KIT306/606 Tutorial 6

Student ID	Name

The following tutorial work should be completed by tutorial 7 (week8).

● Datasets: Training dataset and testing dataset

Last Week, we learned how to collect, pre-process, transform the Wine data, and divided the wine data into two types of dataset: training dataset and testing dataset.

Wine Data Set

Download: Data Folder, Data Set Description

Abstract: Using chemical analysis determine the origin of wines

Data Set Characteristics:	Multivariate	Number of Instances:	178	Area:	Physical
Attribute Characteristics:	Integer, Real	Number of Attributes:	13	Date Donated	1991-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	442655

(Please refer tutorial 5) we divided into 7:3 (7- training dataset and 3- testing dataset) as follows: .seed is an integer vector, containing the random number generator (RNG) **state** for random number generation in **R**.

```
> # Partitioning the data into training and test data
> data.size <- nrow(wine.scale)
> set.seed(1111)
> samp <- c(sample(1:data.size, data.size * 0.7))
> data.tr <- wine.scale[samp, ]
> data.test <- wine.scale[-samp, ]
```

```
> summary(data.tr)
 Type     Alcohol               Malic               Ash                Alcalinity          Magnesium           Phenols
 1:41   Min.   :-1.897718177   Min.   :-1.29468144   Min.   :-3.66881295   Min.   :-2.66350471   Min.   :-2.08238105   Min.   :-2.10131846
 2:50   1st Qu.:-0.816822248   1st Qu.:-0.68151186   1st Qu.:-0.57051311   1st Qu.:-0.74708674   1st Qu.:-0.62955249   1st Qu.:-0.79110255
 3:33   Median : 0.023874586   Median :-0.47115441   Median :-0.02375432   Median :-0.14820613   Median :-0.08692977   Median : 0.16759202
        Mean   : 0.004801705   Mean   :-0.02069784   Mean   :-0.01052628   Mean   :-0.04002124   Mean   : 0.08867446   Mean   : 0.04968805
        3rd Qu.: 0.778346103   3rd Qu.: 0.69923861   3rd Qu.: 0.56856771   3rd Qu.: 0.60039464   3rd Qu.: 0.71825232   3rd Qu.: 0.80672173
        Max.   : 2.253414907   Max.   : 2.96617577   Max.   : 3.14744670   Max.   : 3.14563725   Max.   : 4.35907571   Max.   : 2.53237195
  Flavanoids          Nonflavanoids        Proanthocyanins         Color                 Hue                 Dilution             Proline
 Min.   :-1.69119985   Min.   :-1.86297878   Min.   :-2.06321410   Min.   :-1.629691113   Min.   :-2.08884004   Min.   :-1.8897232   Min.   :-1.38101917
 1st Qu.:-0.79267435   1st Qu.:-0.81841060   1st Qu.:-0.59560339   1st Qu.:-0.827374184   1st Qu.:-0.69977837   1st Qu.:-0.9636544   1st Qu.:-0.77925511
 Median : 0.13588543   Median :-0.17559941   Median : 0.05957997   Median :-0.176030871   Median : 0.05490867   Median : 0.2863625   Median :-0.21559748
 Mean   : 0.02123845   Mean   :-0.05118434   Mean   : 0.08015132   Mean   : 0.009590142   Mean   :-0.02602878   Mean   : 0.0155728   Mean   : 0.01083872
 3rd Qu.: 0.88423953   3rd Qu.: 0.54756319   3rd Qu.: 0.66671654   3rd Qu.: 0.519526143   3rd Qu.: 0.59084585   3rd Qu.: 0.8039751   3rd Qu.: 0.72039173
 Max.   : 3.05421616   Max.   : 2.15459116   Max.   : 3.47526919   Max.   : 3.425768243   Max.   : 2.15490741   Max.   : 1.9553990   Max.   : 2.96311399
```

```
> summary(data.test)
 Type     Alcohol              Malic                Ash                Alcalinity          Magnesium          Phenols
 1:18   Min.   :-2.42738798   Min.   :-1.42895215   Min.   :-2.42949302   Min.   :-2.48384052   Min.   :-1.5222544   Min.   :-1.6219712
 2:21   1st Qu.:-0.76139169   1st Qu.:-0.62109004   1st Qu.:-0.58873840   1st Qu.:-0.59736659   1st Qu.:-0.8220960   1st Qu.:-1.0187925
 3:15   Median : 0.06082829   Median :-0.21156437   Median :-0.18778195   Median : 0.15123418   Median :-0.4370089   Median :-0.1839293
        Mean   :-0.01102614   Mean   : 0.04752837   Mean   : 0.02417146   Mean   : 0.09190064   Mean   :-0.2036228   Mean   :-0.1140985
        3rd Qu.: 0.90768408   3rd Qu.: 0.61196265   3rd Qu.: 0.92396093   3rd Qu.: 0.60039464   3rd Qu.: 0.1581256   3rd Qu.: 0.7468033
        Max.   : 1.69910930   Max.   : 3.10044648   Max.   : 2.01747852   Max.   : 2.69647679   Max.   : 2.3986323   Max.   : 1.6056339
  Flavanoids            Nonflavanoids        Proanthocyanins      Color                Hue                  Dilution              Proline
 Min.   :-1.5610513142   Min.   :-1.7826274   Min.   :-1.6613683   Min.   :-1.36225214   Min.   :-1.82634019   Min.   :-1.84746912   Min.   :-1.48898739
 1st Qu.:-0.9828914517   1st Qu.:-0.7380592   1st Qu.:-0.7834226   1st Qu.:-0.74973061   1st Qu.:-0.88571576   1st Qu.:-0.80872272   1st Qu.:-0.77607957
 Median : 0.0007311716   Median :-0.2157751   Median :-0.2723796   Median :-0.13720908   Median : 0.01115870   Median : 0.18072723   Median :-0.38707642
 Mean   :-0.0487697690   Mean   : 0.1175344   Mean   :-0.1840512   Mean   :-0.02202181   Mean   : 0.05976978   Mean   :-0.03575977   Mean   :-0.02488891
 3rd Qu.: 0.7315652835   3rd Qu.: 1.0497594   3rd Qu.: 0.3609643   3rd Qu.: 0.42247168   3rd Qu.: 0.82053321   3rd Qu.: 0.75819981   3rd Qu.: 0.91092389
 Max.   : 1.6125707883   Max.   : 2.3956454   Max.   : 2.3920327   Max.   : 2.24386051   Max.   : 3.29240673   Max.   : 1.33567238   Max.   : 2.54076771
```

