Business Analytics

Today objective

Supervised Learning

Machine Learning Approach

Predictive Analysis

Classification analysis

Decision Tree based Approach

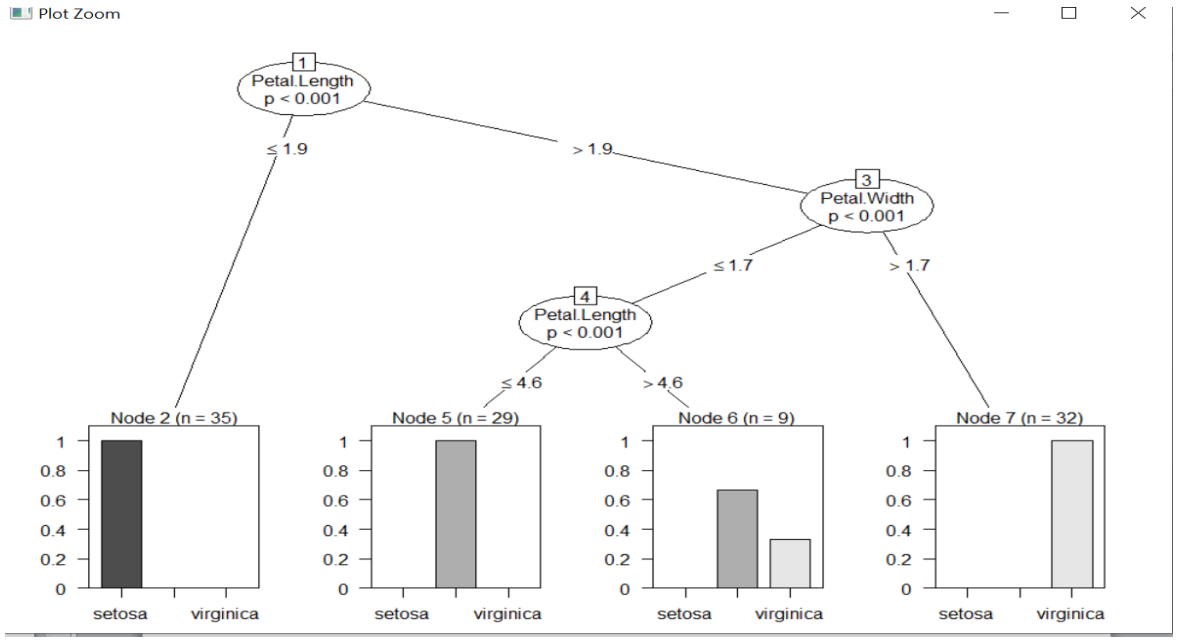
Simulation using R

Splitting into training and test dataset

```
library(caTools)
set.seed(123)
split <- sample.split(iris$Species, SplitRatio = 0.7)
train = subset(iris, split == TRUE)
test = subset(iris, split == FALSE)
```

R code to run decision tree model

```
library("party")
m <- ctree(Species ~ ., data = train)
plot(m)
```



R code to run decision tree model

```
train_predict <- predict(m,train,type="response")
table(train_predict,train$Species)
```

```
train_predict setosa versicolor virginica
    setosa         35          0         0
    versicolor      0         35         3
    virginica       0          0        32
```

R code to run decision tree model

Library(e1071)
Library(caret)

confusionMatrix(train_predict,train$Species)

Confusion Matrix and Statistics

Prediction	Reference: setosa	versicolor	virginica
setosa	35	0	0
versicolor	0	35	3
virginica	0	0	32

Overall Statistics

Accuracy : 0.9714
95% CI : (0.9188, 0.9941)
No Information Rate : 0.3333
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.9571

install.packages("installr")
library(installr)
updateR()

R code to run decision tree model for test data

```
> test_predict <-
predict(m,test,type="response")
> table(test_predict,test$Species)
```

test_predict	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	14	2
virginica	0	1	13

Multiple row prediction

```
newdata2<-
data.frame(Sepal.Length=c(6,3,4,5),Sepal.Width=c(4
,3,2,4),Petal.Length=c(6.3,4.5,5,6.3),Petal.Width=c(2
,3,4,2.3),Species=c('setosa','virginica','virginica','seto
sa'))
predict(m, newdata2,type="response")
```

```
> predict(m, newdata2,type="response")
[1] virginica virginica virginica virginica
Levels: setosa versicolor virginica
```

