Classification Problem

Verena

31 3 2022

```
knitr::opts_chunk$set(echo = TRUE)
```

The IRIS dataset

The iris dataset is a built-in dataset in R that contains measurements on 4 different attributes (in centimeters) for 50 flowers from 3 different species.

https://www.kaggle.com/code/vinayshaw/iris-species-100-accuracy-using-naive-bayes/notebook

```
data(iris)
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

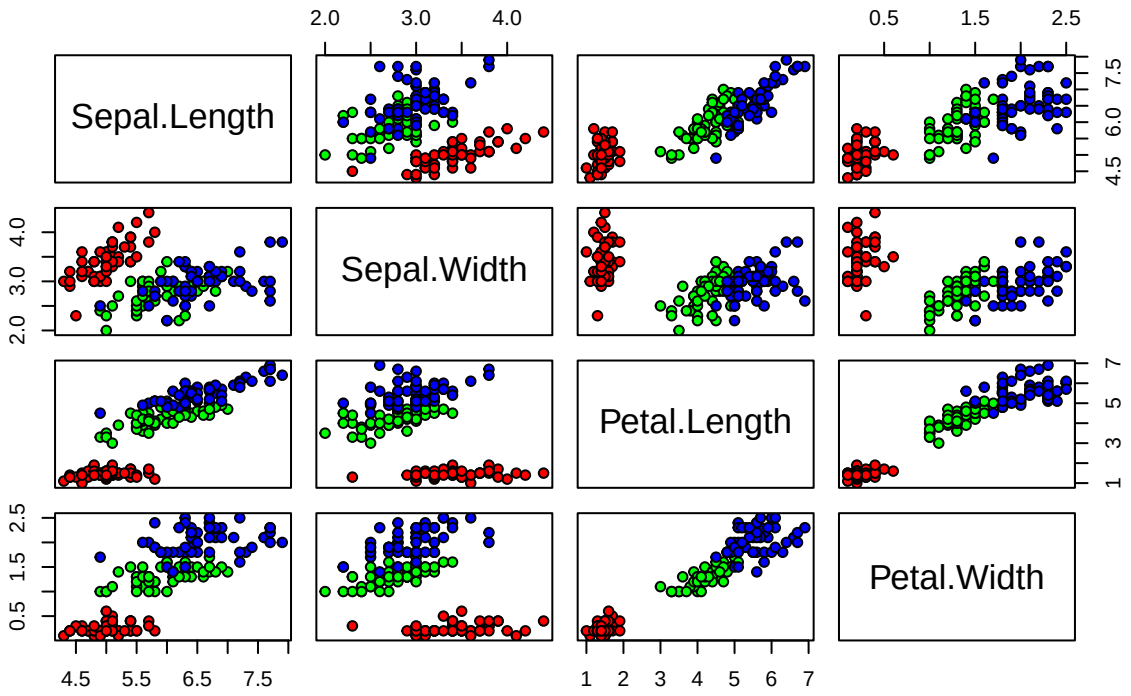
```
dim(iris)
```

```
## [1] 150   5
```

```
summary(iris)
```

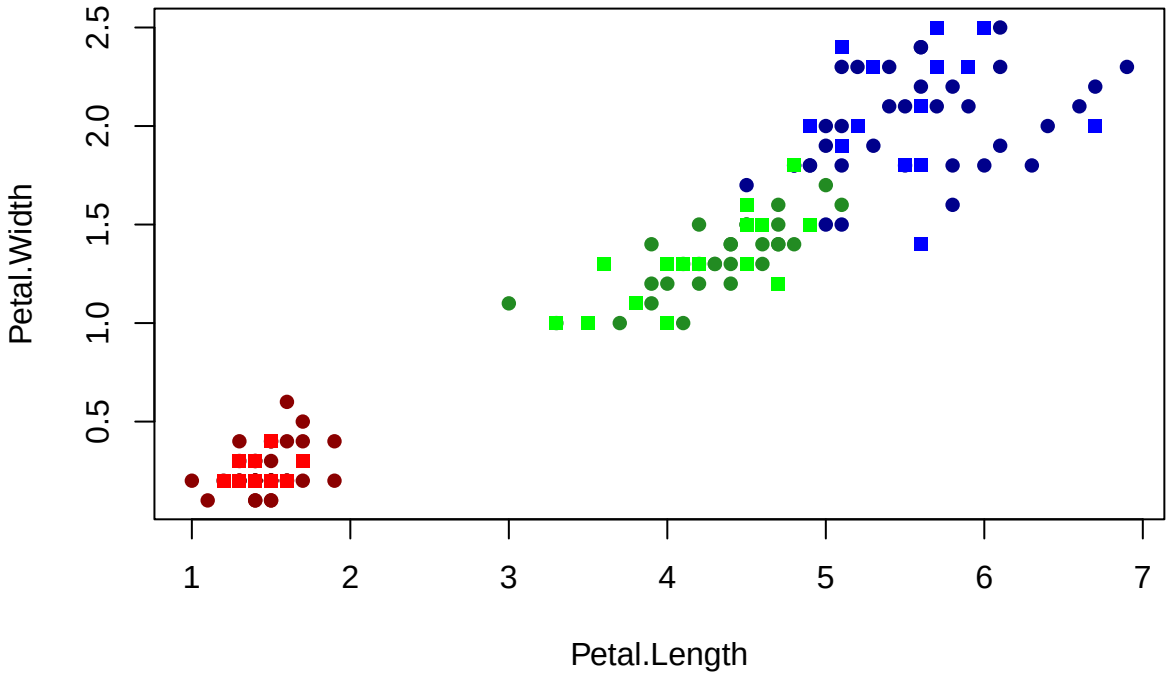
```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
pairs(iris[1:4],main="Iris Data(red=setosa,green=versicolor,blue=virginica)", pch=21,bg=c("red","green"
```




```
# Determine sample size
set.seed(3)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33))

# Split the `iris` data
iris.training <- iris[ind==1, 1:4]
iris.test <- iris[ind==2, 1:4]

# Split the class attribute
iris.trainingtarget <- iris[ind==1, 5]
iris.testtarget <- iris[ind==2, 5]

make.grid<-function(x,n=200){
  grange=apply(x,2,range)
  x1=seq(from=grange[1,1],to=grange[2,1],length=n)
  x2=seq(from=grange[1,2],to=grange[2,2],length=n)
  expand.grid(X1=x1,X2=x2)
}

xtrain<-iris.training[,3:4]
xtest<-iris.test[,3:4]
ytest<-iris.testtarget
ytrain<-iris.trainingtarget
xgrid = make.grid(xtest, n = 150)
```

```
colnames(xgrid)<-c("Petal.Length", "Petal.Width")

plot(xtrain[,1],xtrain[,2],col=c("darkred","forestgreen","darkblue")[unclass(ytrain)],pch=16,xlab="Petal
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```



Naive Bayes Estimation

```
library("e1071")
model_nb<-naiveBayes(iris.training, iris.trainingtarget)
table(predict(model_nb, iris.test), iris.testtarget,  dnn=list('predicted','actual'))
```

```
##             actual
## predicted    setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0         20         1
##   virginica       0          1        14
```

```
model_nb$apriori
```

```
## iris.trainingtarget
##     setosa versicolor  virginica
##         39         29         35
```

```
model_nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = iris.training, y = iris.trainingtarget)
##
## A-priori probabilities:
## iris.trainingtarget
##     setosa versicolor  virginica
##  0.3786408  0.2815534  0.3398058
##
## Conditional probabilities:
##                    Sepal.Length
## iris.trainingtarget     [,1]      [,2]
##          setosa     4.953846 0.3135922
##          versicolor 6.041379 0.5401833
##          virginica  6.680000 0.6596791
##
##                    Sepal.Width
## iris.trainingtarget     [,1]      [,2]
##          setosa     3.433333 0.3206353
##          versicolor 2.810345 0.2845357
##          virginica  2.982857 0.3560285
##
##                    Petal.Length
## iris.trainingtarget     [,1]      [,2]
##          setosa     1.466667 0.1811271
##          versicolor 4.296552 0.4648216
##          virginica  5.560000 0.5941875
##
##                    Petal.Width
## iris.trainingtarget      [,1]      [,2]
##          setosa     0.2461538 0.1143544
##          versicolor 1.3275862 0.1830233
##          virginica  2.0028571 0.2617652
```