In this week, we will learn how to use three different classification techniques (incl. **decision tree**, **rule-based method** and **support vector machine**) in order to find the unique pattern (model). The found unique pattern will help you to predict/ classify the class (Type of wine) based on the other 13 attributes (e.g. Alcohol, Malic, Ash, etc.)

Data mining with RWeka

Weka (Waikato Environment for Knowledge Analysis) is a popular suite of data mining software written in Java, developed at the University of Waikato, New Zealand. Weka is a collection of machine learning algorithms for data mining tasks written in Java, containing tools for data pre-processing, classification, regression, clustering, association rules, and visualization

Weka provides the package **RWeka**, which contains the interface code, the Weka jar is in a separate package RWekajars. For more information on Weka see <http://www.cs.waikato.ac.nz/ml/weka/>.

First, install **RWeka** package (Please use Melbourne Mirror)

```
> install.packages("RWeka")
trying URL 'http://cran.ms.unimelb.edu.au/bin/macosx/mavericks/contrib/3.2/RWeka_0.4-24.tgz'
Content type 'application/x-gzip' length 536260 bytes (523 KB)
==================================================
downloaded 523 KB


The downloaded binary packages are in
	/var/folders/r6/01v1jzhn28x_x1s0z0l6hb300000gn/T//Rtmpb6HVlv/downloaded_packages
```

You can find the manual of RWeka package (https://cran.r-project.org/web/packages/RWeka/RWeka.pdf). RWeka provides various types of classification techniques. In this week, we will use the following classification techniques from RWeka package.