Enhancing test data

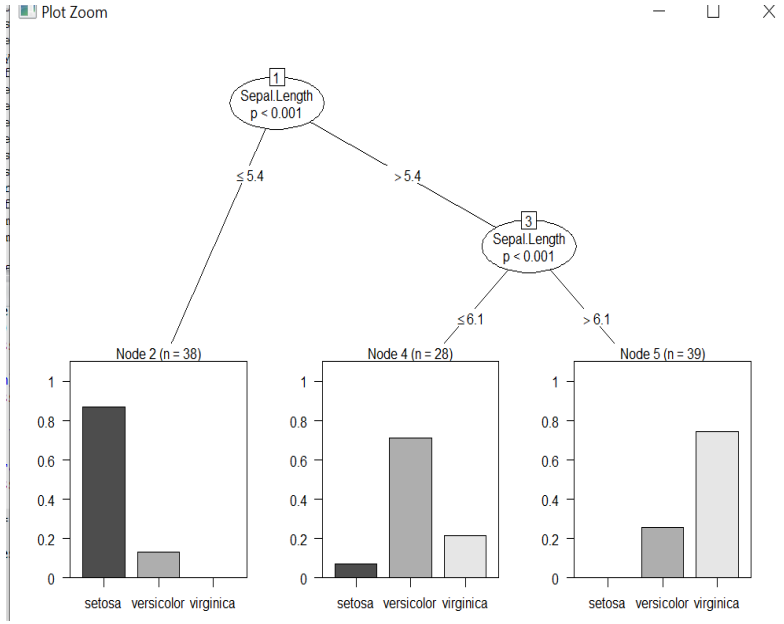
```
newtest<-rbind(test,newdata2)

test_predict_etd <- predict(m,newtest,type="response")

table(test_predict_etd,newtest$Species)
```

```
est_predict_etd setosa versicolor virginica
      setosa         15          0         0
      versicolor      0         14         2
      virginica       2          1        15
```

```
test_predict setosa versicolor virginica
    setosa         15          0         0
    versicolor      0         14         2
    virginica       0          1        13
```

Multiple Checking

```
m1 <- ctree(Species ~ Sepal.Length, data = train)
plot(m1)
```



table(train_predict_new,train$Species)

train_predict	setosa	versicolor	virginica
setosa	33	5	0
versicolor	2	20	6
virginica	0	10	29

```
mean(train_predict_new !=
train$Species) * 100
```

```
train_predict_new <-
predict(m1,train,type="response")
```

[1] 21.90476

Generating training and testing datasets

s <- sample(150, 100) #s contains 100 samples out of 150 in the iris dataset

s # It will show the 100 samples selected randomly from the 150 sample dataset

```
Console ~/
> s <- sample(150, 100)
> s
  [1]  51  63  37  69  31 105 116  14   9  75 132  86  25  29  73  44  58  64 138 111 100  92  17  83  16 114  59  95  55  94  19
 [32] 145 133 109  10 143  87 141  91  13   4 139  28 103  22 115  39  45  77  47 107  56 131   3  26 136  93   2  48  74  34 125
 [63]  11  15 121  46  27 126  50  88 148 140 150 127  53  21 112  12  38  70   7 130  81  79  20  68 104   8 119 142 137 149  52
 [94]  72 128 102 124 118  82  33
>
```

Generating training and testing datasets

iris_train <- iris[s,]

- #the iris_train contains 100 samples out of 150

iris_test <- iris[-s,]

- #the iris_test contains rest 50 samples

Building a Decision Tree

```
library(rpart)
dtm <- rpart(Species ~ Sepal.Length + Sepal.Width
+ Petal.Length + Petal.Width, iris_train,
method="class")
```