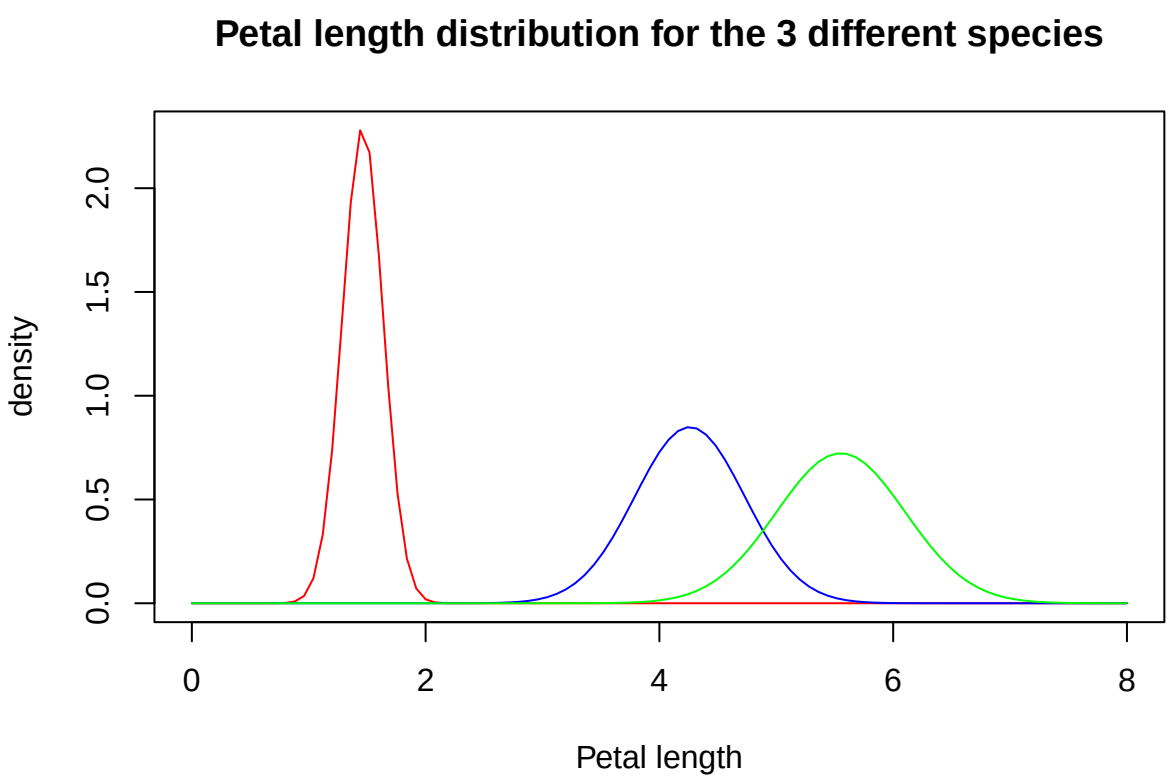
```
model_nb$tables$Petal.Length
```

```
##                    Petal.Length
## iris.trainingtarget     [,1]      [,2]
##          setosa     1.466667 0.1811271
##          versicolor 4.296552 0.4648216
##          virginica  5.560000 0.5941875
```

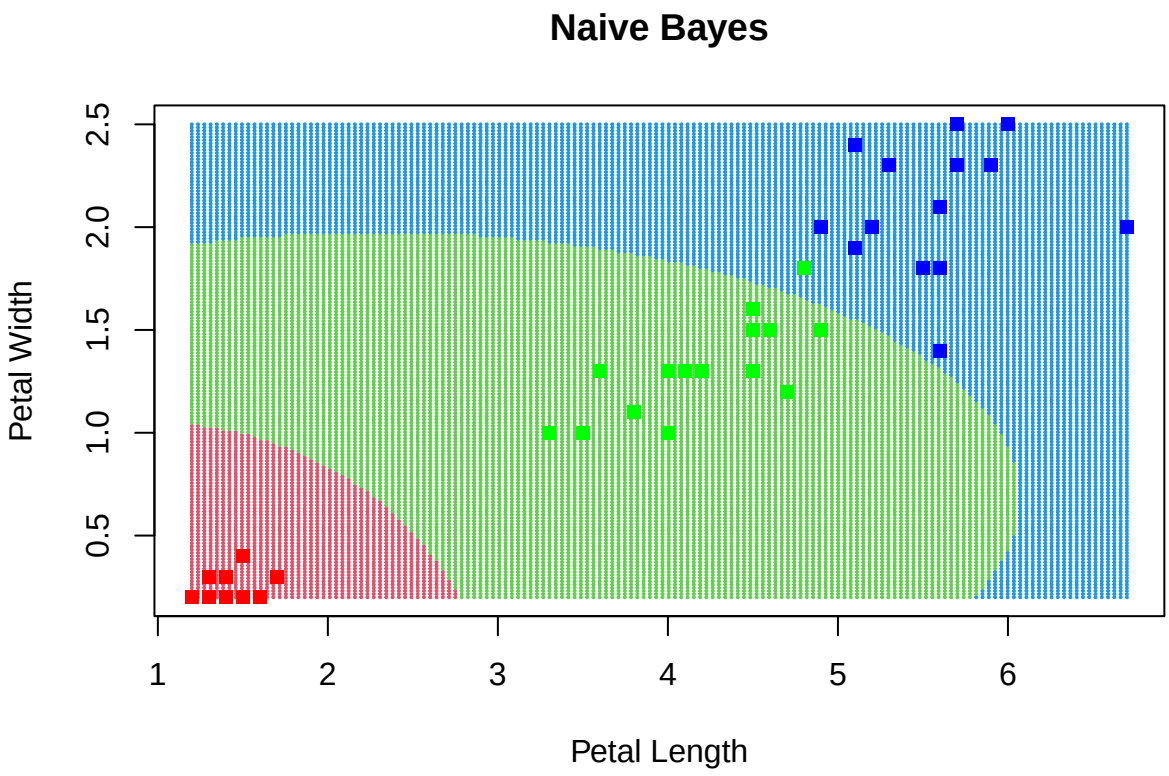
```
score_nb<-mean(predict(model_nb, iris.test)== iris.testtarget)
score_nb
```

```
## [1] 0.9574468
```

```
plot(function(x) dnorm(x, 1.462, 0.1736640), 0, 8,  col="red",xlab="Petal length", ylab="density", main=
curve(dnorm(x, 4.260, 0.4699110), add=TRUE, col="blue")
curve(dnorm(x, 5.552, 0.5518947), add=TRUE, col="green")
```

