**IMPLEMENT CLASSIFIERS ON THE DATASET**

CLASSIFIERS TO TRAIN:

* Decision Trees
* Perceptron (Single Linear Classifier)
* Neural Net
* Support Vector Machines
* Naïve Bayes Classifiers

1. **DECISION TREE ->**

**Code:**

install.packages("rpart", dependencies = T)

library(rpart)

require(rpart)

ionosphere <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data", header=FALSE)

iii<-ionosphere

install.packages("caret")

library(caret)

install.packages("RANN")

library(RANN)

maxs = apply(iii, MARGIN = 2, max)

mins = apply(iii, MARGIN = 2, min)

scaled = as.data.frame(scale(iii, center = mins, scale = maxs - mins))

scaled[is.na(scaled)] <- 0

iiii<-scaled

trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

train <- iiii[trainIndex, ]

test <- iiii[-trainIndex, ]

dtm<-rpart(V35~.,train,method = "class")

print(dtm)

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

p<-predict(dtm,test,type = "class")

plot(test[,35], p)

d<-table(p,test[,35])

print(d)

y<-(diag(d))/(sum(d))

Z<-sum(y)

print(Z\*100)

-------------------------------------------------------------------------

**LOG:**

trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> dtm<-rpart(V35~.,train,method = "class")

> print(dtm)

n= 280

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 280 97 1 (0.34642857 0.65357143)

2) V5< 0.61577 63 3 0 (0.95238095 0.04761905) \*

3) V5>=0.61577 217 37 1 (0.17050691 0.82949309)

6) V27>=0.9999725 36 8 0 (0.77777778 0.22222222)

12) V22>=0.73857 15 0 0 (1.00000000 0.00000000) \*

13) V22< 0.73857 21 8 0 (0.61904762 0.38095238)

26) V3< 0.8492175 10 0 0 (1.00000000 0.00000000) \*

27) V3>=0.8492175 11 3 1 (0.27272727 0.72727273) \*

7) V27< 0.9999725 181 9 1 (0.04972376 0.95027624) \*

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

Error: could not find function "rpart.plot"

> p<-predict(dtm,test,type = "class")

> plot(test[,35], p)

> d<-table(p,test[,35])

> print(d)

p 0 1

0 21 2

1 8 40

> y<-(diag(d))/(sum(d))

> Z<-sum(y)

> print(Z\*100)

[1] 85.91549

> ionosphere <- read.csv("http://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data", header=FALSE)

> maxs = apply(ionosphere, MARGIN = 2, max)

> mins = apply(ionosphere, MARGIN = 2, min)

> scaled = as.data.frame(scale(ionosphere, center = 34, scale = maxs - mins))

Error in scale.default(ionosphere, center = 34, scale = maxs - mins) :

length of 'center' must equal the number of columns of 'x'

> install.packages("rpart", dependencies = T)

Error in install.packages : Updating loaded packages

> library(rpart)

> require(rpart)

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> dtm<-rpart(V35~.,train,method = "class")

> print(dtm)

n= 280

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 280 101 1 (0.36071429 0.63928571)

2) V5< 0.5729875 56 1 0 (0.98214286 0.01785714) \*

3) V5>=0.5729875 224 46 1 (0.20535714 0.79464286)

6) V27>=0.9999725 45 11 0 (0.75555556 0.24444444)

12) V1< 0.5 15 0 0 (1.00000000 0.00000000) \*

13) V1>=0.5 30 11 0 (0.63333333 0.36666667)

26) V18< 0.13702 9 0 0 (1.00000000 0.00000000) \*

27) V18>=0.13702 21 10 1 (0.47619048 0.52380952)

54) V22>=0.73857 8 1 0 (0.87500000 0.12500000) \*

55) V22< 0.73857 13 3 1 (0.23076923 0.76923077) \*

7) V27< 0.9999725 179 12 1 (0.06703911 0.93296089) \*

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

Error: could not find function "rpart.plot"

> p<-predict(dtm,test,type = "class")