Or

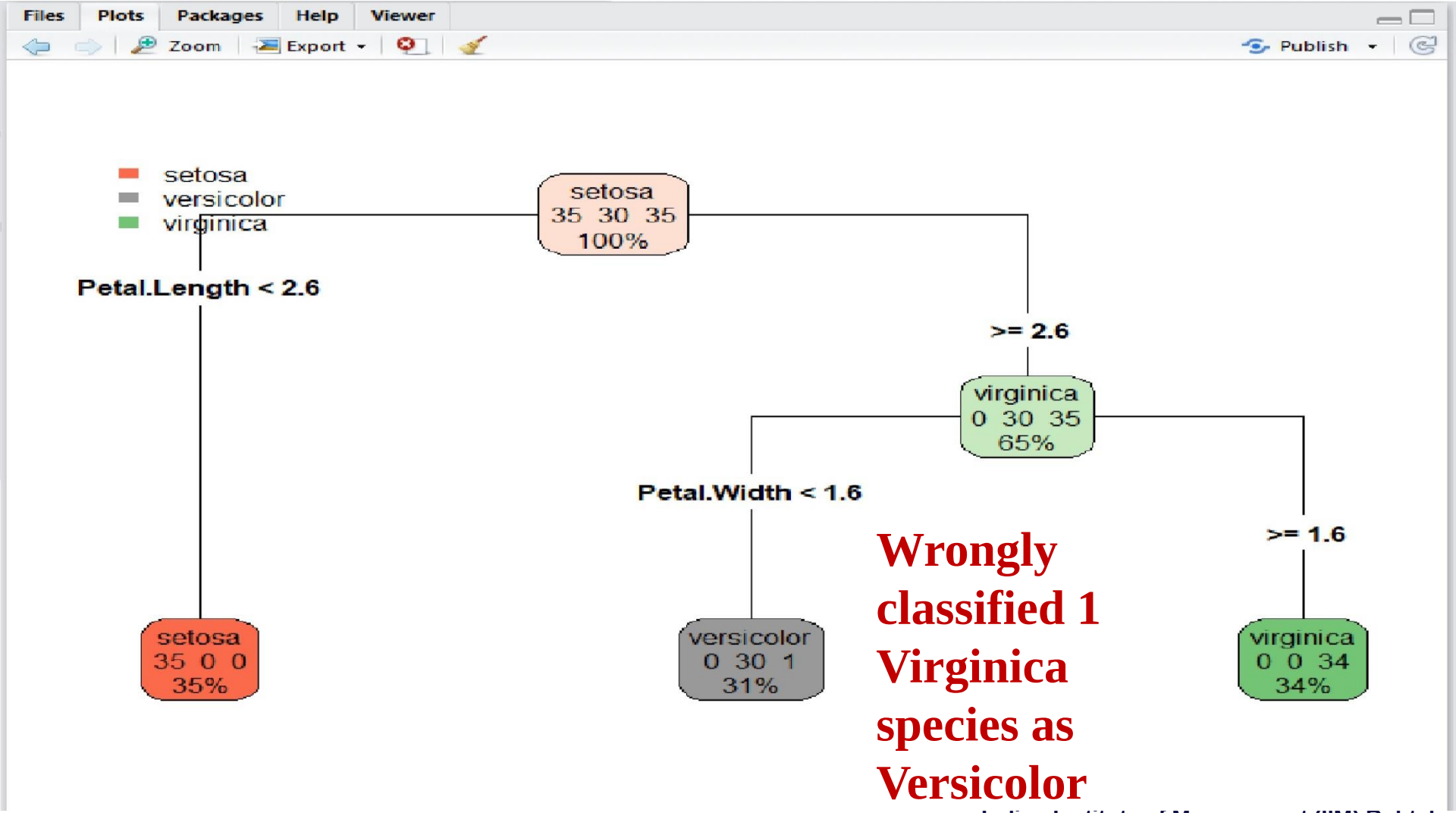
```
dtm <- rpart(Species ~ ., iris_train, method =
"class")
```

- You can use "." as a replacement for all the input columns
- We are building a decision tree from the training data
- The method = "class" determines classification

The Decision Tree

rpart.plot(dtm, type=4, extra=101)

?rpart.plot # check the properties of rpart.plot



Predicting from the testing dataset

p <- predict(dtm, iris_test)

- We take decision tree and predict the iris_test dataset
- table(p,iris_test$Species)