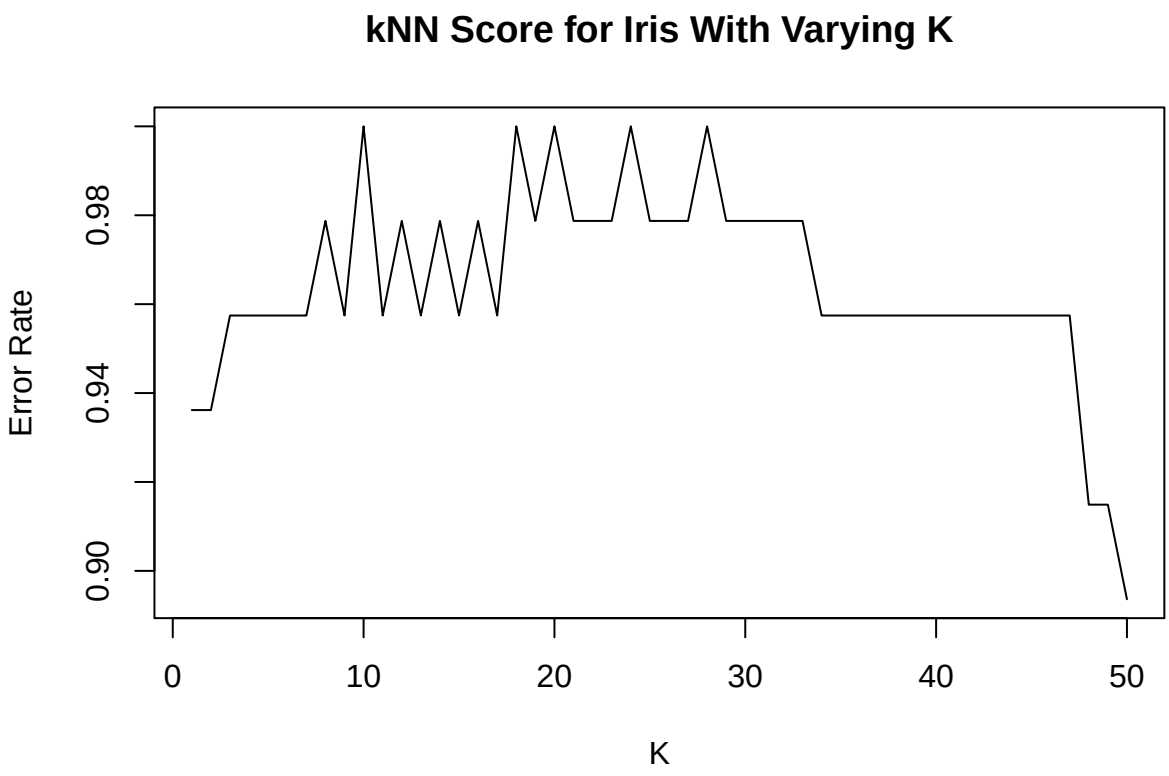



```
#Illustration
model_nb<-naiveBayes(xtrain, ytrain)
nb_x <- predict(model_nb,newdata=list("Petal.Length"=xgrid[,1], "Petal.Width"= xgrid[,2]))
plot(xgrid,col=c(2,3,4)[as.numeric(nb_x)],pch = 20,cex = .2, main = "Naive Bayes",xlab="Petal Length",y
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```

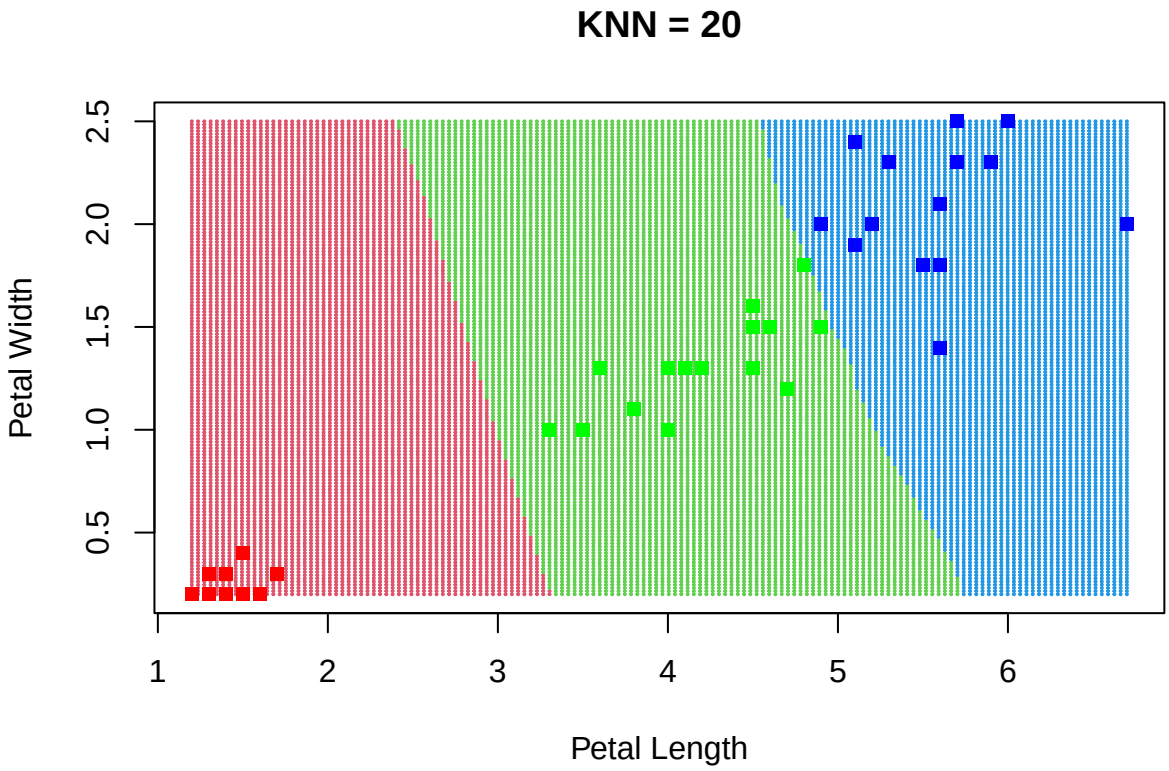


K-Nearest-Neighbors estimation

```
library(FNN) #knn regression

#First Attempt to Determine Right K####
iris_acc<-numeric() #Holding variable

for(i in 1:50){
  #Apply knn with k = i
  predict<-knn(iris.training,iris.test,
               iris.trainingtarget,k=i)
  iris_acc<-c(iris_acc,
              mean(predict==iris.testtarget))
}
#Plot k= 1 through 30
plot(iris_acc,type="l",ylab="Error Rate",
     xlab="K",main="kNN Score for Iris With Varying K")
```




```
#Illustration



k<-20
knn_x <- knn(train=xtrain,  test=xgrid , cl=factor(ytrain), k=i)
plot(xgrid,col=c(2,3,4)[as.numeric(knn_x)],pch = 20,cex = .2, main = paste("KNN =", k),xlab="Petal Leng
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```



Logistic Regression

```
library(nnet)
fit <- multinom(iris.trainingtarget ~  ., data=cbind(iris.training,iris.trainingtarget))
```

```
## # weights:  18 (10 variable)
## initial  value 113.157066
## iter  10 value 8.067276
## iter  20 value 4.933856
## iter  30 value 4.747766
## iter  40 value 4.722652
## iter  50 value 4.719503
## iter  60 value 4.719117
## iter  70 value 4.719039
## iter  80 value 4.718826
## iter  90 value 4.718567
## iter 100 value 4.718444
## final  value 4.718444
## stopped after 100 iterations
```

```
summary(fit)
```

```
## Call:
## multinom(formula = iris.trainingtarget ~ ., data = cbind(iris.training,
##     iris.trainingtarget))
```

```
##
## Coefficients:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    12.53520    -2.363754   -8.352525     9.143761   0.6613419
## virginica    -17.63227    -5.417008  -11.838608    16.440940  14.9045856
##
## Std. Errors:
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    65.45560     130.2007    150.4919     154.9567    105.1227
## virginica     65.72887     130.2188    150.5600     155.0390    105.2455
##
## Residual Deviance: 9.436889
## AIC: 29.43689
```