1. Tree-based Classification
   J48 Decision Tree: Implementation of algorithm C4.5 Decision Tree
2. Rule-based Classification
   JRip RIPPER: Implementation of algorithm Propositional Rule Learner
3. Support Vector Machine based Classification
   SMO SVM Classifier: Implementation of Support Vector Machine using Sequential Minimal Optimization

Execute the package RWeka with the following command and you are now ready to use RWeka library.

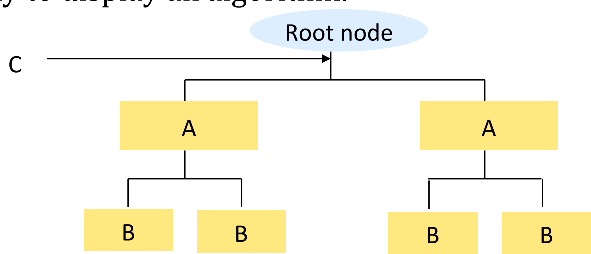
```
> library("RWeka")
```

We will use the wine data as it is. Don't forget to convert the Type column into the categorical variable. ☺

```
> wine<-read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/
wine.data",sep=",",col.names=c("Type","Alcohol","Malic","Ash","Alcalinity","Magnesium"
,"Phenols","Flavanoids","Nonflavanoids","Proanthocyanins","Color","Hue","Dilution","Pr
oline"))
> wine$Type=factor(wine$Type)
```

● Data Mining using J48 Decision Tree Classifier

The first classification technique is decision tree classifier. 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.



A. Nodes of the tree
B. Leaves (terminal nodes) of the tree
C. Branches (decision point) of the tree

RWeka provides the J48 function, which is the implementation of C4.5 decision tree. If you want to see the detailed information of C4.5 decision tree, please read https://en.wikipedia.org/wiki/C4.5_algorithm.
We will mine the unique and important pattern by using training dataset (`data.tr`) that extracted from wine data in last tutorial.

```
> wine_dt=J48(Type~.,data=data.tr)
> wine_dt
J48 pruned tree
------------------

Dilution <= -0.65027
|   Color <= -0.542681: 2 (6.0)
|   Color > -0.542681: 3 (34.0/1.0)
Dilution > -0.65027
|   Alcohol <= -0.284073: 2 (40.0)
|   Alcohol > -0.284073
|   |   Ash <= -1.408877: 2 (2.0)
|   |   Ash > -1.408877: 1 (42.0/1.0)

Number of Leaves  :  5

Size of the tree :   9
```

The above tree represents the learned pattern (model) by using training dataset and C4.5 decision tree classification algorithm. In order to evaluate the accuracy (performance) of the learned pattern, we will use the testing data (`data.test`) processed in the last tutorial.
We will use predict function, which is a generic function for predictions from the results of various model fitting functions.

```
> prediction_dt=predict(wine_dt,data.test[,2:14])
```

The learned pattern (model) by using data.tr and c4.5 decision tree

We need all attributes (features) value except class in order to make prediction

The rest is exactly same as what you did in the last tutorial work.
The following commands allow you to compare the actual wine type and the predicted wine type.