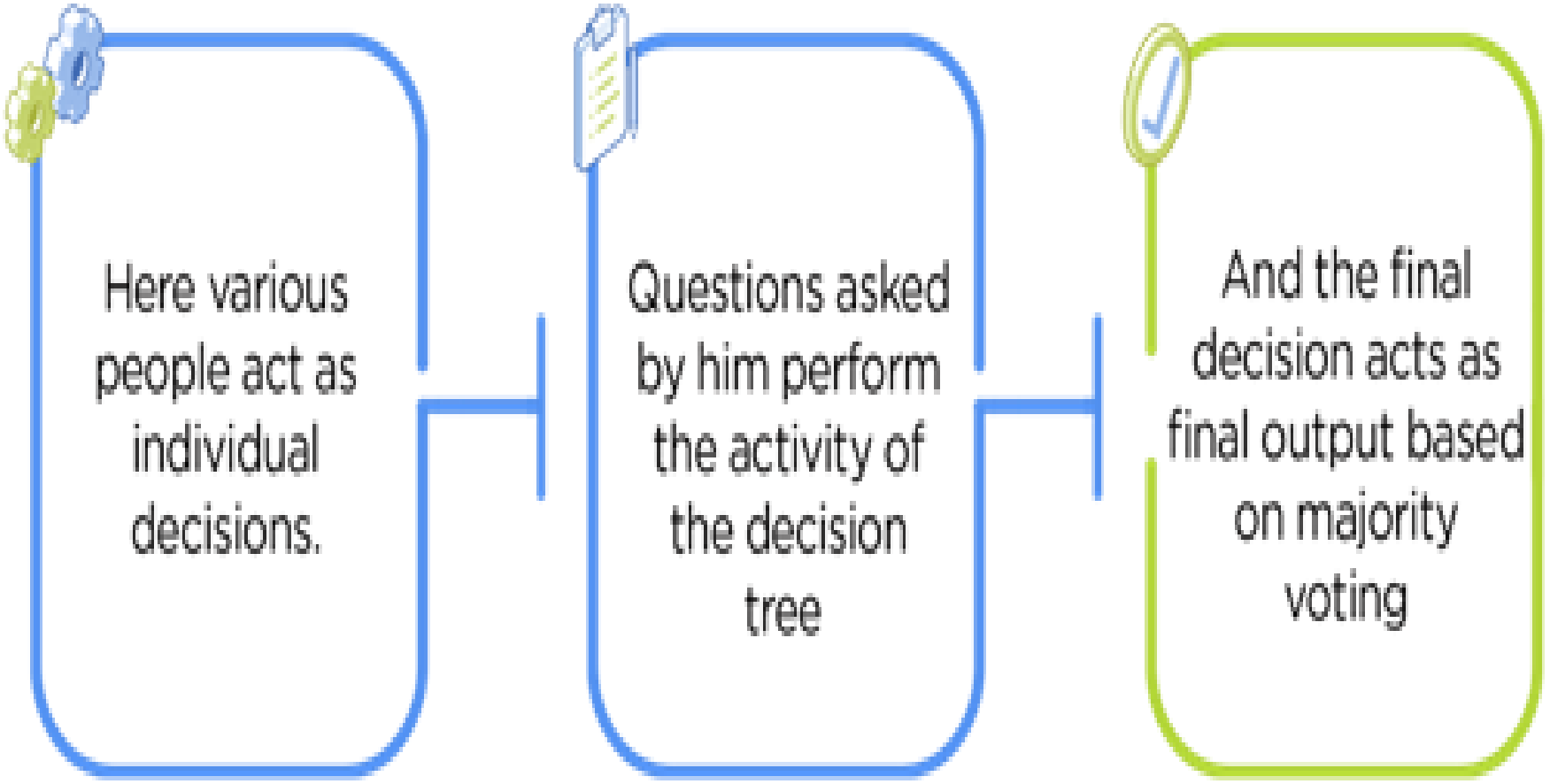
```
Console ~/
> p <- predict(dtm, iris_test, type="class")
> table(iris_test[,5],p)
             p
              setosa versicolor virginica
  setosa          15          0         0
  versicolor       0         18         2
  virginica        0          3        12
>
```

- It classified all Setosa species correctly.
- Out of 20 Versicolor, it classified 18 correctly and the rest 2 are classified under Virginica.
- Similarly for a total of 15 Virginica, it classified 12 correctly and 3 incorrectly in Versicolor.