> plot(test[,35], p)

> d<-table(p,test[,35])

> print(d)

p 0 1

0 20 1

1 5 45

> y<-(diag(d))/(sum(d))

> Z<-sum(y)

> print(Z\*100)

[1] 91.5493

> install.packages("rpart", dependencies = T)

Installing package into ‘C:/Users/mastr/Documents/R/win-library/3.3’

(as ‘lib’ is unspecified)

Warning in install.packages :

package ‘rpart’ is in use and will not be installed

> install.packages("rpart.plot")

Installing package into ‘C:/Users/mastr/Documents/R/win-library/3.3’

(as ‘lib’ is unspecified)

trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.3/rpart.plot\_2.1.0.zip'

Content type 'application/zip' length 716190 bytes (699 KB)

downloaded 699 KB

package ‘rpart.plot’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\mastr\AppData\Local\Temp\Rtmpwpp3N9\downloaded\_packages

> library(rpart.plot)

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> dtm<-rpart(V35~.,train,method = "class")

> print(dtm)

n= 280

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 280 97 1 (0.34642857 0.65357143)

2) V5< 0.63153 59 3 0 (0.94915254 0.05084746) \*

3) V5>=0.63153 221 41 1 (0.18552036 0.81447964)

6) V27>=0.9999725 41 11 0 (0.73170732 0.26829268)

12) V22>=0.46473 30 4 0 (0.86666667 0.13333333) \*

13) V22< 0.46473 11 4 1 (0.36363636 0.63636364) \*

7) V27< 0.9999725 180 11 1 (0.06111111 0.93888889) \*

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

> p<-predict(dtm,test,type = "class")

> plot(test[,35], p)

> d<-table(p,test[,35])

> print(d)

p 0 1

0 22 4

1 7 38

> y<-(diag(d))/(sum(d))

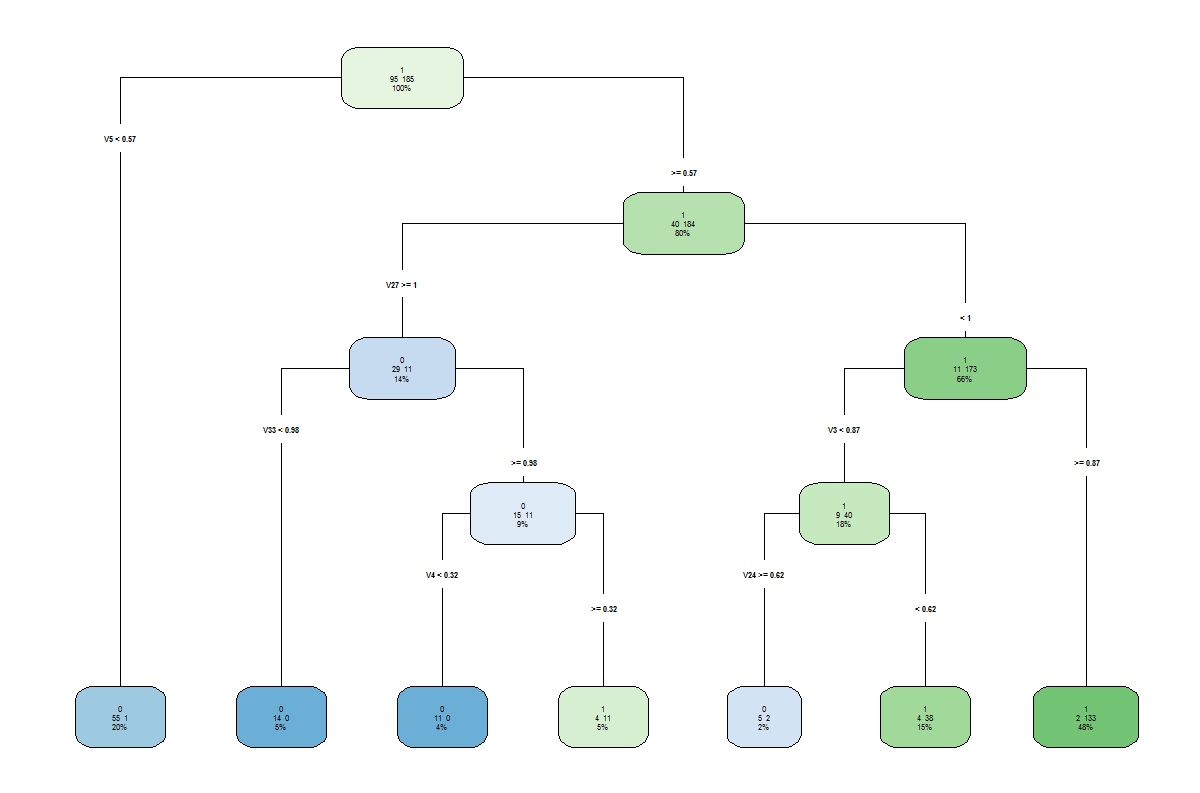
> Z<-sum(y)

