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.

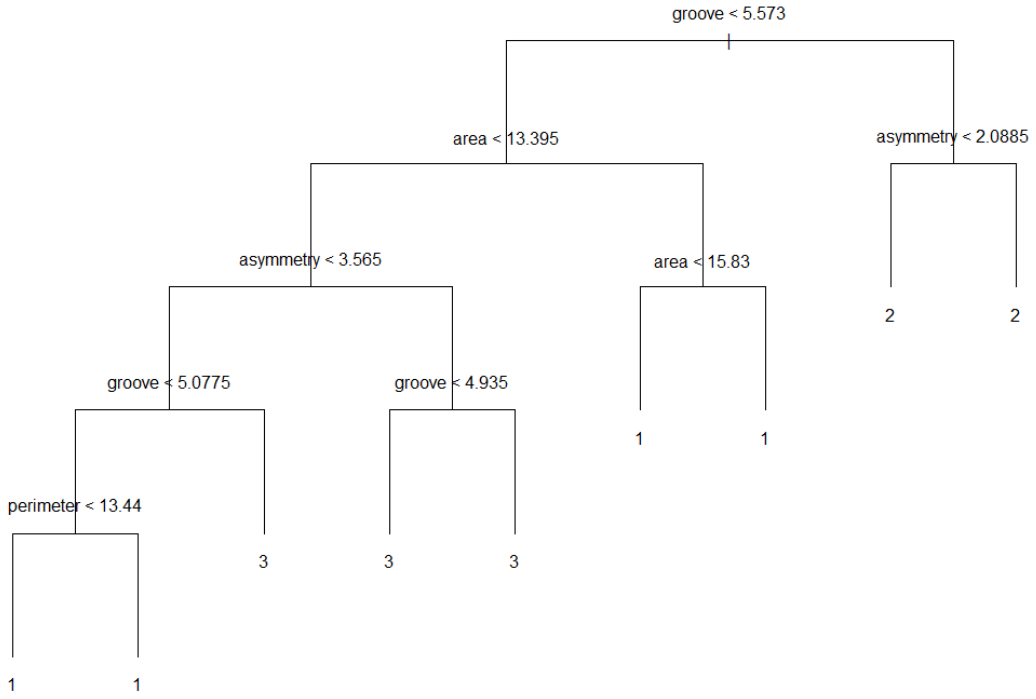


Figure 1 The first DT without subset

```
> print(train_predict)

         Predicted:1 Predicted:2 Predicted:3
Actual:1          49           1           2
Actual:2           1          51           0
Actual:3           5           0          59
> # Compute test performance of the DT  by using
```

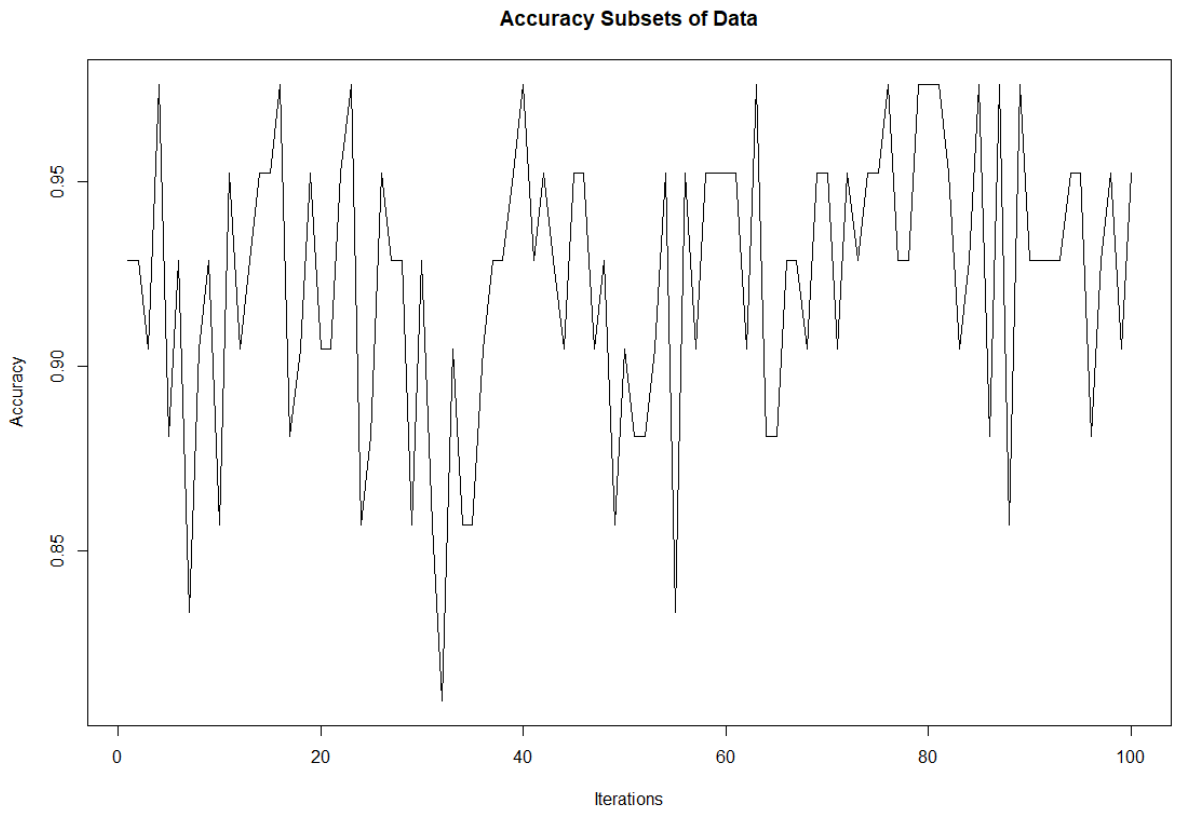
Figure 2 Training Predict

```
> print(test_predict)

          Predicted:1 Predicted:2 Predicted:3
Actual:1           14           1           1
Actual:2            0          17           0
Actual:3            1           0           8
>
```