Random Forest

Random Forest

- Random forests are an ensemble learning method that operate by constructing a lot of decision trees at training time.
- What is ensemble learning method?
  - ***Ensemble methods*** *are* ***learning*** *algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions.*


Here various people act as individual decisions.
Questions asked by him perform the activity of the decision tree
And the final decision acts as final output based on majority voting

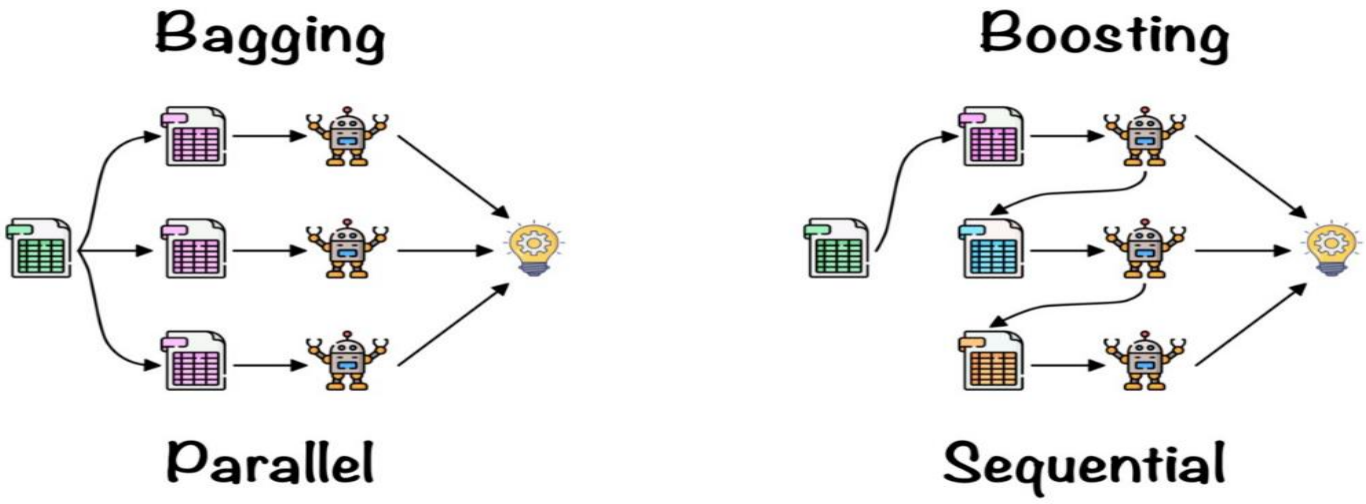
Working of Random Forest Algorithm

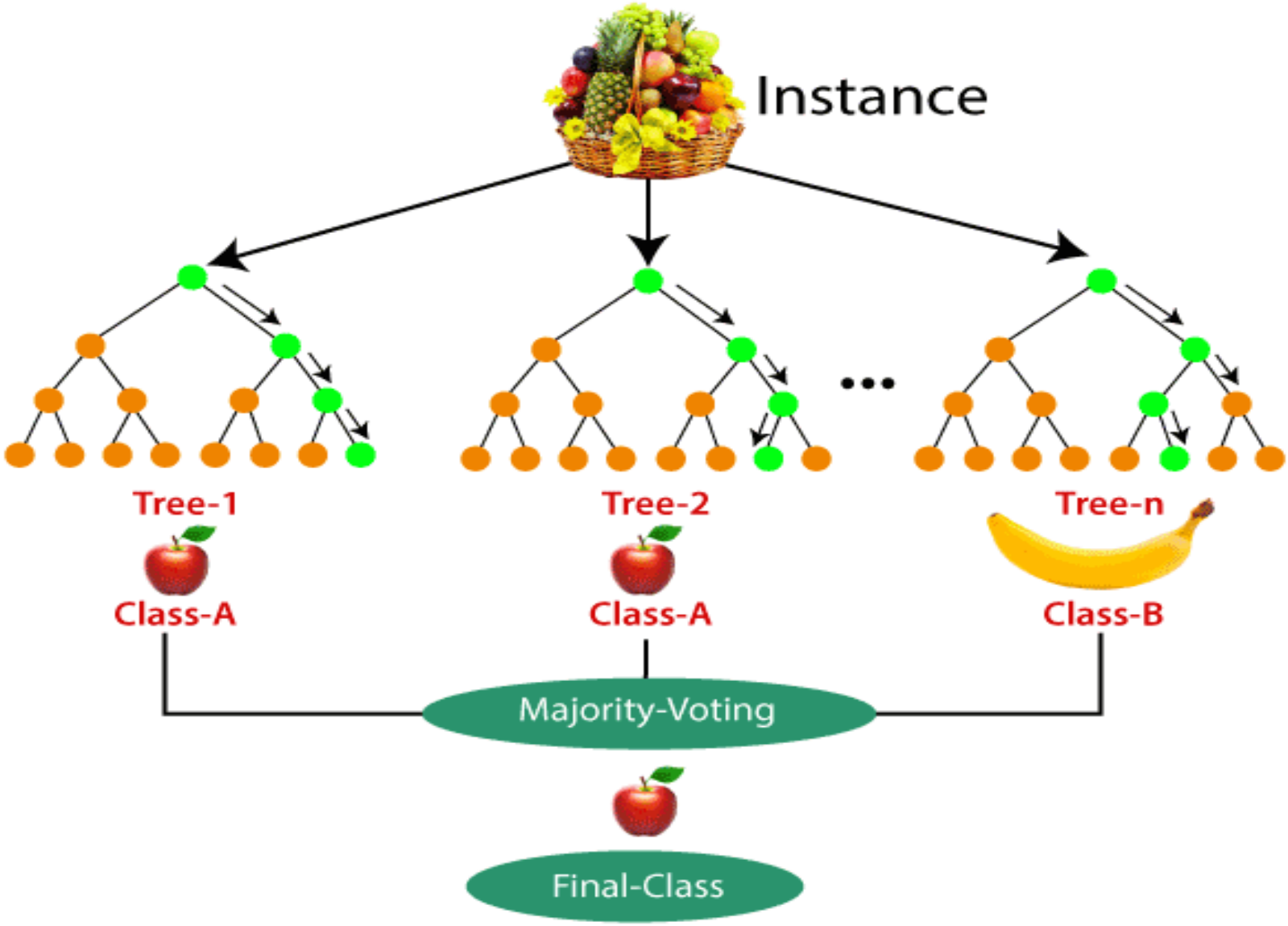
Before understanding the working of the random forest we must look into the ensemble technique. ***Ensemble*** simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

Ensemble uses two types of methods:

1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

2. **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST



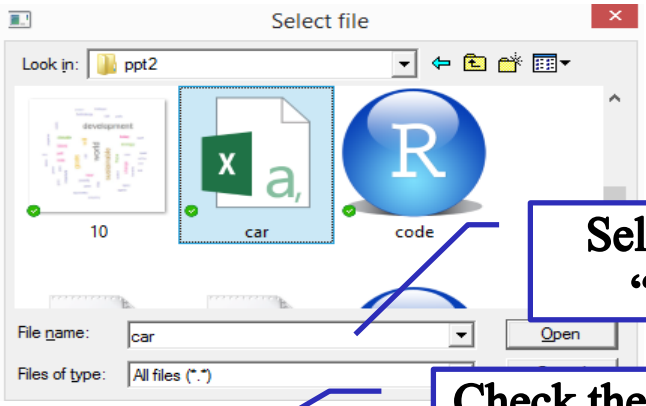

Instance
Tree-1
Tree-2
Tree-n
Class-A
Class-A
Class-B
Majority-Voting
Final-Class

Random Forest

Import t

```
> data<- read.csv(file.choose(), header = TRUE)
```

It enables you to select the dataset file from your folder



Select the file "car.csv"

data= read.csv("car.csv")

Check the internal structure of dataset

```
> str(data)
'data.frame':   1728 obs. of  7 variables:
 $ BuyingPrice: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Maintenance: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ NumDoors   : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 ...
 $ NumPersons : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 1 2 ...
 $ BootSpace  : Factor w/ 3 levels "big","med","small": 3 3 3 2 2 2 1 1 1 3 ...
 $ Safety     : Factor w/ 3 levels "high","low","med": 2 3 1 2 3 1 2 3 1 2 ...
 $ Condition  : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3 3 ...
>
```

All variables are factors with different number of levels

```
> summary(data)
 BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety    Condition
 high :432   high :432   2    :432   2   :576   big  :576   high:576   acc  : 384
 low  :432   low  :432   3    :432   4   :576   med  :576   low :576   good :  69
 med  :432   med  :432   4    :432   more:576   small:576   med :576   unacc:1210
 vhigh:432   vhigh:432   5more:432                                     vgood:  65
>
```