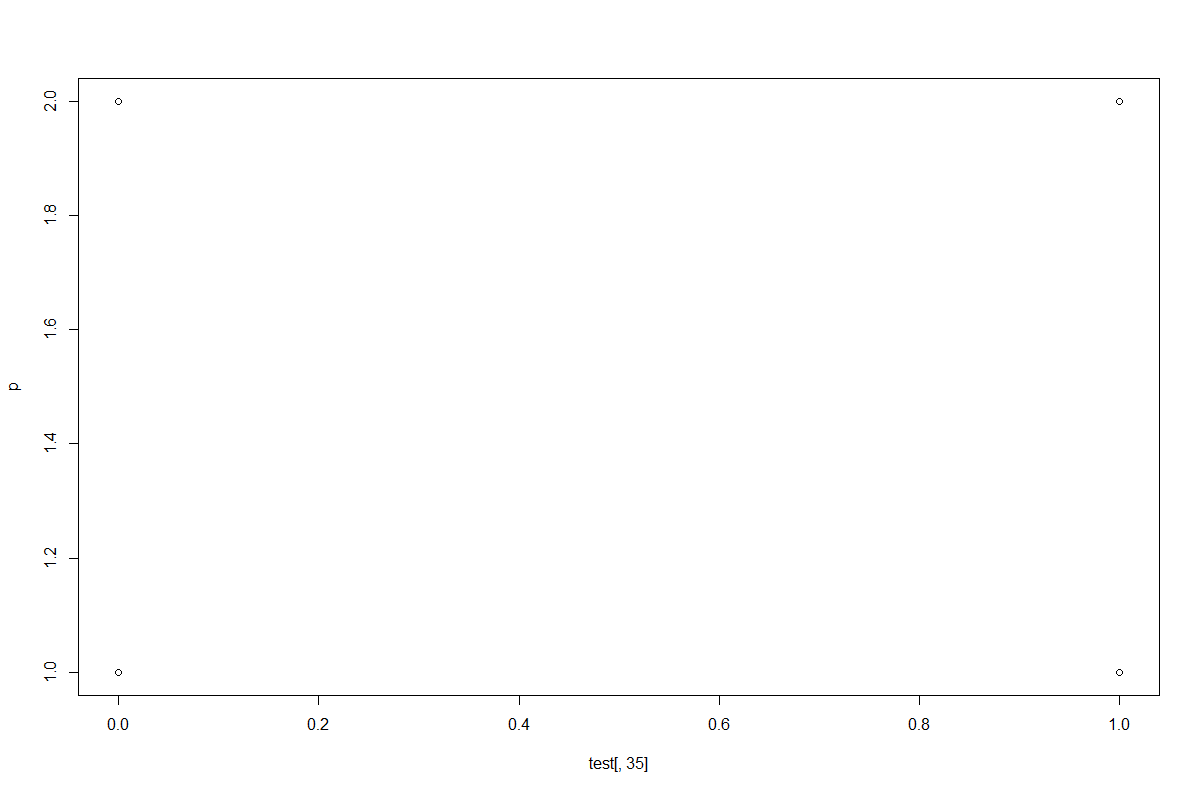
> print(Z\*100)