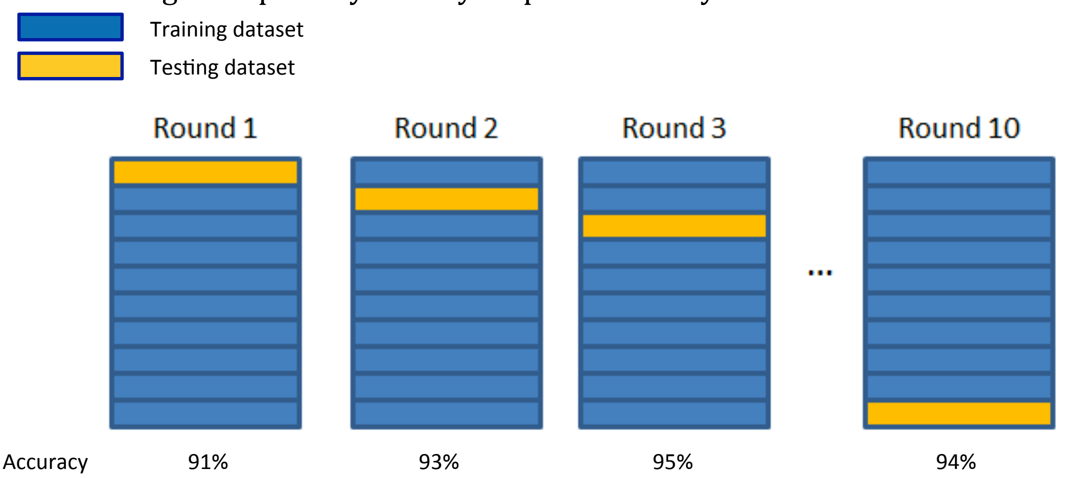
```
> prediction_dt
> actual=data.test$Type
> model.confusion.matrix=table(actual,prediction_dt)
> model.confusion.matrix
      prediction_dt
actual  1  2  3
     1 18  0  0
     2  3 17  1
     3  2  0 13
```

The way to see the error rate can be found in the last week tutorial work! ☺

- K-fold cross validation (normally, k is 10)

However, is there any approach without splitting training and testing dataset? Yes, there is! ☺ You can apply k-fold cross validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation)

The following example may be very helpful to make you understand ☺



Final Accuracy = Average(Round1, Round2, Round3, ... ,Round10)

So, assume that we did not split the data into training and testing dataset, and use whole **wine data** with 10-fold cross validation. Make sure that you used factor command to convert the Type column into categorical variable.

```
> wine_dt=J48(Type~., data=wine)
> wine_dt
J48 pruned tree
------------------

Flavanoids <= 1.57
|   Color <= 3.8: 2 (13.0)
|   Color > 3.8: 3 (49.0/1.0)
Flavanoids > 1.57
|   Proline <= 720: 2 (54.0/1.0)
|   Proline > 720
|   |   Color <= 3.4: 2 (4.0)
|   |   Color > 3.4: 1 (58.0)

Number of Leaves  :   5

Size of the tree :    9
```

Now, we do not need to have testing dataset to evaluate ☺ Let's just apply 10-fold cross validation by executing following command!

```
> eval_j48 <- evaluate_Weka_classifier(wine_dt, numFolds = 10, complexity = FALSE,
+      seed = 1, class = TRUE)
> eval_j48
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         167               93.8202 %
Incorrectly Classified Instances        11                6.1798 %
Kappa statistic                          0.9058
Mean absolute error                      0.0486
Root mean squared error                  0.2019
Relative absolute error                 11.0723 %
Root relative squared error             43.0865 %
Coverage of cases (0.95 level)          94.382  %
Mean rel. region size (0.95 level)      33.8951 %
Total Number of Instances              178

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.983    0.034    0.935      0.983    0.959      0.938    0.977     0.942     1
                 0.944    0.056    0.918      0.944    0.931      0.884    0.937     0.884     2
                 0.875    0.008    0.977      0.875    0.923      0.899    0.946     0.901     3
Weighted Avg.    0.938    0.036    0.940      0.938    0.938      0.906    0.953     0.908

=== Confusion Matrix ===

  a  b  c   <-- classified as
 58  1  0 |  a = 1
  3 67  1 |  b = 2
  1  5 42 |  c = 3
```

So, which one is better for you?

- ➢ Splitting training dataset and testing dataset?
- ➢ Or 10-fold cross validation approach?

And, which one achieves better performance (accuracy)?

● Data Mining using JRip Rule-based Classifier

The second is rule-based Classifier, which allows classifying records by using a collection of "if…then…" rules. The rule should be structured as following manner:

Rule: (*Condition*) → *y*.