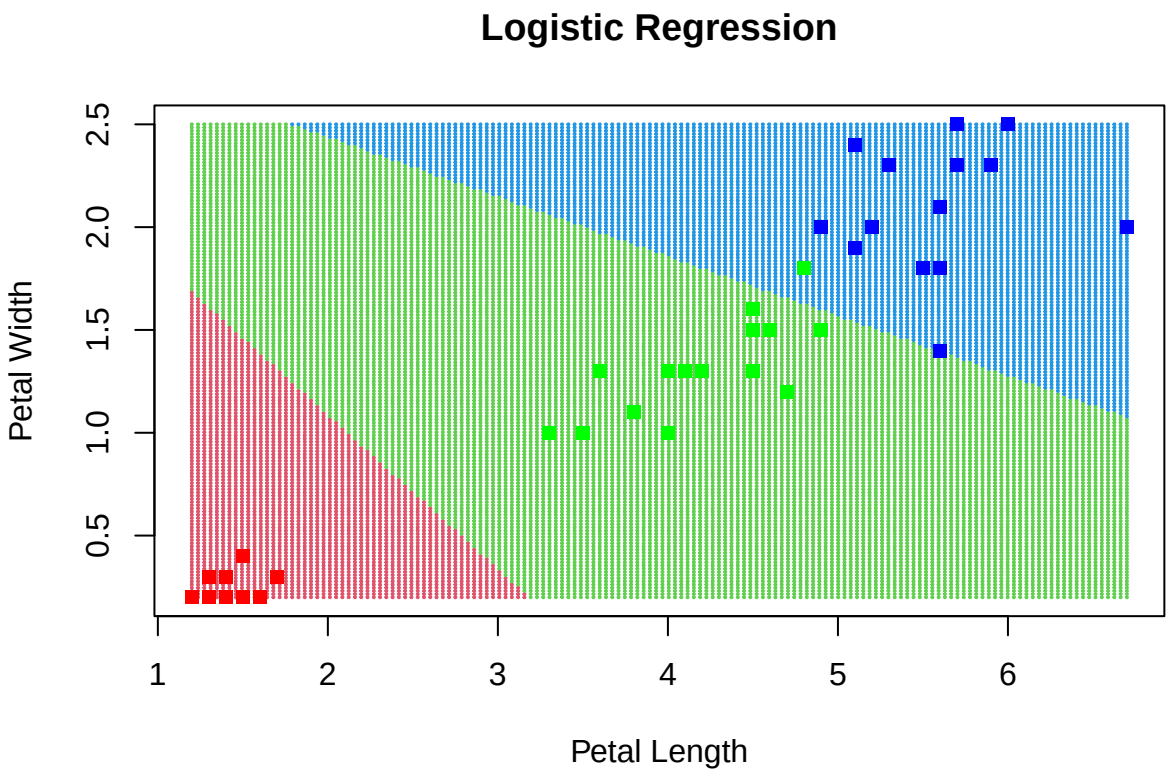
```
predicted.classes <-  predict(fit,iris.test)
head(predicted.classes)
```

```
## [1] setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

```
# Model accuracy
scorelog<-mean(predicted.classes == iris.testtarget)


log_model <- multinom(ytrain ~ ., data=cbind(xtrain))
```

```
## # weights:  12 (6 variable)
## initial  value 113.157066
## iter  10 value 9.675025
## iter  20 value 7.426201
## iter  30 value 7.384976
## iter  40 value 7.356171
## iter  50 value 7.346781
## iter  60 value 7.342964
## iter  70 value 7.340718
## iter  80 value 7.338835
## iter  90 value 7.336898
## iter 100 value 7.335650
## final  value 7.335650
## stopped after 100 iterations
```

```
log_x <- predict(log_model,newdata=xgrid)
plot(xgrid,col=c(2,3,4)[as.numeric(log_x)],pch = 20,cex = .2, main = "Logistic Regression",xlab="Petal
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```

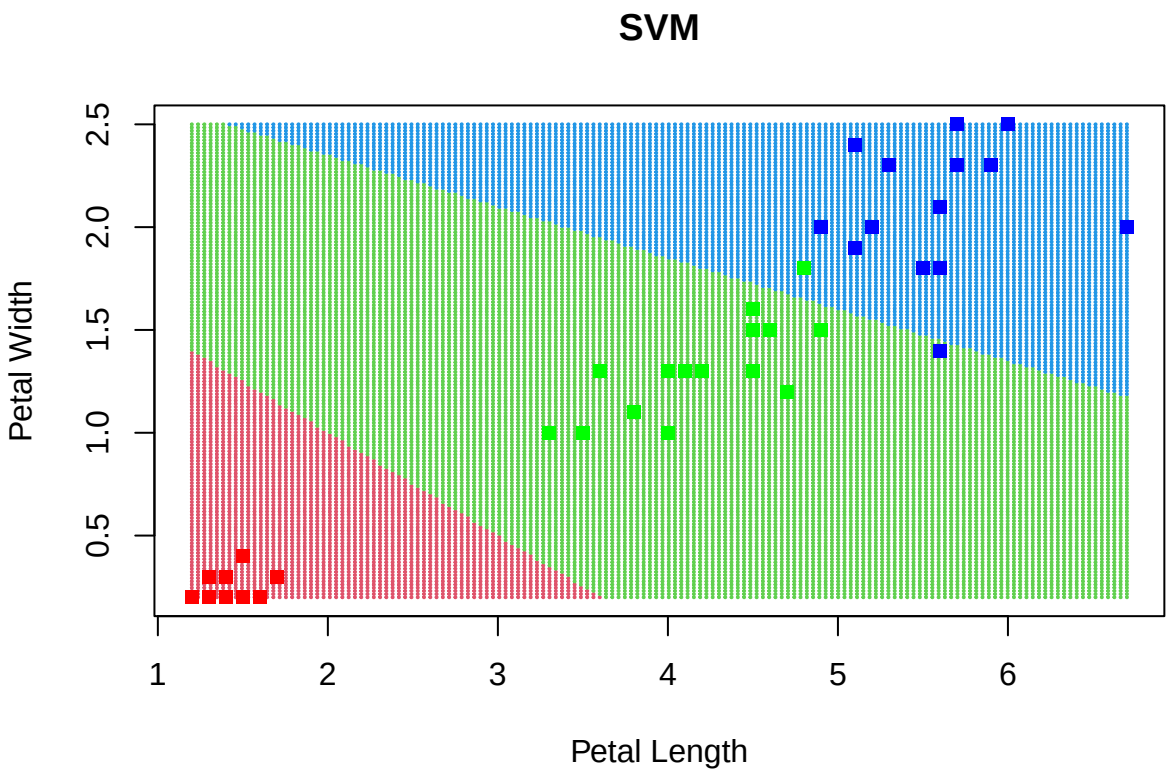


Support Vector Machine

```
library("e1071")

svm_model <- svm(iris.trainingtarget ~  ., data=cbind(iris.training,iris.trainingtarget), kernel="linea
#plot(svm_model, data=cbind(iris.training,iris.trainingtarget),Petal.Width~Petal.Length, slice = list(S
pred = predict(svm_model,iris.test)
tab<-table(iris.testtarget,pred)
score_svm<-sum(diag(tab)/sum(tab))
score_svm
```

```
## [1] 0.9787234
```

```
#Illustration
svm_model <- svm(ytrain ~ ., data=cbind(xtrain),   kernel="linear")
svm_x <- predict(svm_model,newdata=cbind(xgrid[,1],xgrid[,2]))
plot(xgrid,col=c(2,3,4)[as.numeric(svm_x)],pch = 20,cex = .2, main = "SVM",xlab="Petal Length",ylab="Pe
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```



Neural Network