[1] 84.50704

--------------------------------------------------------------------------------

**Output:**





After running the decision tree classifier for 5 times using random samples of train data and test data, we get-

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment** | **Classifier** | **Train/Test**  **Ratio** | **Outcome**  **(Accuracy)** |
| 1. | DECISION TREE | 80/20 | 84.4% |
| 2. | DECISION TREE | 80/20 | 84.5% |
| 3. | DECISION TREE | 80/20 | 88.7% |
| 4. | DECISION TREE | 80/20 | 77.4% |
| 5. | DECISION TREE | 80/20 | 81.6% |

**-------------------------------------------------------------------------------------------------------------**

1. **NEURAL NETS ->**

**Code:**

install.packages("neuralnet", dependencies = T)

library(neuralnet)

scaled[is.na(scaled)] <- 0

iiii<-scaled

trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))

train <- scaled[trainIndex, ]

test <- scaled[-trainIndex, ]

n <- names(train)

f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9+V8+V7+V6+V5+V4+V3+V2+V1

nn <- neuralnet(f,data=train,hidden = c(1,1), threshold = 0.01, stepmax = 1e+7) **//We keep changing the parameters**

plot(nn)

nn$result.matrix

pred <- compute(nn,test[,1:34])

pred.scaled <- pred$net.result \*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

real.values <- (test$V35)\*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

MSE.nn<-sum((real.values - pred.scaled)^2)/nrow(test)

MSE.nn

accuracy<-(100-MSE.nn)

accuracy

plot(real.values, pred.scaled, col='red',main='Real vs predicted NN')

**LOG:**

iiii<-scaled

> trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))

> train <- scaled[trainIndex, ]

> test <- scaled[-trainIndex, ]

> f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 +

V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9+V8+V7+V6+V5+V4+V3+V2+V1

> nn <- neuralnet(f,data=train,hidden =c(2,2), threshold = 0.001, stepmax = 1e+5)

> plot(nn)

> pred <- compute(nn,test[,1:34])

> pred.scaled <- pred$net.result \*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

> real.values <- (test$V35)\*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

> MSE.nn<-sum((real.values - pred.scaled)^2)/nrow(test)

> MSE.nn

[1] 0.654350042

> accuracy<-(100-MSE.nn)