The *Condition* is conjunctions of attributes, and *y* is the class label

In this tutorial, we will use **JRip**, which is the implementation of a propositional rule learner, called "Repeated Incremental Pruning to Produce Error Reduction" (RIPPER). You can find the detailed information from
http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/JRip.html

The following commands allows you to mine the pattern (model) using **JRip**

```
> wine_jrip=JRip(Type~., data=wine)
> wine_jrip
JRIP rules:
===========

(Dilution <= 2.15) and (Color >= 3.85) => Type=3 (48.0/2.0)
(Flavanoids <= 0.5) => Type=3 (2.0/0.0)
(Proline >= 760) and (Alcohol >= 12.85) => Type=1 (58.0/1.0)
(Alcohol >= 13.24) and (Alcohol <= 13.24) => Type=1 (2.0/0.0)
 => Type=2 (68.0/0.0)

Number of Rules : 5
```

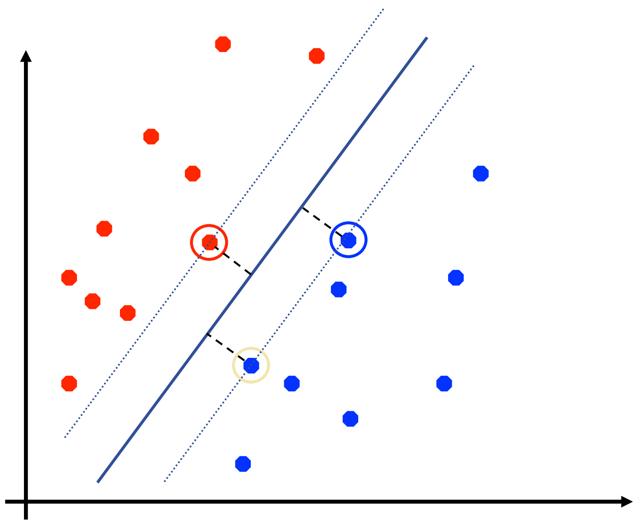
The model contains the above 5 rules. Please have a look the rules carefully, and find which one is condition and class.

Let's try to apply 10-fold cross validation to evaluate the learned pattern.

```
> eval_jrip <- evaluate_Weka_classifier(wine_jrip, numFolds = 10, complexity = FALSE, seed = 1, class = TRUE)
> eval_jrip
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         164               92.1348 %
Incorrectly Classified Instances        14                7.8652 %
Kappa statistic                          0.8799
Mean absolute error                      0.0607
Root mean squared error                  0.2262
Relative absolute error                 13.8153 %
Root relative squared error             48.2704 %
Coverage of cases (0.95 level)          93.2584 %
Mean rel. region size (0.95 level)      36.3296 %
Total Number of Instances              178

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.898    0.025    0.946      0.898    0.922      0.885    0.933     0.901     1
                 0.944    0.093    0.870      0.944    0.905      0.840    0.928     0.829     2
                 0.917    0.008    0.978      0.917    0.946      0.928    0.964     0.950     3
Weighted Avg.    0.921    0.048    0.924      0.921    0.922      0.879    0.940     0.885

=== Confusion Matrix ===

  a  b  c   <-- classified as
 53  6  0 |  a = 1
  3 67  1 |  b = 2
  0  4 44 |  c = 3
```

Data Mining using SMO Support Vector Machine Classifier

The third classification technique we will learn today is 'Support Vector Machine'. ☺
In machine learning, support vector machines (SVMs, also support vector networks[1]) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier.



RWeka package provides the function SMO, which is the implementation of John C. Platt's sequential minimal optimization algorithm for training a support vector classifier using polynomial or RBF kernels.

