Decision Tree

Using the tree library, I create a model. Dataset target feature “type” was except before creating model. I use 80% of samples for training, and 20% of data for testing.

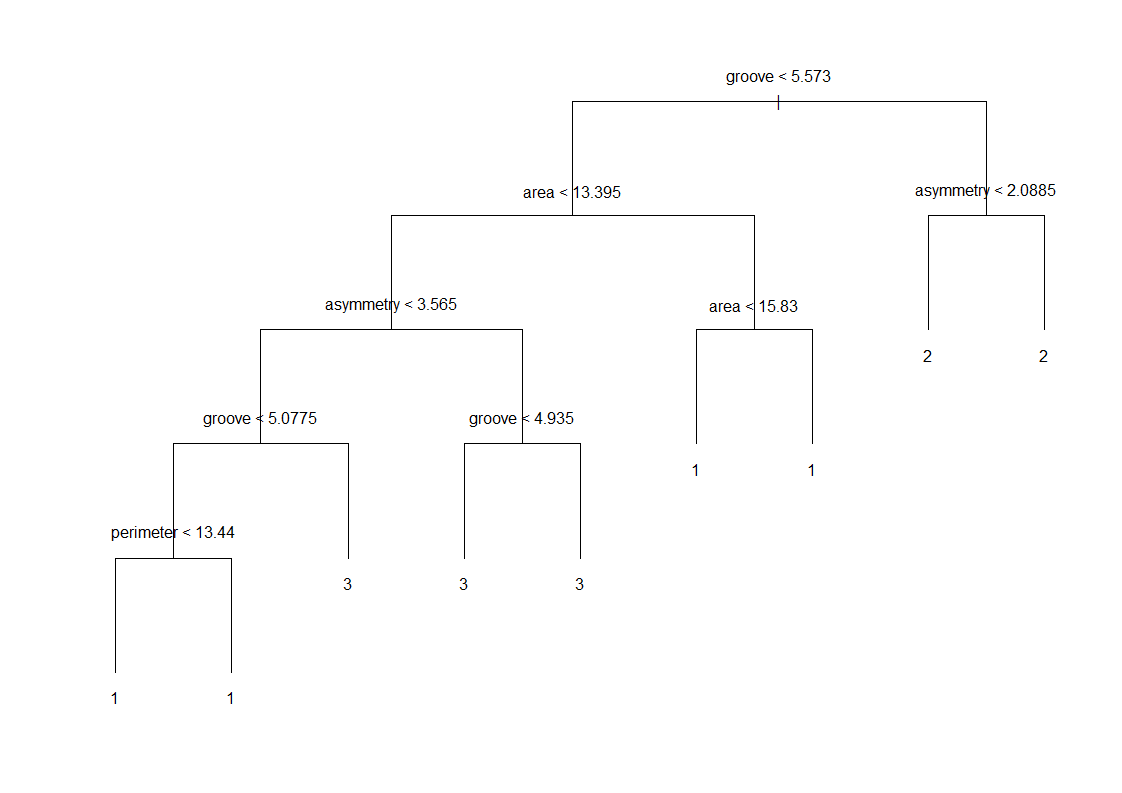


Figure 1 The first DT without subset