> accuracy

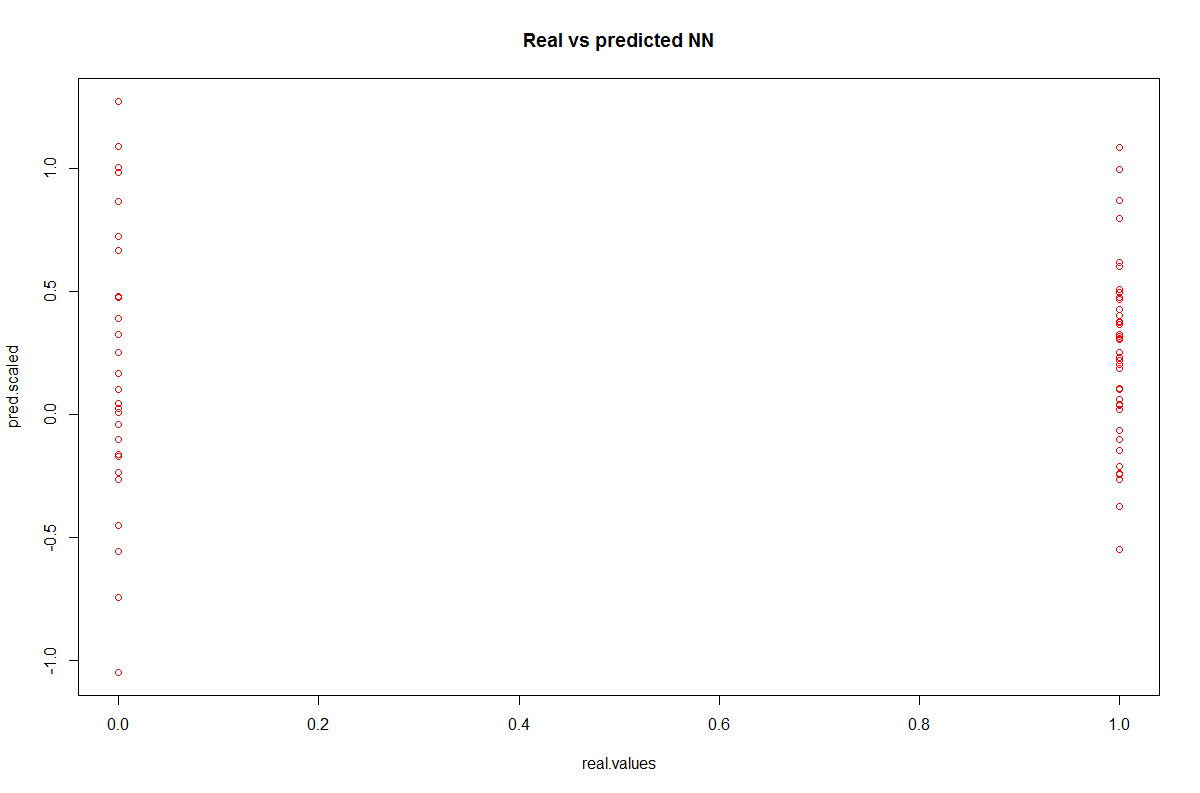
[1] 99.34564996

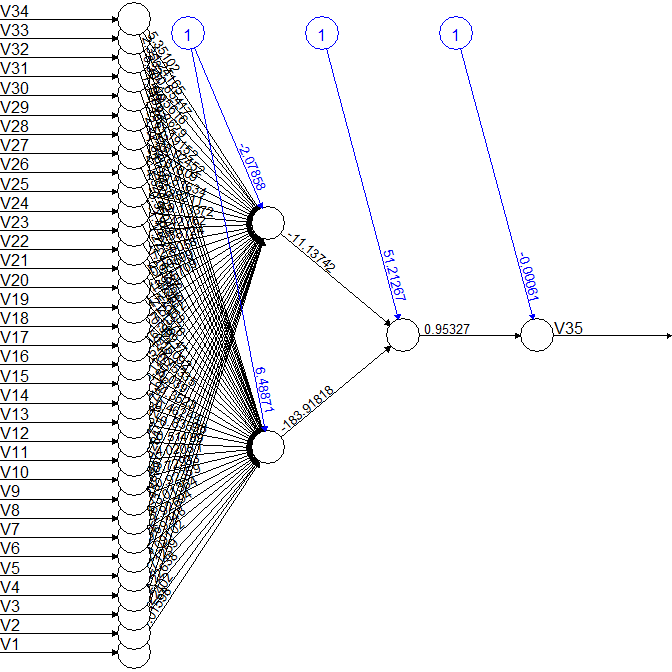
> plot(real.values, pred.scaled, col='red',main='Real vs predicted NN')

> plot(nn)

|  |
| --- |
| > iiii<-scaled  > trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))  > train <- scaled[trainIndex, ]  > test <- scaled[-trainIndex, ]  > f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9+V8+V7+V6+V5+V4+V3+V2+V1  > nn <- neuralnet(f,data=train,hidden =c(2,1), threshold = 0.001, stepmax = 1e+5)  > plot(nn)  > pred <- compute(nn,test[,1:34])  > pred.scaled <- pred$net.result \*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)  > real.values <- (test$V35)\*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)  > MSE.nn<-sum((real.values - pred.scaled)^2)/nrow(test)  > MSE.nn  [1] 0.6154350575  > accuracy<-(100-MSE.nn)  > accuracy  [1] 99.38456494  > plot(real.values, pred.scaled, col='red',main='Real vs predicted NN')  > plot(nn) |
|  |
| |  | | --- | |  | |