The following commands allows you to mine the pattern (model) using **SMO.**

```
> wine_svm=SMO(Type~., data=wine)
> wine_svm
```

Let's try to apply 10-fold cross validation to evaluate the learned pattern.

```
> eval_svm <- evaluate_Weka_classifier(wine_svm, numFolds = 10, complexity = FALSE, seed = 1, class = TRUE)
> eval_svm
=== 10 Fold Cross Validation ===

=== Summary ===

Correctly Classified Instances         175               98.3146 %
Incorrectly Classified Instances         3                1.6854 %
Kappa statistic                          0.9745
Mean absolute error                      0.226
Root mean squared error                  0.279
Relative absolute error                 51.4678 %
Root relative squared error             59.5404 %
Coverage of cases (0.95 level)         100      %
Mean rel. region size (0.95 level)      66.6667 %
Total Number of Instances              178

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.008    0.983      1.000    0.992      0.987    0.996     0.983     1
                 0.958    0.000    1.000      0.958    0.978      0.965    0.980     0.975     2
                 1.000    0.015    0.960      1.000    0.980      0.972    0.992     0.960     3
Weighted Avg.    0.983    0.007    0.984      0.983    0.983      0.974    0.989     0.974

=== Confusion Matrix ===

  a  b  c   <-- classified as
 59  0  0 |  a = 1
  1 68  2 |  b = 2
  0  0 48 |  c = 3
```

You can get the confusion matrix by using vector call.

```
> eval_svm[4]
$confusionMatrix
    predicted
      1  2  3
   1 59  0  0
   2  1 68  2
   3  0  0 48

> eval_svm[4]$confusionMatrix
    predicted
      1  2  3
   1 59  0  0
   2  1 68  2
   3  0  0 48
> eval_svm[4]$confusionMatrix[1]
[1] 59
> eval_svm[4]$confusionMatrix[2]
[1] 1
> eval_svm[4]$confusionMatrix[3]
[1] 0
> eval_svm[4]$confusionMatrix[4]
[1] 0
> eval_svm[4]$confusionMatrix[5]
[1] 68
```

Ok, for now, we learned 5 different classification techniques (Nearest neighbour, Neural Network, Decision Tree, Classification Rule, Support Vector Machine). Which is the best approach for the wine data? And are you satisfied with the performance of the extracted patterns (models)?

- **Other Packages for Decision Tree**

RWeka is not only package that allows users to use classifier and clustering techniques. You can make another types of decision tree using other packages.

Install the package party, which is a computational toolbox for recursive partitioning. The core of the package is ctree(), an implementation of conditional inference trees which embed tree-structured regression models into a well defined theory of conditional inference procedures.

```
> install.packages("party")
--- Please select a CRAN mirror for use in this session ---
also installing the dependencies ‘TH.data’, ‘multcomp’, ‘coin’


  There are binary versions available but the source versions are later:
      binary source needs_compilation
coin  1.0-24  1.1-0              TRUE
party 1.0-22 1.0-23              TRUE

Do you want to install from sources the packages which need compilation?
y/n: n
```

IMPORTANT: Press n for this command!