```
library(neuralnet)


iris$setosa <- iris$Species=="setosa"
iris$virginica <- iris$Species == "virginica"
iris$versicolor <- iris$Species == "versicolor"
iris.training_nn <- iris[ind==1,]
iris.test_nn <- iris[(ind==2),]



library(neuralnet)
iris.net <- neuralnet(setosa+versicolor+virginica ~
                        Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                      data=iris.training_nn, hidden=c(10,10), rep = 5, err.fct = "ce", act.fct = "logis
                      linear.output = F, lifesign = "minimal", stepmax = 1000000,
                      threshold = 0.01)


#plot(iris.net, rep="best")
```

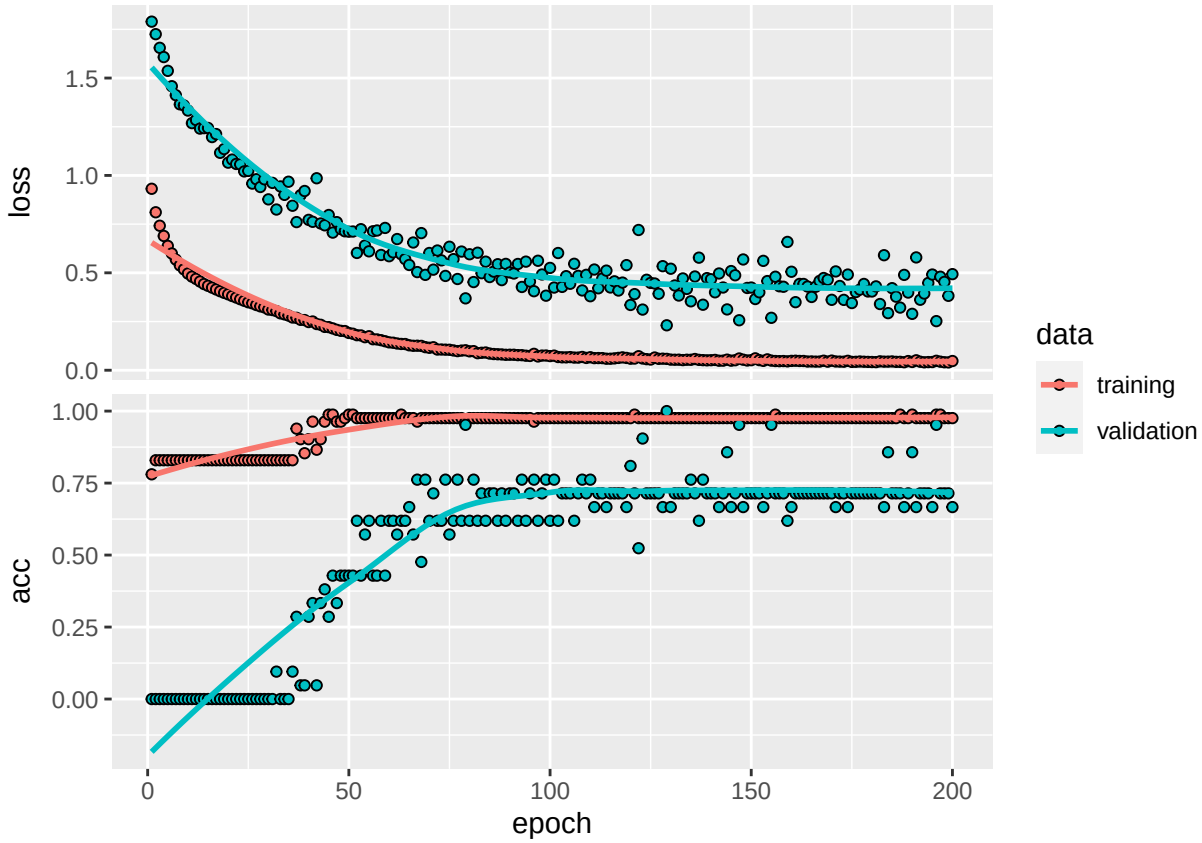
```
iris.prediction <- compute(iris.net, iris.test_nn)
idx <- apply(iris.prediction$net.result, 1, which.max)
predicted <- c('setosa', 'versicolor', 'virginica')[idx]
tab<-table(predicted, iris.test_nn$Species)

score_net<-sum(diag(tab)/sum(tab))
score_net


#Illustration
library(neuralnet)
iris.net <- neuralnet(setosa+versicolor+virginica ~ Petal.Length + Petal.Width,
                      data=iris.training_nn, hidden=c(10,10), rep = 5, err.fct = "ce", act.fct = "logis
                      linear.output = F, lifesign = "minimal", stepmax = 1000000,
                      threshold = 0.01)

iris.prediction <- compute(iris.net, cbind(xgrid[,1],xgrid[,2]))
idx <- apply(iris.prediction$net.result, 1, which.max)
predicted_x <- c(1,2, 3)[idx]

#plot(xgrid,col=c(2,3,4)[as.numeric(predicted_x)],pch = 20,cex = .2, main = "Neural Network",xlab="Peta
#points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)

plot(xgrid,col=c(2,3,4)[as.numeric(predicted_x)],pch = 20,cex = .2, main = "Neural Network",xlab="Petal

#points(xtrain[,1],xtrain[,2],col=c("darkred","forestgreen","darkblue")[unclass(ytrain)],pch=".",cex=5)
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```

```
library(keras)
library(tensorflow)
library(datasets)

data(iris)

#iris[,5] <- as.numeric(iris[,5]) -1
#iris <- as.matrix(iris)
#dimnames(iris) <- NULL


# Split the `iris` data

#iris.training <- iris[ind==1, 1:4]
#iris.test <- iris[ind==2, 1:4]

# Split the class attribute

iris.training_keras <- as.matrix(iris.training)
iris.test_keras <-as.matrix(iris.test)

iris.trainingtarget_keras <- as.numeric(iris.trainingtarget)-1
iris.testtarget_keras <- as.numeric(iris.testtarget) -1
iris.trainLabels <- to_categorical(iris.trainingtarget_keras)
```

```
## Loaded Tensorflow version 1.13.1
```

```
iris.testLabels <- to_categorical(iris.testtarget_keras)


# Initialize a sequential model
model3 <- keras_model_sequential()

# Add layers to the model
model3 %>%
  layer_dense(units = 20, activation = 'relu', input_shape = c(4)) %>%
  layer_dense(units = 10, activation = 'relu') %>%
  layer_dense(units = 3, activation = 'softmax')

# Compile the model
model3 %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = 'adam',
  metrics = 'accuracy'
)

x = as.matrix(apply(iris.training_keras, 2, function(x) (x-min(x))/(max(x) - min(x))))


# Fit the model to the data
fit=model3 %>% fit(
  iris.training_keras, iris.trainLabels,
  epochs = 200, batch_size = 5,
  validation_split = 0.2
)

plot(fit)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



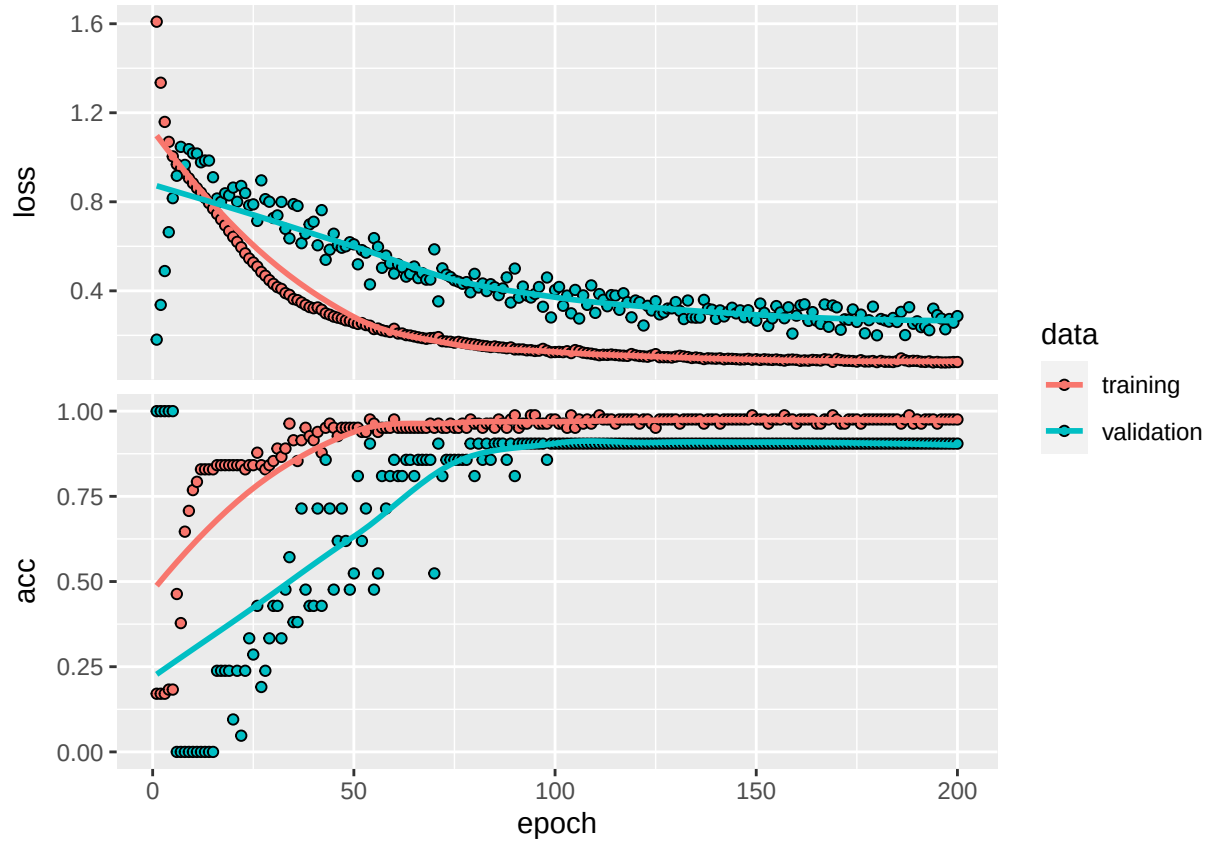

