Below image depicts the visualization between attribute “left” in X-axis and “satisfaction_level” in Y axis



4. Observation Summary

Comparison between DecisionTree - tree model and rule model in R:

The accuracy is almost the same between both the models, 0.977 for tree based and 0.975 for rule based. In the confusion matrix of tree model, number of class Yes classified as No is 89 and No classified as Yes is 13. In the confusion matrix of rule model, number of class Yes classified as No is 96 and No classified as Yes is 13. Execution time taken for Tree based (1950 milliseconds) model is more compared to rule based model (560 milliseconds).

Comparison between DecisionTree and NaiveBayes model in R:

DecisionTree is more accurate than NaiveBayes based on the accuracy 0.79 and 0.97. In confusion matrix of NaiveBayes, number of class Yes classified as No is 601 and No classified as Yes is 316 which obviously is a lot of wrong classifications compared to decision tree. The execution time for NaiveBayes (1220 milliseconds) compared to DecisionTree models (1950 and 560 milliseconds).

Comparison between DecisionTree in Weka and R:

Time taken to build the model is 140 milliseconds in weka tool which is not very different from 100 milliseconds taken by RStudio. Comparing the confusion matrix of R and Weka outputs, both weka and R are equally accurate. Number of Yes classified as No is 19 and No classified as Yes is 100, which is approximately same as DecisionTree classification using R as discussed above.

Comparison between NaiveBayes in Weka and R:

Time taken to build the model is 10 milliseconds in Weka and 1220 milliseconds in R which is a huge difference. But based on the accuracy, number of Yes classified as No is 254 and No classified as Yes is 416. Based on the manually calculated accuracy $((666+3164 = 3830)(666+3164+416+254 = 4500)(3830/4500 = 0.85))$ from the confusion matrix which is 0.85, accuracy of weka tool is slightly higher than implementation using R.

5. References:

- [1] <https://www.kaggle.com/ludobenistant/hr-analytics>
- [2] [https://en.wikipedia.org/wiki/Weka_\(machine_learning\)](https://en.wikipedia.org/wiki/Weka_(machine_learning))
- [3] [https://en.wikipedia.org/wiki/R_\(programming_language\)](https://en.wikipedia.org/wiki/R_(programming_language))
- [4] https://en.wikipedia.org/wiki/Decision_tree
- [5] https://en.wikipedia.org/wiki/Naive_Bayes_classifier
- [6] https://en.wikipedia.org/wiki/Statistical_classification
- [7] <http://connor-johnson.com/2014/08/29/decision-trees-in-r-using-the-c50-package/>
- [8] <https://bicornor.com/2015/07/16/naive-bayes-using-r/>