**Outputs:**





After running the NeuralNet classifier for 5 times using random samples of train data and test data, we get-

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Experiment** | **Classifier** | **Train/Test**  **Ratio** | **Hidden Layers** | **Threshold** | **Stepmax** | **Outcome**  **(Accuracy)** |
| 1 | NEURALNET | 80/20 | C (2,1) | 0.01 | 1e+07 | 99.36% |
| 2 | NEURALNET | 80/20 | C (1,0) | 0.01 | 1e+07 | 99.37% |
| 3 | NEURALNET | 80/20 | C (1,1) | 0.02 | 1e+07 | 99.33% |
| 4 | NEURALNET | 80/20 | C (2,2) | 0.01 | 1e+07 | 99.43% |
| 5 | NEURALNET | 80/20 | C (2,1) | 0.001 | 1e+07 | 99.28% |

1. **PERCEPTRON ->**

**Code:**

install.packages("neuralnet", dependencies = T)

library(neuralnet)

scaled[is.na(scaled)] <- 0

iiii<-scaled

trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))

train <- scaled[trainIndex, ]

test <- scaled[-trainIndex, ]

n <- names(train)

f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9+V8+V7+V6+V5+V4+V3+V2+V1

nn <- neuralnet(f,data=train,hidden = 0, threshold = 0.01, stepmax = 1e+7) **//We keep changing the parameters**

plot(nn)

nn$result.matrix

pred <- compute(nn,test[,1:34])

pred.scaled <- pred$net.result \*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

real.values <- (test$V35)\*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

MSE.nn<-sum((real.values - pred.scaled)^2)/nrow(test)

MSE.nn

accuracy<-(100-MSE.nn)

accuracy

plot(real.values, pred.scaled, col='red',main='Real vs predicted NN')

**LOG:**

> trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))

> train <- scaled[trainIndex, ]

> test <- scaled[-trainIndex, ]

> f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9+V8+V7+V6+V5+V4+V3+V2+V1

> nn <- neuralnet(f,data=train,hidden =0, threshold = 0.001, stepmax = 1e+5)

> plot(nn)

> pred <- compute(nn,test[,1:34])

> pred.scaled <- pred$net.result \*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

> real.values <- (test$V35)\*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

> MSE.nn<-sum((real.values - pred.scaled)^2)/nrow(test)

> MSE.nn

[1] 1.694192697

> accuracy<-(100-MSE.nn)

> accuracy

[1] 98.3058073

> trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))

> train <- scaled[trainIndex, ]

> test <- scaled[-trainIndex, ]

> f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9+V8+V7+V6+V5+V4+V3+V2+V1

> nn <- neuralnet(f,data=train,hidden =0, threshold = 0.02, stepmax = 1e+6)

> plot(nn)

> pred <- compute(nn,test[,1:34])

> pred.scaled <- pred$net.result \*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

> real.values <- (test$V35)\*(max(iiii$V35)-min(iiii$V35))+min(iiii$V35)

> MSE.nn<-sum((real.values - pred.scaled)^2)/nrow(test)