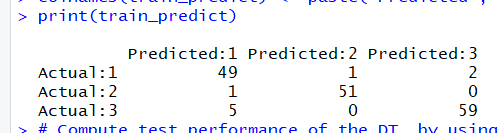


Figure 2 Training Predict

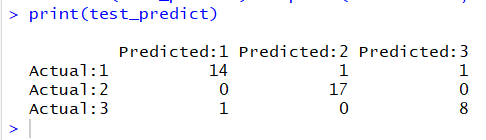
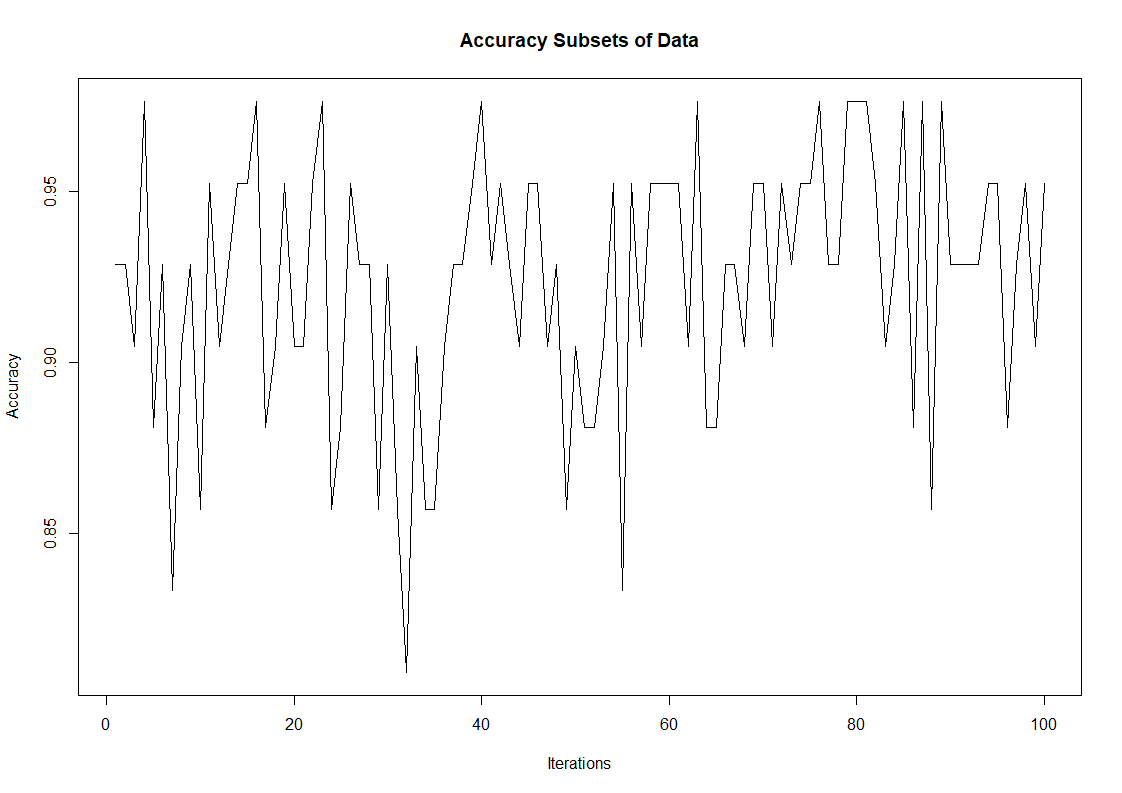


Figure 3 Test Predict

Accuracy of Cross Validation version DT



Maximum accuracy is **0.9230952**

Highest accuracy’s DT is below.