```
# Evaluate the model
score <- model3 %>% evaluate(iris.test_keras, iris.testLabels, batch_size = 128 )

# Print the score
score_keras<-score$acc
print(score_keras)
```

```
## [1] 1
```

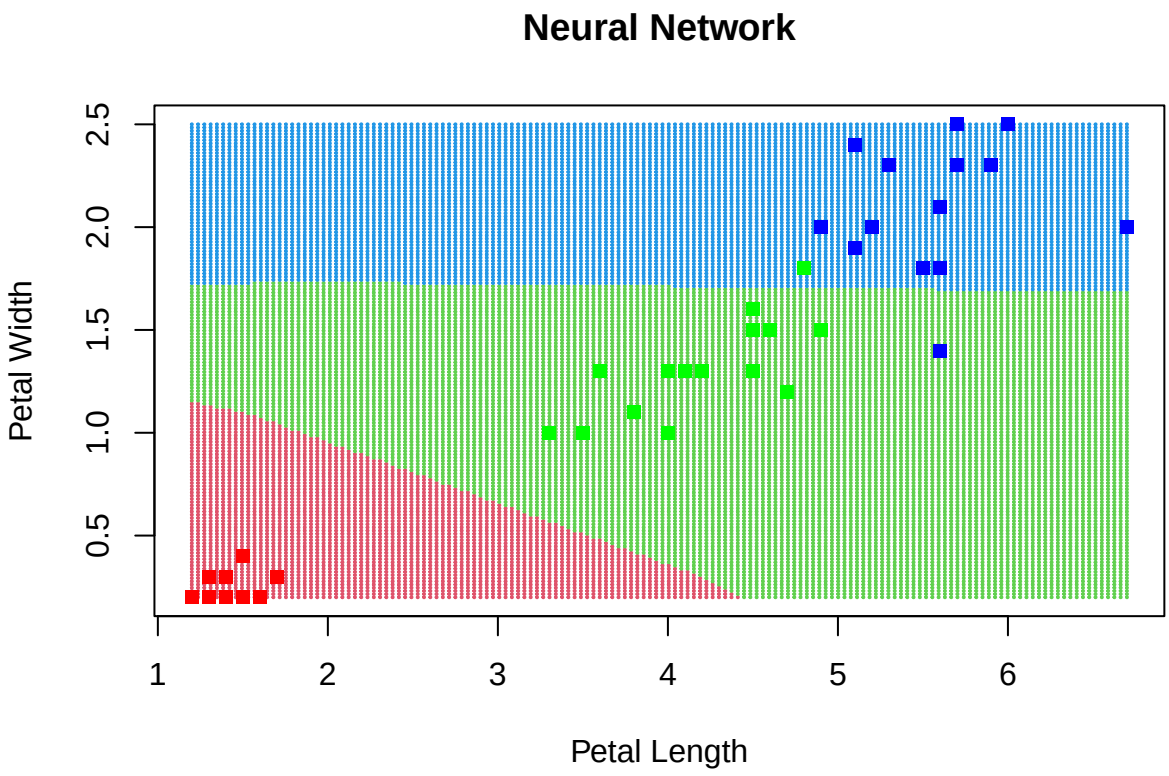
```
library(keras)
library(tensorflow)
library(datasets)
#Illustration
# Initialize a sequential model
model4 <- keras_model_sequential()

# Add layers to the model
model4 %>%
  layer_dense(units = 20, activation = 'relu', input_shape = c(2)) %>%
  layer_dense(units = 10, activation = 'relu') %>%
  layer_dense(units = 3, activation = 'softmax')

# Compile the model
model4 %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = 'adam',
  metrics = 'accuracy'
```

```
)


# Fit the model to the data
fit=model4 %>% fit(
  as.matrix(xtrain), iris.trainLabels,
  epochs = 200, batch_size = 5,
  validation_split = 0.2
)

plot(fit)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
# Evaluate the model
score4<- model4 %>% evaluate(as.matrix(xtest), iris.testLabels, batch_size = 128)




iris.prediction <- predict(model4,as.matrix(xgrid))
idx <- apply(iris.prediction, 1, which.max)
predicted_x <- c(1,2, 3)[idx]

plot(xgrid,col=c(2,3,4)[as.numeric(predicted_x)],pch = 20,cex = .2, main = "Neural Network",xlab="Petal
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```



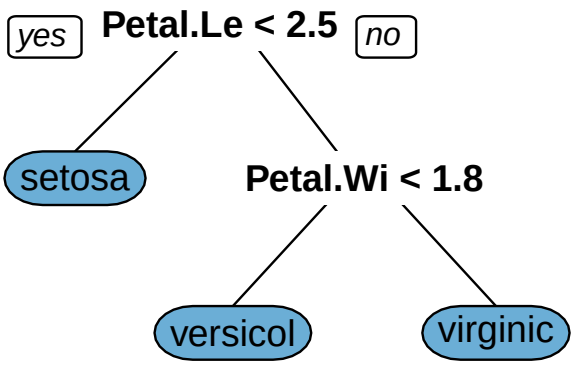

```
#plot(xgrid,col=c(2,3,4)[as.numeric(predicted_x)],pch = 20,cex = .2, main = "Neural Network",xlab="Peta

#points(xtrain[,1],xtrain[,2],col=c("darkred","forestgreen","darkblue")[unclass(ytrain)],pch=".",cex=5)
#points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)


# Print the score
print(score4)
```

```
## $loss
## [1] 0.1396599
##
## $acc
## [1] 0.9574468
```