Figure 3 Test Predict

Accuracy of Cross Validation version DT



`Maximum accuracy is` **0.9230952**

Highest accuracy's DT is below.

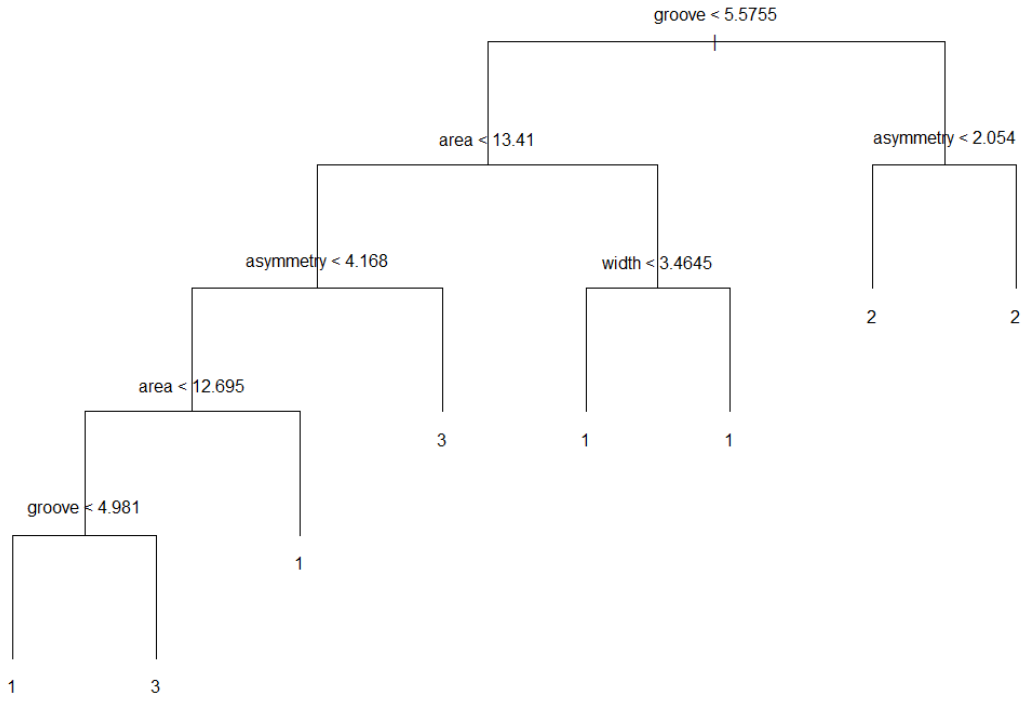
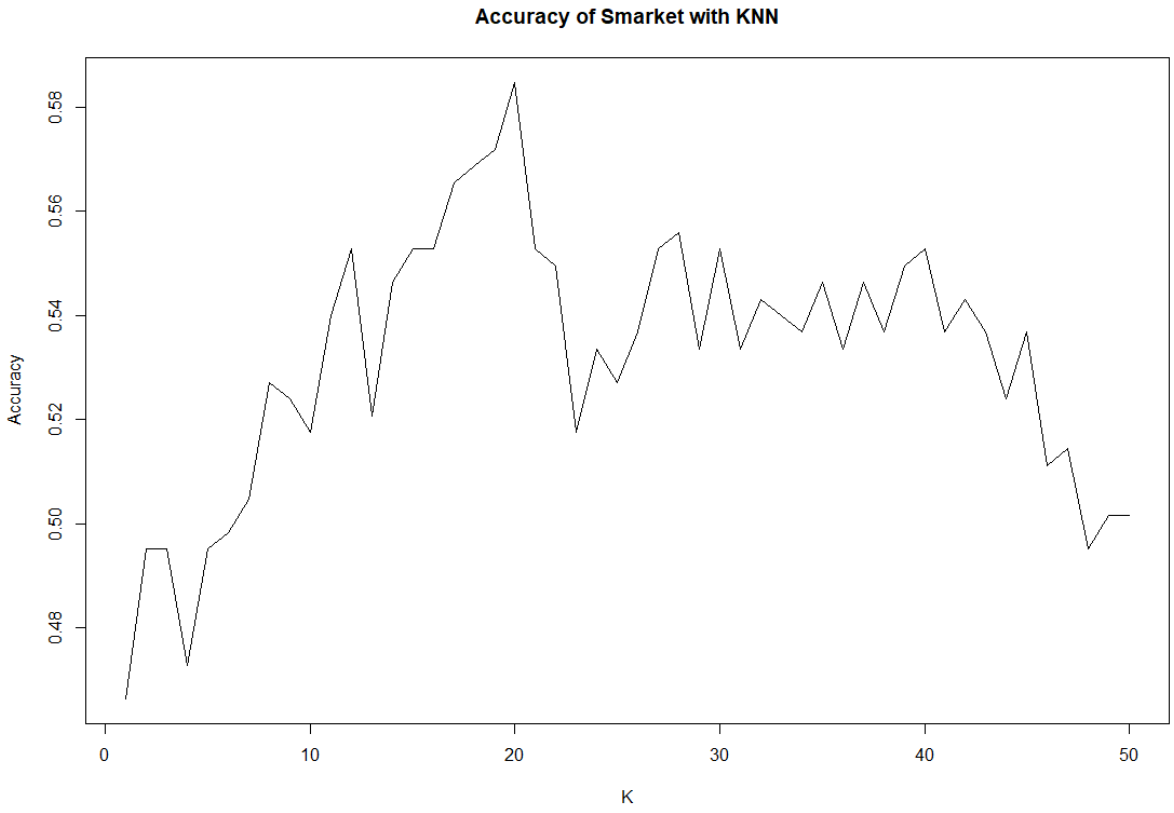