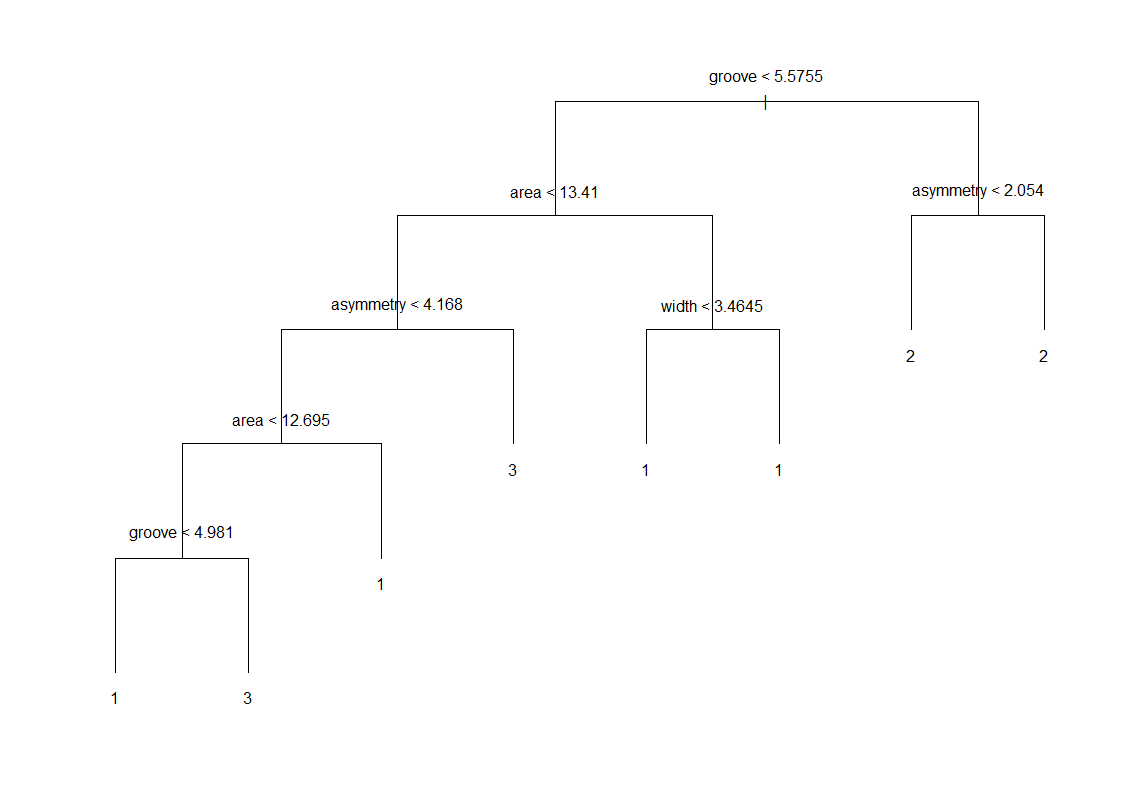
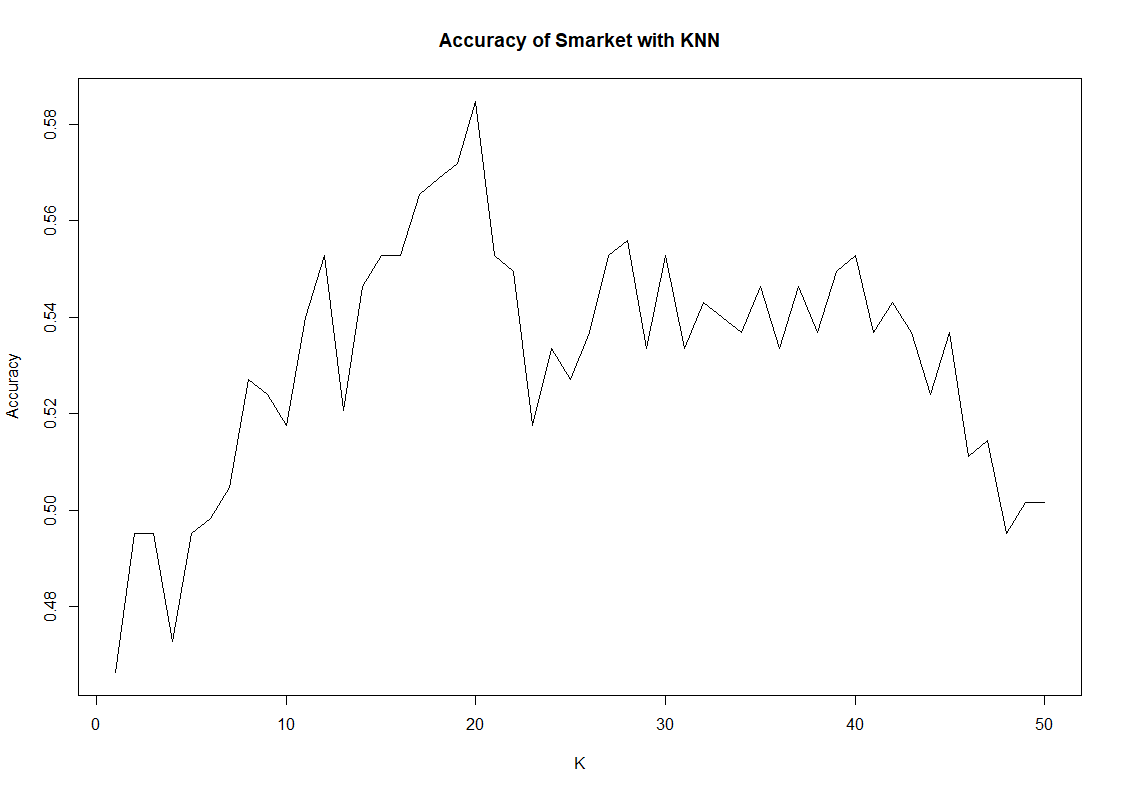