Decision Trees

```
library(rpart.plot)
library(caret)


dtree_fit_gini <- train(iris.trainingtarget ~ ., data=cbind(iris.training,iris.trainingtarget), method =
                    parms = list(split = "gini"),
                   tuneLength = 10)
```

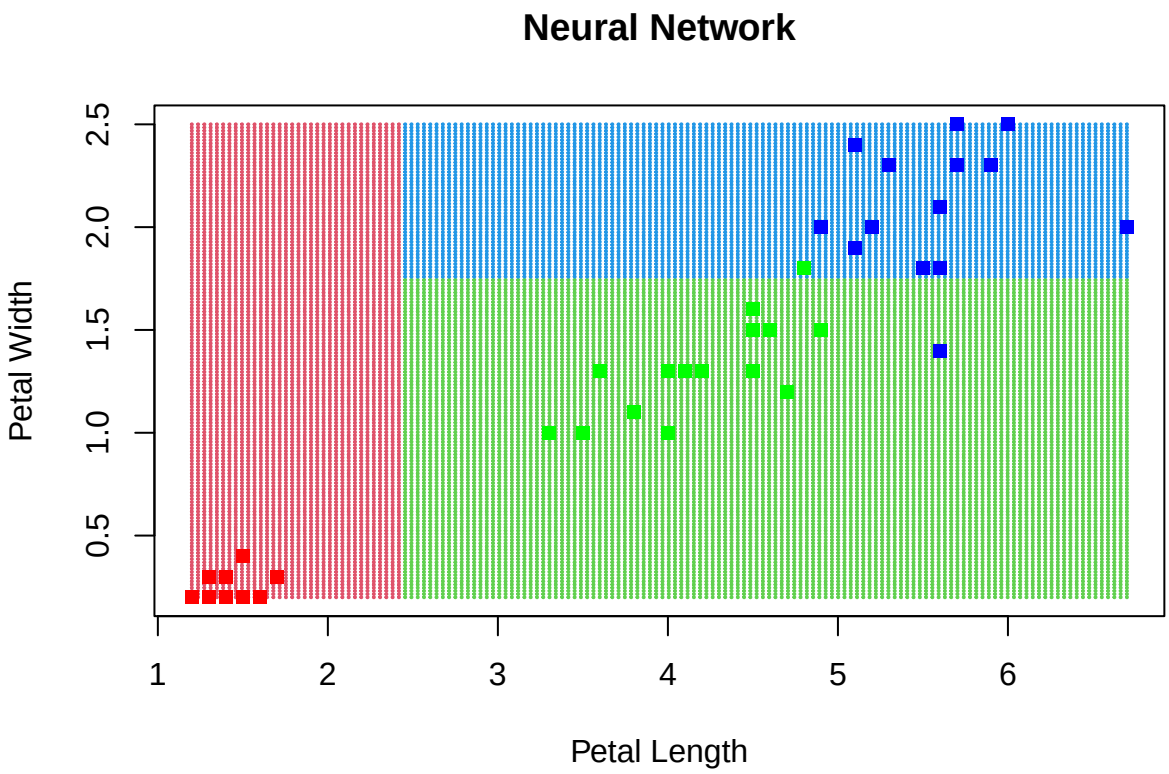
```
prp(dtree_fit_gini$finalModel,box.palette = "Blues", tweak = 1.2)
```




```
test_pred_gini<-predict(dtree_fit_gini,newdata = as.matrix(iris.test))
con_mat<-confusionMatrix(test_pred_gini, iris.testtarget)
score_dt_gini<-sum(diag(con_mat$table))/(sum(con_mat$table))
score_dt_gini
```

```
## [1] 0.9574468
```

```
predicted_x<-predict(dtree_fit_info,newdata = list("Sepal.Length"=xgrid[,1], "Sepal.Width"=xgrid[,2],

plot(xgrid,col=c(2,3,4)[as.numeric(predicted_x)],pch = 20,cex = .2, main = "Neural Network",xlab="Petal

#points(xtrain[,1],xtrain[,2],col=c("darkred","forestgreen","darkblue")[unclass(ytrain)],pch=".",cex=5)
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```



Random Forest

```
library(randomForest)
```

```
## randomForest 4.7-1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attache Paket: 'randomForest'
```

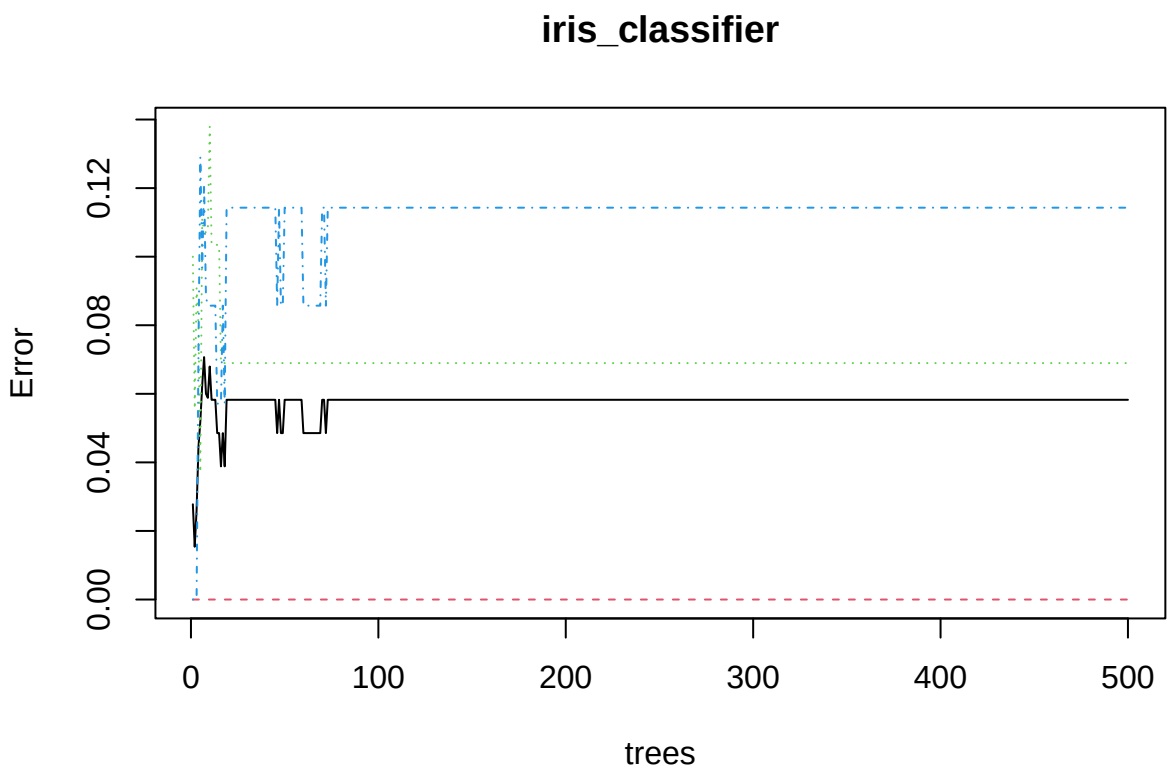
```
## Das folgende Objekt ist maskiert 'package:ggplot2':
##
##     margin
```

```
iris_classifier <- randomForest(iris.trainingtarget ~ ., data=cbind(iris.training,iris.trainingtarget),
                                importance = T)
iris_classifier
```

```
##
## Call:
##  randomForest(formula = iris.trainingtarget ~ ., data = cbind(iris.training,      iris.trainingtarge
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 5.83%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         39          0         0  0.00000000
```

```
## versicolor      0         27         2  0.06896552
## virginica       0          4        31  0.11428571
```