Execute the library party and start it

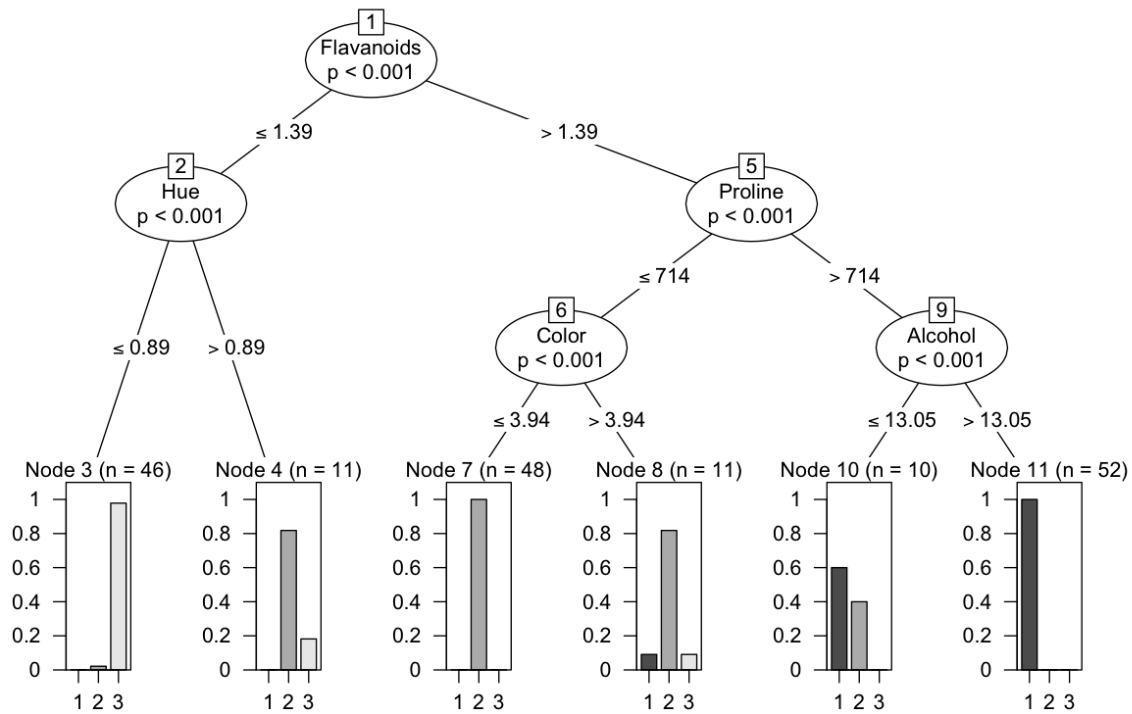
```
> library('party')
```

We will use the core function ctree(), conditional inference decision tree.

```
> wine_ctree <- ctree(Type ~ ., data=wine)
> wine_ctree

	 Conditional inference tree with 6 terminal nodes

Response:  Type 
Inputs:  Alcohol, Malic, Ash, Alcalinity, Magnesium, Phenols, Flavanoids, 
Nonflavanoids, Proanthocyanins, Color, Hue, Dilution, Proline 
Number of observations:  178 

1) Flavanoids <= 1.39; criterion = 1, statistic = 128.816
  2) Hue <= 0.89; criterion = 1, statistic = 29.21
    3)*  weights = 46 
  2) Hue > 0.89
    4)*  weights = 11 
1) Flavanoids > 1.39
  5) Proline <= 714; criterion = 1, statistic = 86.929
    6) Color <= 3.94; criterion = 1, statistic = 20.426
      7)*  weights = 48 
    6) Color > 3.94
      8)*  weights = 11 
  5) Proline > 714
    9) Alcohol <= 13.05; criterion = 1, statistic = 20.638
      10)*  weights = 10 
    9) Alcohol > 13.05
      11)*  weights = 52 
```

You will see different tree nodes and branches.
And, you can also plot this tree using the plot function ☺