Figure 4 Highest accuracy’s DT

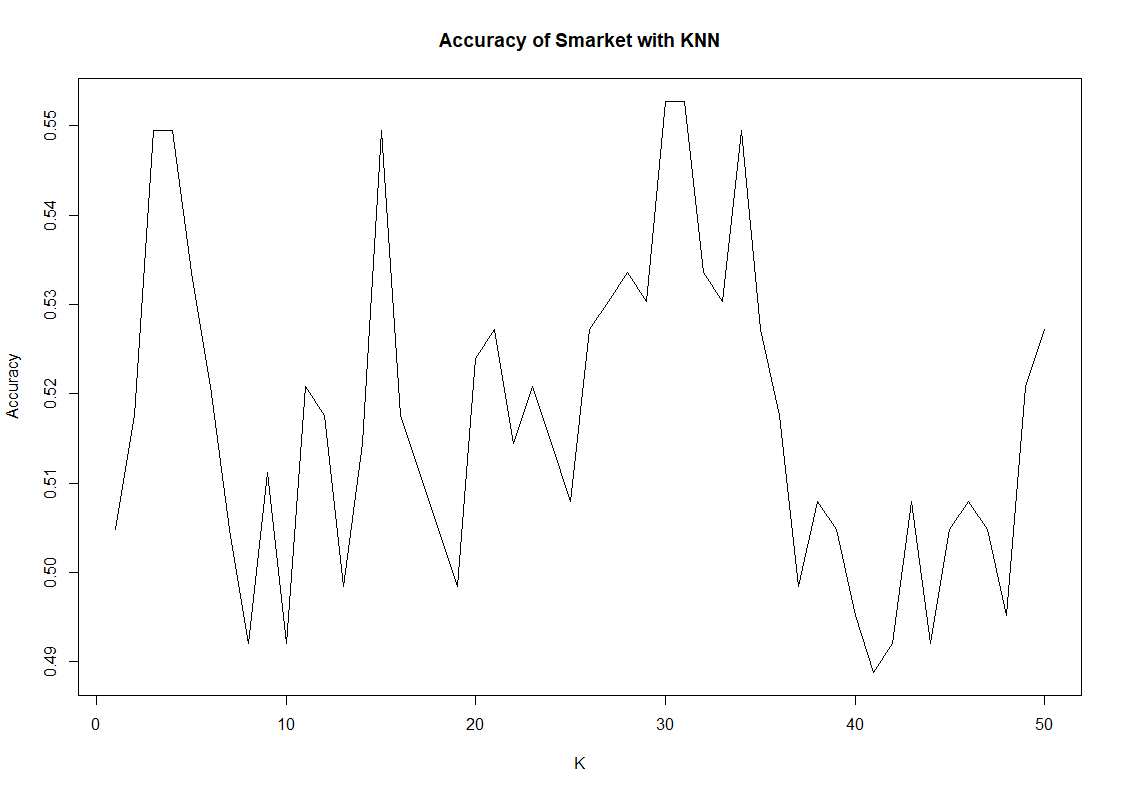
KNN

Knn Combinations;