Figure 4 Highest accuracy's DT

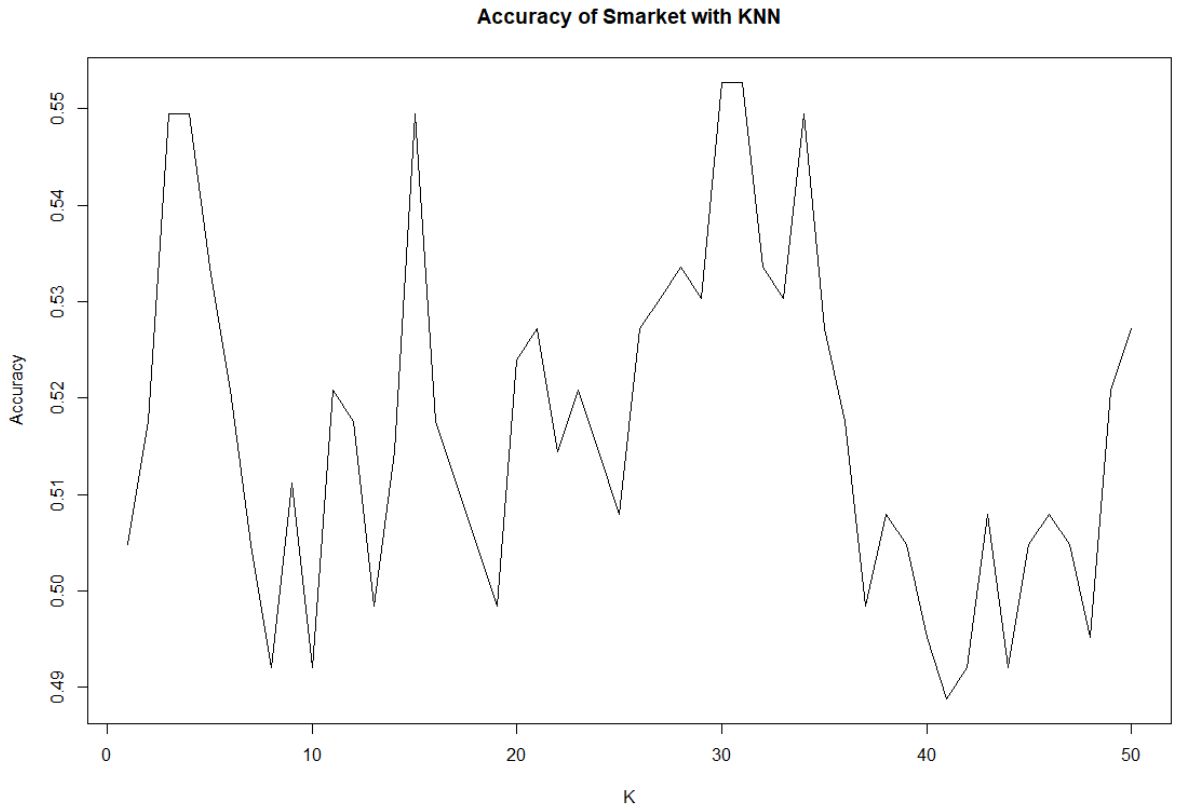
KNN

Knn Combinations;

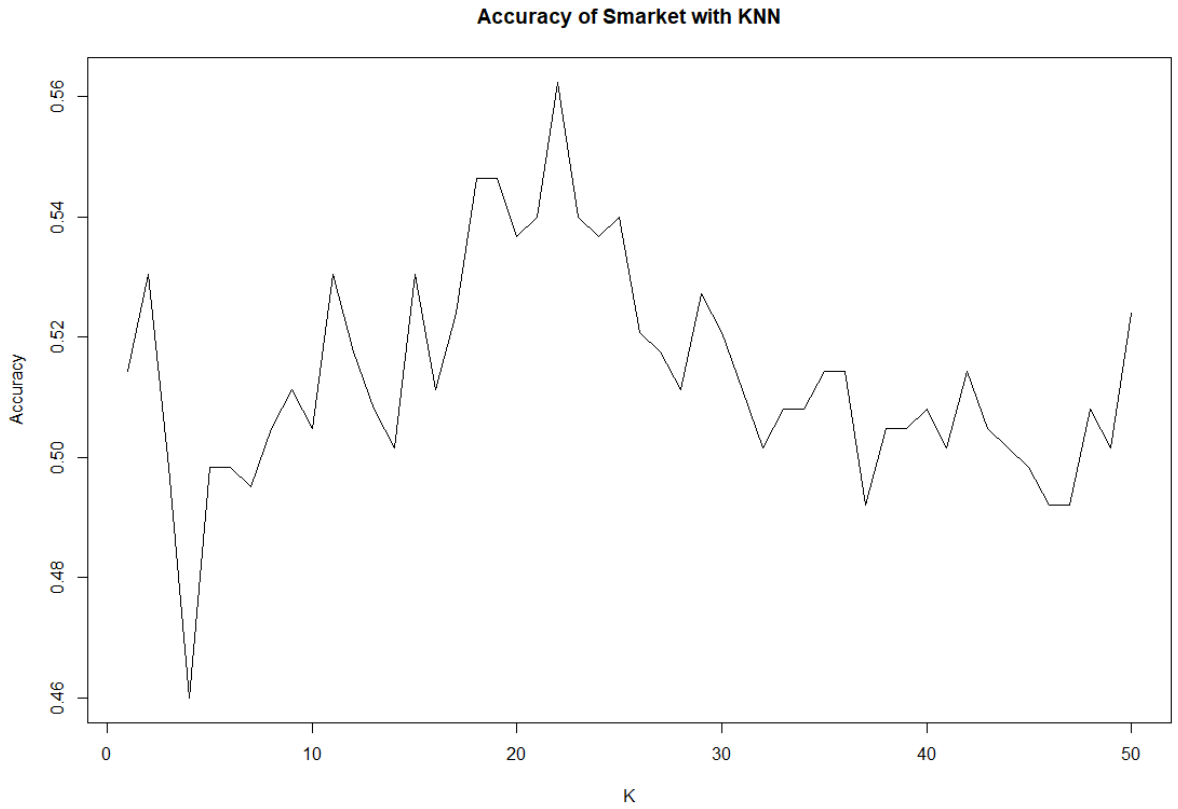
1- Lag1 Lag2 Lag3



2- Lag3 Lag4 Lag5



3- 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]]))
```