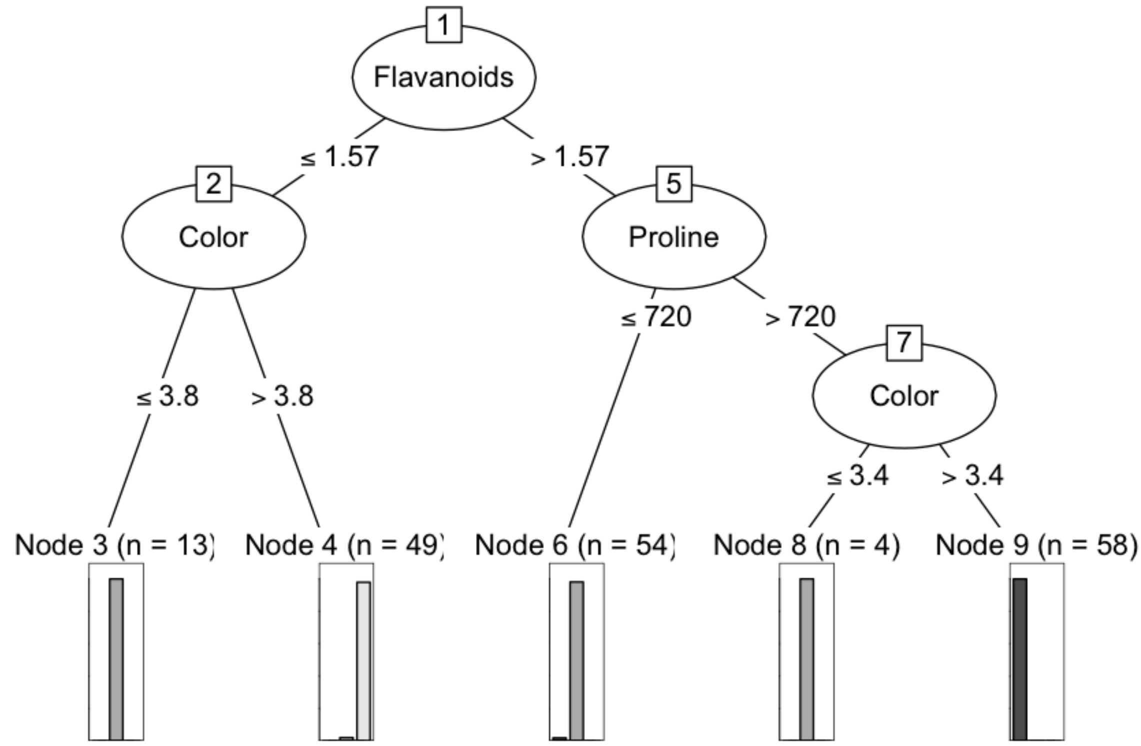
```
> plot(wine_ctree)
```



The plot function will show you the node number and rules, and classified class.

And, hope you remembered that we have a pattern (model) that learned by using RWeka J48 decision tree.

```
> wine_dt
J48 pruned tree
------------------

Flavanoids <= 1.57
|   Color <= 3.8: 2 (13.0)
|   Color > 3.8: 3 (49.0/1.0)
Flavanoids > 1.57
|   Proline <= 720: 2 (54.0/1.0)
|   Proline > 720
|   |   Color <= 3.4: 2 (4.0)
|   |   Color > 3.4: 1 (58.0)

Number of Leaves  :  5

Size of the tree :   9

> plot(wine_dt)
```



Tutorial 6 Question

The data we will use for tutorial 6 questions is Wine.
Please complete the following command

```
> wine<-read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",sep=",",col.names=c("Type","Alcohol","Malic","Ash","Alcalinity","Magnesium","Phenols","Flavanoids","Nonflavanoids","Proanthocyanins","Color","Hue","Dilution","Proline"))
> wine$Type=factor(wine$Type)
```

NOTE: Tutorial 6 requires to use the following libraries:

- library("RWeka")
- library("rvest")
- library("party")

1. **Make three variables (cf_j48, cf_jrip, cf_svm), which include the following confusion matrix result of wine data using three classifiers(J48, jRip, and SMO) and 10-fold cross validation (class is Type)**

```
> cf_j48          > cf_jrip         > cf_svm
   predicted         predicted         predicted
     1  2  3           1  2  3           1  2  3
  1 58  1  0        1 53  6  0        1 59  0  0
  2  3 67  1        2  3 67  1        2  1 68  2
  3  1  5 42        3  0  4 44        3  0  0 48
```

2. **Make three variables (cf_j48_rate, cf_jrip_rate, cf_svm_rate), which include the error rate of confusion matrix result of wine data using prop.table function and round function**

```
> cf_j48_rate               > cf_jrip_rate              > cf_svm_rate
  predicted                   predicted                   predicted
        1     2     3               1     2     3               1     2     3
  1 32.58  0.56  0.00         1 29.78  3.37  0.00         1 33.15   0.0  0.00
  2  1.69 37.64  0.56         2  1.69 37.64  0.56         2  0.56  38.2  1.12
  3  0.56  2.81 23.60         3  0.00  2.25 24.72         3  0.00   0.0 26.97
```

3. **Make a function `best` that compare error rate of all learned model, and pointed the model that achieved the highest accuracy among J48, JRip, SMO classifiers in RWeka. The output of function `best` should be as follows:**

```
> best()
[1] "J48 Error Rate= 6.18"
[1] "JRip Error Rate= 7.87"
[1] "SVM Error Rate= 1.68"
[1] "The best model for wine data is Support Vector Machine"
```