Check the summary of the dataset

Lets divide the dataset into training and validation dataset in 7:3 ratio. The model will be trai

```
> set.seed(100)
> train <- sample(nrow(data), 0.7*nrow(data), replace = FALSE)
>
```

In order to select 70 percent data points from dataset a random sample of size 0.7* nrow(data) is created

```
> TrainSet <- data[train,]
```

TrainSet contains rows selected through random sample

```
> ValidSet <- data[-train,]
```

ValidSet contains remaining rows of dataset

```
> summary(TrainSet)
 BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety    Condition
 high :313   high :287   2    :305   2   :406   big  :416   high:396   acc  :264
 low  :292   low  :317   3    :300   4   :399   med  :383   low :412   good : 52
 med  :305   med  :303   4    :295   more:404   small:410   med :401   unacc:856
 vhigh:299   vhigh:302   5more:309                                     vgood: 37
```

Summary function can be used to check the TrainSet and ValidSet

```
> model <- randomForest(Condition ~ ., data = TrainSet, importance = TRUE)
```

model <- randomForest(as.factor(Condition) ~ ., data = TrainSet)

```
model1 <- randomForest(Condition ~ BuyingPrice+Maintenance + NumDoors + NumPersons+BootSpace+
Safety, data = TrainSet, importance = TRUE)
```

Equation of the variables. Here ~ . Indicates that all variables are taken as predictors

Model assesses importance of predictor variables

In case, you want to manually select predictor variables then their names should be mentioned in equation form

In above mentioned models, we have not specified number of trees to be trained and number of predictors to be taken at a time for training.

Default value for number of trees is 500 and number of pre

```
Call:
 randomForest(formula = Condition ~ ., data = TrainSet, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 3.47%
Confusion matrix:
      acc good unacc vgood class.error
acc   252    7     5     0  0.04545455
good    2   44     1     5  0.15384615
unacc  15    1   840     0  0.01869159
vgood   6    0     0    31  0.16216216
```

Error rate is 3.47 percent

```
randomForest(formula = as.factor(Condition) ~ ., data = TrainSet)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4.22%
Confusion matrix:
      acc good unacc vgood class.error
acc   250    3    11     0  0.05303030
good   16   33     0     3  0.36538462
unacc  14    0   842     0  0.01635514
vgood   4    0     0    33  0.10810811
```

```
> model2 <- randomForest(Condition ~ ., data = TrainSet, ntree = 500, mtry = 6,
 importance = TRUE)
```

Number of trees and predictors can be specified

```
> model2

Call:
 randomForest(formula = Condition ~ ., data = TrainSet, ntree = 500,      mtry
= 6, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 6

        OOB estimate of  error rate: 2.4%
Confusion matrix:
       acc good unacc vgood class.error
acc    255    4     5     0  0.03409091
good     2   47     1     2  0.09615385
unacc   12    1   843     0  0.01518692
vgood    1    1     0    35  0.05405405
```

On comparing the summary of this model2 with previous model, we can infer that as the number of predictor valriables is increased to 6, the error

```
> predTrain <- predict(model2, TrainSet, type = "class")
```

Predicting on training dataset using model2 which is having lower error rate

```
> table(predTrain, TrainSet$Condition)

predTrain acc good unacc vgood
    acc   264    0     0     0
    good    0   52     0     0
    unacc   0    0   856     0
    vgood   0    0     0    37
```

Actual and predicted classes are put together to check classification accuracy

Table indicates that all data points are classified accurately

```
> predValid <- predict(model2, ValidSet, type = "class")
```

```
> table(predValid,ValidSet$Condition)

predValid acc good unacc vgood
    acc   117    0     2     0
    good    1   17     0     0
    unacc   1    0   352     0
    vgood   1    0     0    28
```

```
> mean(predValid == ValidSet$Condition)
[1] 0.9903661
```

```
> importance(model2)
                acc     good     unacc    vgood MeanDecreaseAccuracy
BuyingPrice 152.18451 79.01578  99.72550 64.21225            199.19811
Maintenance 126.77856 79.08087  96.55375 42.66366            172.94427
NumDoors     27.07687 14.82813  35.96107 16.17452             43.93335
NumPersons  156.90469 54.57009 185.46941 53.36289            229.69909
BootSpace    79.81613 63.58950  74.74986 48.46084            128.62323
Safety      171.94923 94.71288 199.70164 86.04295            267.48905
            MeanDecreaseGini
BuyingPrice         71.82806
Maintenance         90.23912
NumDoors            31.62115
NumPersons         126.16119
BootSpace           71.99321
Safety             149.15912
```