```
plot(iris_classifier)
```




```
predicted_table <- predict(iris_classifier, iris.test)


tab<-table(observed = iris.testtarget, predicted = predicted_table)
score_rf<-sum(diag(tab))/(sum(tab))
score_rf
```

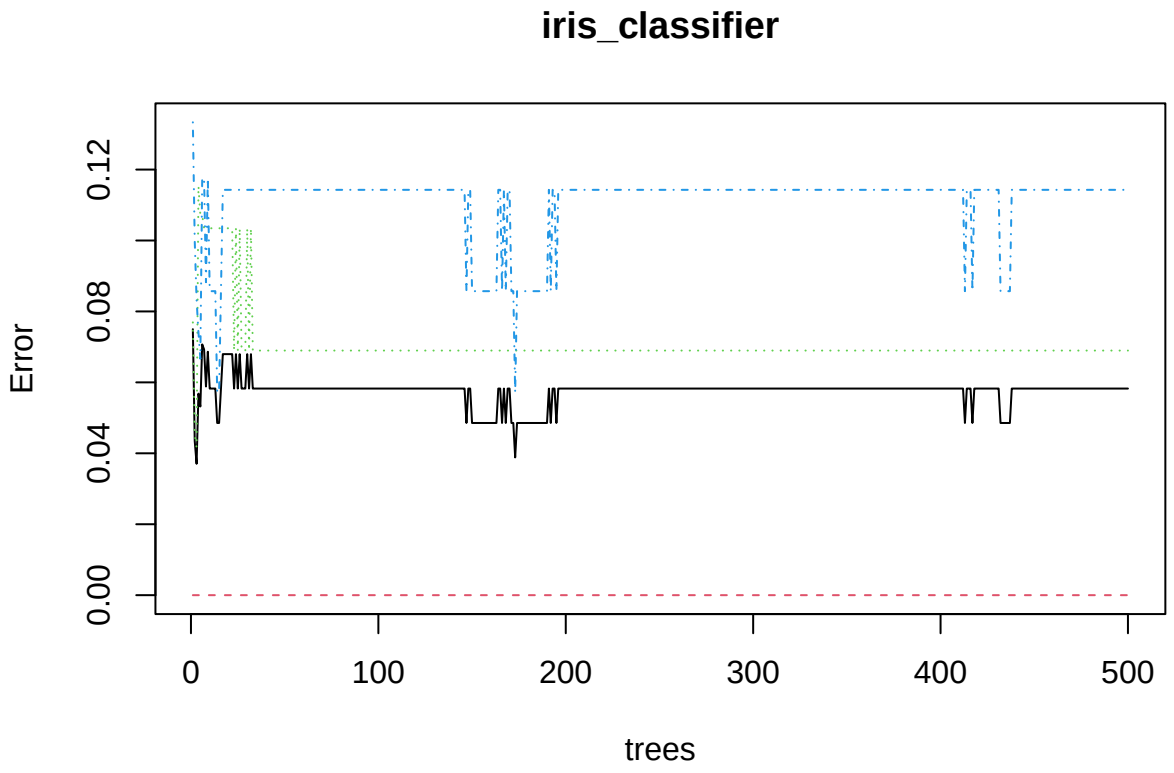
```
## [1] 0.9361702
```

```
#Illustration

iris_classifier <- randomForest(iris.trainingtarget ~ ., data=cbind(iris.training[,3:4],iris.trainingta
                                importance = T)
iris_classifier
```

```
##
## Call:
##  randomForest(formula = iris.trainingtarget ~ ., data = cbind(iris.training[,      3:4], iris.traini
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 5.83%
```

```
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         39          0         0  0.00000000
## versicolor      0         27         2  0.06896552
## virginica       0          4        31  0.11428571
```

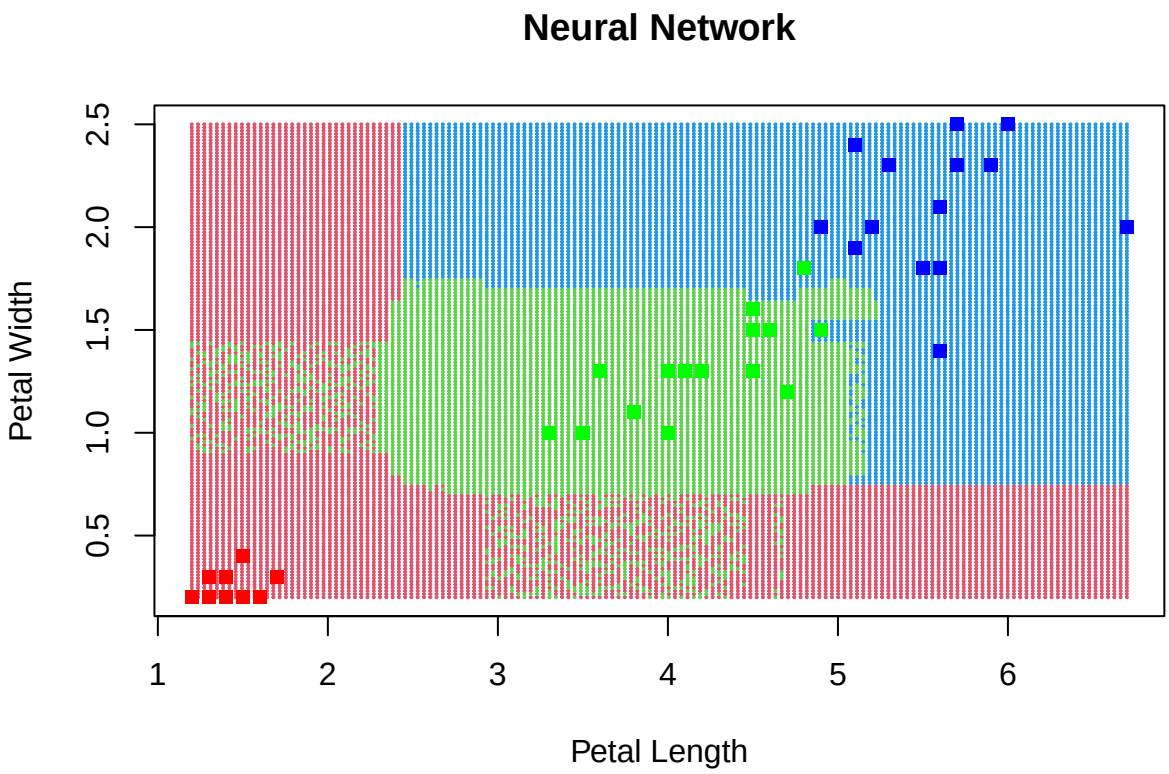
```
plot(iris_classifier)
```




```
predicted_table <- predict(iris_classifier, iris.test)
table(observed = iris.testtarget, predicted = predicted_table)
```

```
##             predicted
## observed     setosa versicolor virginica
##   setosa         11          0         0
##   versicolor      0         18         3
##   virginica       0          0        15
```

```
predicted_x<-predict(iris_classifier,newdata = list("Petal.Length"=xgrid[,1], "Petal.Width"=xgrid[,2] ))

plot(xgrid,col=c(2,3,4)[as.numeric(predicted_x)],pch = 20,cex = .2, main = "Neural Network",xlab="Petal

#points(xtrain[,1],xtrain[,2],col=c("darkred","forestgreen","darkblue")[unclass(ytrain)],pch=".",cex=5)
points(xtest[,1],xtest[,2],col=c("red","green","blue")[unclass(ytest)],pch=".",cex=7)
```



Summary

```
#Illustration

summary<-c("NB"= score_nb, "kNN"= iris_acc[20], "logistic"=scorelog,"SVM"=score_svm, "NN"=score_keras,
print(summary)
```

```
##        NB       kNN  logistic       SVM        NN        DT        RF
## 0.9574468 1.0000000 0.9787234 0.9787234 1.0000000 0.9574468 0.9361702
```