> MSE.nn

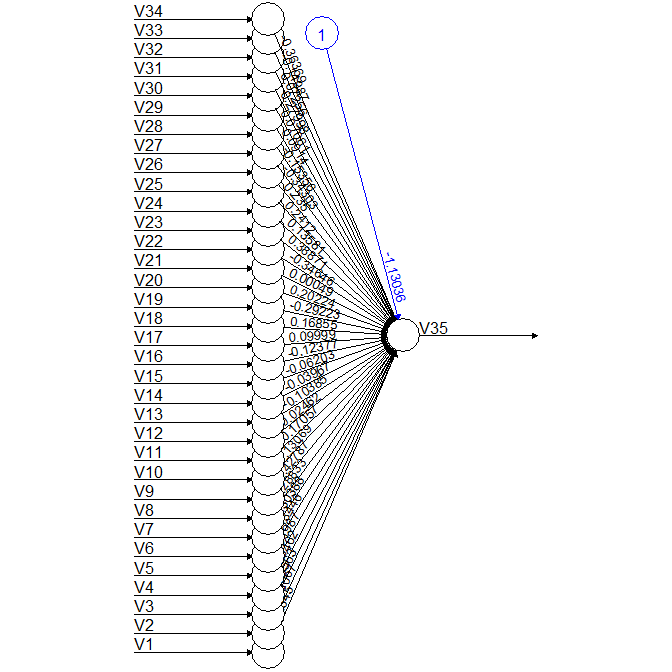
[1] 0.6245665731

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.37543343

**Output:**



After running the Perceptron classifier for 5 times using random samples of train data and test data, we get-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment** | **Classifier** | **Train/Test**  **Ratio-(Random samples)** | **Threshold** | **Stepmax** | **Outcome**  **(Accuracy)** |
| 1 | PERCEPTRON | 80/20 | 0.01 | 1e+05 | 97.49% |
| 2 | PERCEPTRON | 80/20 | 0.02 | 1e+06 | 96.83% |
| 3 | PERCEPTRON | 80/20 | 0.03 | 1e+07 | 99.40% |
| 4 | PERCEPTRON | 80/20 | 0.04 | 1e+06 | 99.08% |
| 5 | PERCEPTRON | 80/20 | 0.001 | 1e+05 | 98.38% |

1. **SVM ->**

**Code:**

install.packages("e1071", dependencies = T)

library(e1071)

require(e1071)

iiii<-scaled

index <- 1:nrow(iiii)

trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

train <- iiii[trainIndex, ]

test <- iiii[-trainIndex, ]

f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9 + V8 + V7 + V6 + V5 + V4 + V3 + V1

svm.model <- svm(f, data = trainset, cost=5, gamma=0.2020 ,type="C-classification")

**//We keep changing the parameters**

summary(svm.model)

svm.pred <- predict(svm.model, test[,-35])

table(predicted= svm.pred,test[,35])

d<-table(predicted= svm.pred,test[,35])

print(d)

y<-(diag(d))/(sum(d))

Z<-sum(y)

print(Z\*100)

**LOG:**

> library(e1071)

> iiii<-scaled

> index <- 1:nrow(iiii)

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9 + V8 + V7 + V6 + V5 + V4 + V3 + V1

> svm.model <- svm(f, data = trainset, cost=5, gamma=0.2020, type= "C-classification")

> summary(svm.model)

Call:

svm(formula = f, data = trainset, cost = 5, gamma = 0.202, type = "C-classification")

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 5

gamma: 0.202

Number of Support Vectors: 60

( 23 37 )

Number of Classes: 2

Levels:

0 1

> svm.pred <- predict(svm.model, test[,-35])

> table(predicted= svm.pred,test[,35])

predicted 0 1

0 15 8

1 11 37

> d<-table(predicted= svm.pred,test[,35])

> print(d)

predicted 0 1

0 15 8

1 11 37

> y<-(diag(d))/(sum(d))

> Z<-sum(y)

> print(Z\*100)

[1] 73.23943662

> iiii<-scaled

> index <- 1:nrow(iiii)

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii))

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

>

> f <- V35 ~ V34 + V33 + V32 + V31 + V30 + V29 + V28 + V27 + V26 + V25 + V24 + V23 + V22 + V21 + V20 + V19 + V18 + V17 + V16 + V15 + V14 + V13 + V12 + V11 + V10 + V9 + V8 + V7 + V6 + V5 + V4 + V3 + V1

> svm.model <- svm(f, data = trainset, cost=1, gamma=0.0303, type= "C-classification")

> summary(svm.model)

Call:

svm(formula = f, data = trainset, cost = 1, gamma = 0.0303, type = "C-classification")

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 1

gamma: 0.0303

Number of Support Vectors: 44

( 23 21 )

Number of Classes: 2

Levels:

0 1

> svm.pred <- predict(svm.model, test[,-35])