Predictions have been made on validation dataset

Predicted and actual classes are put together to check accuracy

Table indicates that some of the data points are misclassified

Checks average accuracy of prediction by comparing actual and predicted class. It indicates 99.03% accuracy

Accuracy is how much accuracy decreases on excluding a particular variable from tree

calculated based on the reduction in sum of squared errors whenever a variable is chosen to split. Safety variable is most important

Decision tree implementation

Install three packages: “rpart”, “caret”, “e1071”

Call these libraries:

>library(rpart)

>library(caret)

>library(e1071)

Train the decision tree model on same training dataset by using all the predictor variables at a time

```
> model_dt = train(Condition ~ ., data = TrainSet, method = "rpart")
```

Predicting on training dataset by using trained decision tree model

```
> model_dt_1 = predict(model_dt, data = TrainSet)
```

Compare predicted class with the actual class

```
> table(model_dt_1, TrainSet$Condition)

model_dt_1 acc good unacc vgood
     acc   241    52   132    37
     good    0     0     0     0
     unacc  23     0   724     0
     vgood   0     0     0     0
```

Average prediction accuracy is calculated. The model accuracy is 79.8 percent.

```
> mean(model_dt_1 == TrainSet$Condition)
[1] 0.7981803
```

Now predict class on Validation dataset and check accuracy

```
> model_dt_vs = predict(model_dt, newdata = ValidSet)

> table(model_dt_vs, ValidSet$Condition)

model_dt_vs acc good unacc vgood
      acc   107   17    58    28
      good    0    0     0     0
      unacc  13    0   296     0
      vgood   0    0     0     0

> mean(model_dt_vs == ValidSet$Condition)
[1] 0.7764933
```

Now predict the 'condition' of cars in validation dataset. This data was not used for training the original model.

Compare predicted class with the actual class of validation dataset

Accuracy of the decision tree is 77.64 percent while it was 99.03 percent in case of random forest. This difference in the prediction accuracy clearly indicates the superiority of random forest.

Predict the ‘Condition’ of car for unseen data

Here we have the values of predictor variables for 11 cars, but we do not know the ‘conditi

	A	B	C	D	E	F
1	BuyingPrice	Maintenance	NumDoors	NumPersons	BootSpace	Safety
2	vhigh	med	4	2	small	low
3	vhigh	low	3	more	med	med
4	low	med	2	2	small	high
5	high	low	3	4	med	low
6	high	vhigh	5more	2	big	med
7	vhigh	vhigh	4	2	small	high
8	vhigh	high	2	more	med	med
9	low	vhigh	2	2	small	high
10	med	low	3	4	med	low
11	vhigh	vhigh	5more	2	med	med
12	vhigh	med	2	more	big	med

Lets predict the condition of these 11 cars by using both random forest and decision tree models

```
> unseen_data<-read.csv(file.choose(), header = TRUE)
```

Load the unseen data

```
> dt<-predict(model_dt, newdata = unseen_data)
```

Predict the class by applying trained decision tree model

```
> dt
 [1] unacc acc   unacc unacc unacc unacc acc   unacc unacc unacc acc
Levels: acc good unacc vgood
```

Predicted classes for the 11 cars

```
> rf<-predict(model2, unseen_data, type = "class")
```

Predict class by applying trained random forest on unseen data

```
> rf
    1     2     3     4     5     6     7     8     9    10    11
unacc   acc unacc unacc unacc unacc unacc unacc unacc unacc   acc
Levels: acc good unacc vgood
```

Classes predicted by Random forest model

Both the models have predicted same class for majority of the cases except one case. As we have seen the random forest model shown better accuracy on validation data so we can rely on its prediction with more confidence.
We cannot check the accuracy of this prediction as we do not know the actual condition of the cars.

Thank you !!!