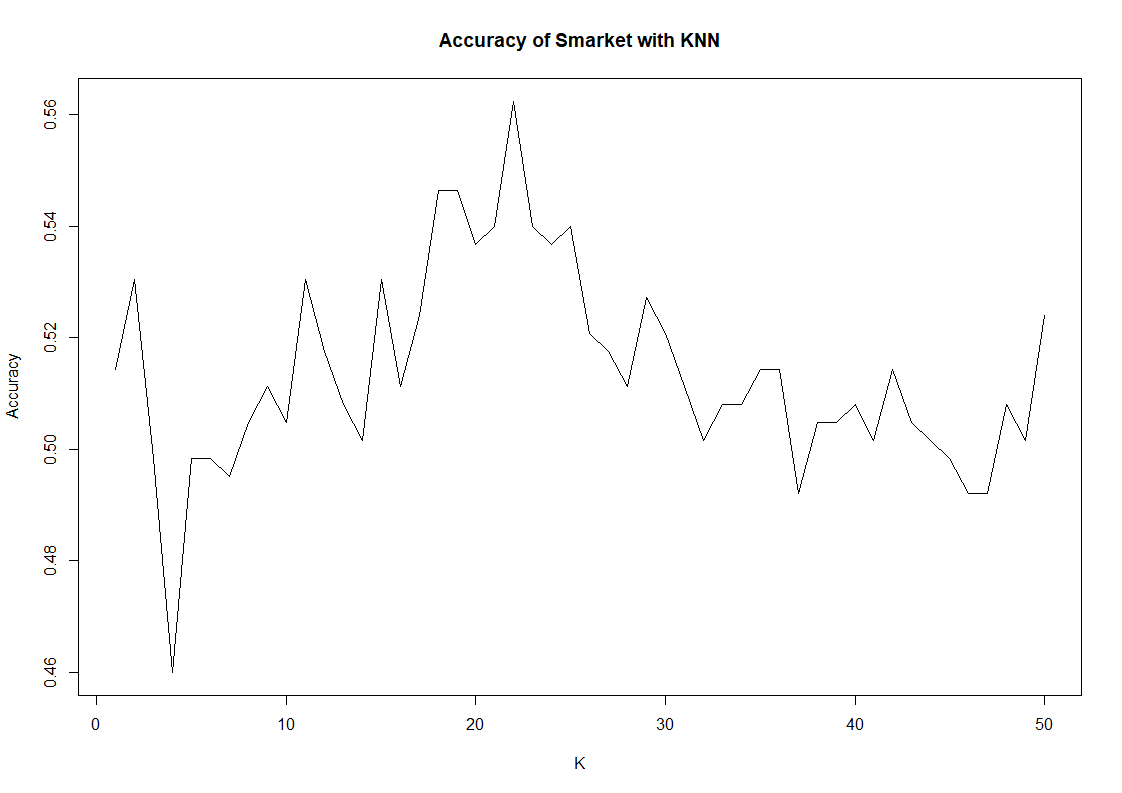
1. Lag1 Lag2 Lag3



1. Lag3 Lag4 Lag5



1. Lag1 Lag2 Lag3 Lag4 Lag5



Which k value did provide the highest accuracy for each combination? Report both k values and the highest accuracy value.

|  |  |  |
| --- | --- | --- |
| Combination | Max Accuracy | K-value |
| 1 | 0.5846645 | 20 |
| 2 | 0.5527157 | 31 |
| 3 | 0.5623003 | 22 |

What is the highest accuracy to predict the “Direction” column when you set a specific k value and the descriptive feature combination?

k-value : 20

Accuracy: 0.5846645

Combination: 1

R Code

# Decision Tree Tutorial on Iris Data Set

library(tree) # Contains the "tree" function

dataSet <- read.delim(file = "wheat\_types.txt", sep = ";")

set.seed(551235) #Set the seed for reproducibility

#first DT

dt <- tree(as.factor(type) ~ ., data = dataSet, split = "deviance")

summary(dt)

misclass.tree(dt)

#Use 80% of samples for training and 20% of them for test purposes

train <- sample(1:nrow(dataSet), size=nrow(dataSet)\*0.8)

dt2 <- tree(as.factor(type) ~ . -type, data = dataSet, subset = train)

# plot final DT

plot(dt2, type = "uniform")

text(dt2)

# Compute training performance of the DT by using only training samples (their indices were saved in the "sub" vector)

train\_predict <- table(predict(dt2, dataSet[train, ], type = "class"), dataSet[train, "type"])

rownames(train\_predict) <- paste("Actual", rownames(train\_predict), sep = ":")

colnames(train\_predict) <- paste("Predicted", colnames(train\_predict), sep = ":")

print(train\_predict)