> table(predicted= svm.pred,test[,35])

predicted 0 1

0 6 0

1 22 43

> d<-table(predicted= svm.pred,test[,35])

> print(d)

predicted 0 1

0 6 0

1 22 43

> y<-(diag(d))/(sum(d))

> Z<-sum(y)

> print(Z\*100)

[1] 69.01408451

**Output:**

After running the SVM classifier for 5 times using random samples of train data and test data, we get-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment** | **Classifier** | **Train/Test**  **Ratio** | **COST** | **GAMMA** | **Outcome**  **(Accuracy)** |
| 1. | SVM | 80/20 | 1 | 0.030303 | 97.18% |
| 2. | SVM | 80/20 | 100 | 1.000000 | 77.46% |
| 3. | SVM | 80/20 | 200 | 0.010101 | 97.18% |
| 4. | SVM | 80/20 | 10 | 0.010101 | 95.77% |
| 5. | SVM | 80/20 | 5 | 0.202020 | 91.54% |

1. **NAÏVE BAYES ->**

**Code:**

install.packages("e1071", dependencies = T)

library(e1071)

scaled[is.na(scaled)] <- 0

iiii<-scaled

trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))

train <- scaled[trainIndex, ]

test <- scaled[-trainIndex, ]

x.test= test[,1:34]

y.test=test[,35]

nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0.0006)

predicted<-predict(nav.model,x.test)

d= table(predicted, y.test)

accuracy<-(sum(diag(d))/sum(d))\*100

accuracy

**LOG:**

|  |
| --- |
| > scaled[is.na(scaled)] <- 0  > iiii<-scaled  > trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))  > train <- scaled[trainIndex, ]  > test <- scaled[-trainIndex, ]  > x.test= test[,1:34]  > y.test=test[,35]  > nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0.0006)  > predicted<-predict(nav.model,x.test)  > d= table(predicted, y.test)  > accuracy<-(sum(diag(d))/sum(d))\*100  > accuracy  [1] 94.3662  > scaled[is.na(scaled)] <- 0  > iiii<-scaled  > trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))  > train <- scaled[trainIndex, ]  > test <- scaled[-trainIndex, ]  > x.test= test[,1:34]  > y.test=test[,35]  > nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0.0006)  > predicted<-predict(nav.model,x.test)  > d= table(predicted, y.test)  > print(d)  y.test  predicted 0 1  0 25 2  1 3 41  > accuracy<-(sum(diag(d))/sum(d))\*100  > accuracy  [1] 92.95775  > iiii<-scaled  > trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))  > train <- scaled[trainIndex, ]  > test <- scaled[-trainIndex, ]  > x.test= test[,1:34]  > y.test=test[,35]  > nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0.0006)  > predicted<-predict(nav.model,x.test)  > d= table(predicted, y.test)  > print(d)  y.test  predicted 0 1  0 15 2  1 2 52  > accuracy<-(sum(diag(d))/sum(d))\*100  > accuracy  [1] 94.3662  > iiii<-scaled  > trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled))  > train <- scaled[trainIndex, ]  > test <- scaled[-trainIndex, ]  > x.test= test[,1:34]  > y.test=test[,35]  > nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0.0006)  > predicted<-predict(nav.model,x.test)  > d= table(predicted, y.test)  > print(d)  y.test  predicted 0 1  0 20 2  1 6 43  > accuracy<-(sum(diag(d))/sum(d))\*100  > accuracy  [1] 88.73239 |

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment** | **Classifier** | **Train/Test**  **Ratio** | **Threshold** | **Outcome**  **(Accuracy)** |
| 1. | Naïve Bayes | 80/20 | 0.0006 | 88.73% |
| 2. | Naïve Bayes | 80/20 | 0.01 | 87.32% |
| 3. | Naïve Bayes | 80/20 | 0.001 | 88.73% |
| 4. | Naïve Bayes | 80/20 | 0.00003 | 90.14% |
| 5. | Naïve Bayes | 80/20 | 0.0 | 85.91% |