# Compute test performance of the DT by using only test samples

test\_predict <- table(predict(dt2, dataSet[-train, ], type = "class"), dataSet[-train, "type"])

rownames(test\_predict) <- paste("Actual", rownames(test\_predict), sep = ":")

colnames(test\_predict) <- paste("Predicted", colnames(test\_predict), sep = ":")

print(test\_predict)

dt2

#Cross-validation version - Construct a new DT for different partitions of the samples - 100 times

dt\_acc <- numeric()

set.seed(2561850)

max = 0.0

dtMax = NULL # for finding DT of the best accuracy

for(i in 1:100){

temp\_train <- sample(1:nrow(dataSet), size=nrow(dataSet)\*0.8)

fit2 <- tree(as.factor(type) ~ .-type, data = dataSet, subset = temp\_train)

test\_predict <- table(predict(fit2, dataSet[-temp\_train, ], type = "class"), dataSet[-temp\_train, "type"])

accuracy = sum(diag(test\_predict)) / sum(test\_predict)

# find the best accuracy

if(accuracy >= max){

max = accuracy

dtMax = fit2

}

dt\_acc <- c(dt\_acc, sum(diag(test\_predict)) / sum(test\_predict))

}

# average accuracy

mean(dt\_acc)

# plot all accuracys

plot(dt\_acc, type="l", ylab="Accuracy", xlab="Iterations", main="Accuracy Subsets of Data")

# plot error rates

plot(1-dt\_acc, type="l", ylab="Error Rate", xlab="Iterations", main="Error Rate for our dataset With Different Subsets of Data")

# What is the average perfomance of all DTs?

# plot final DT

plot(dtMax, type = "uniform")

text(dtMax)

#=========================================================================

#=========================================================================

#=========================================================================

# kNN Tutorial on Iris Data Set

library(class) # Contains the "knn" function

library(ISLR)

set.seed(5910401) #Set the seed for reproducibility

#Create partitions in the Iris data set (75% for training, 25% for testing/evaluation)

Smarket\_sample <- sample(1:nrow(Smarket), size=nrow(Smarket)\*0.75)

Smarket\_train <- Smarket[Smarket\_sample, ] #Select the 75% of rows

Smarket\_test <- Smarket[-Smarket\_sample, ] #Select the 25% of rows

#First try to determine the right K-value

Smarket\_acc <- numeric() #holding variable

combinations <- list(2:4, 3:5, 2:6)

max\_acc <- NULL #find the maximum accuracy that is possible scenery of all combinations

list\_acc <- NULL #for plot combinations accuracy

for(comb in 1:3){

maxAccuracy = 0

maxKValue = 0

for(i in 1:50){

#Apply knn with k = i

predict <- knn(train=Smarket\_train[,combinations[[comb]]], test=Smarket\_test[,combinations[[comb]]], cl=Smarket\_train$Direction, k=i)

tempAccuracy = mean(predict==Smarket\_test$Direction)

Smarket\_acc <- c(Smarket\_acc, tempAccuracy)

if(tempAccuracy >= maxAccuracy){

maxAccuracy = tempAccuracy

maxKValue = i

}

}

print(maxAccuracy)

print(maxKValue)

max\_acc <- c(max\_acc, list(maxAccuracy, maxKValue))

list\_acc <- c(list\_acc, list(Smarket\_acc))

Smarket\_acc <- NULL

}

#determine which combination is the best accuracy

max = 0

maxID = 0

for (a in 1:length(max\_acc)) {

if(a %% 2 == 1){ # accuracy

if(max\_acc[[a]] >= max){

max = max\_acc[[a]]

maxID = a

}

}

}

#plot accuracys of combination1

plot(list\_acc[[1]], type="l", ylab="Accuracy", xlab="K", main="Accuracy of Smarket with KNN")

#plot accuracys of combination2

plot(list\_acc[[2]], type="l", ylab="Accuracy", xlab="K", main="Accuracy of Smarket with KNN")

#plot accuracys of combination3

plot(list\_acc[[3]], type="l", ylab="Accuracy", xlab="K", main="Accuracy of Smarket with KNN")

# Which K-value did provide the best performance ?

print(c("The maximum accuracy is ", max\_acc[[maxID]], " and k-value is ", max\_acc[[maxID+1]]))