**RESULTS:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sampling#** | **Train/Test**  **Percent**  **split** | **Neural Net Accuracy** | **Decision**  **Tree**  **Accuracy** | **SVM Accuracy** | **Perceptron Accuracy** | **Naïve Bayes Accuracy** |
| 1. | 80/20 | 99.42% | 88.73% | 74.64% | 98.97% | 92.95% |
| 2. | 80/20 | 99.32% | 91.54% | 83.09% | 98.20% | 94.36% |
| 3. | 80/20 | 99.37% | 83.09% | 77.46% | 99.61% | 87.32% |
| 4. | 80/20 | 99.30% | 81.69% | 77.46% | 99.11% | 85.91% |
| 5. | 80/20 | 99.30% | 84.50% | 70.42 | 99.58% | 84.50% |

**LOG OF COMPARISON OF ALL THE CLASSIFIERS (for the above results)**

> library(rpart)

> library(rpart.plot)

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii)) **//FOR SAMPLING #1**

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> #creating decision tree

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

> print(dtm)

n= 280

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

\* denotes terminal node

1) root 280 99 1 (0.35357143 0.64642857)

2) V5< 0.52072 53 0 0 (1.00000000 0.00000000) \*

3) V5>=0.52072 227 46 1 (0.20264317 0.79735683)

6) V27>=0.9999725 39 8 0 (0.79487179 0.20512821) \*

7) V27< 0.9999725 188 15 1 (0.07978723 0.92021277)

14) V7< 0.5049025 7 3 0 (0.57142857 0.42857143) \*

15) V7>=0.5049025 181 11 1 (0.06077348 0.93922652) \*

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

> #predicting testdata on the decision tree

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

> plot(test[,35], p)

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

p 0 1

0 24 5

1 3 39

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

> Z<-sum(y)

> print(Z\*100)

[1] 88.73239

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

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/e1071\_1.6-7.zip'

Content type 'application/zip' length 897070 bytes (876 KB)

downloaded 876 KB

package ‘e1071’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\mastr\AppData\Local\Temp\Rtmpo7p9hY\downloaded\_packages

> library(e1071)

> iiii<-scaled

> x.test= test[,1:34]

> y.test=test[,35]

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 24 2

1 3 42

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

> accuracy

[1] 92.95775

> library(neuralnet)

> 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)

> #predicting testdata on the NeuralNet

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.5712614843

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.42873852

> #creating Perceptron model on training data

> 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 = 0.002, stepmax = 1e+5)

> plot(nn)

> #predicting testdata on the Perceptron

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 1.028801827

> accuracy<-(100-MSE.nn)

> accuracy

[1] 98.97119817

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 20 11

1 7 33

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 20 11

1 7 33

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

> accuracy

[1] 74.64788732

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii)) **//FOR SAMPLING #2**

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> #creating SVM model

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 15 6

1 6 44

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 15 6

1 6 44

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

> accuracy

[1] 83.09859155

> #creating Perceptron model on training data

> 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 = 0.002, stepmax = 1e+5)

> plot(nn)

> #predicting testdata on the Perceptron

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 1.797871919

> accuracy<-(100-MSE.nn)

> accuracy

[1] 98.20212808

> 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)

> #predicting testdata on the NeuralNet

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.6754632064

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.32453679

> x.test= test[,1:34]

> y.test=test[,35]

> #creating NaiveBayes model on training data

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 17 0

1 4 50

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

> accuracy

[1] 94.36619718

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

> print(dtm)

n= 280

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

\* denotes terminal node

1) root 280 105 1 (0.37500000000 0.62500000000)

2) V5< 0.52072 57 0 0 (1.00000000000 0.00000000000) \*

3) V5>=0.52072 223 48 1 (0.21524663677 0.78475336323)

6) V27>=0.9999725 43 11 0 (0.74418604651 0.25581395349)

12) V1< 0.5 15 0 0 (1.00000000000 0.00000000000) \*

13) V1>=0.5 28 11 0 (0.60714285714 0.39285714286)

26) V4< 0.2411975 8 0 0 (1.00000000000 0.00000000000) \*

27) V4>=0.2411975 20 9 1 (0.45000000000 0.55000000000)

54) V6< 0.4008325 7 1 0 (0.85714285714 0.14285714286) \*

55) V6>=0.4008325 13 3 1 (0.23076923077 0.76923076923) \*

7) V27< 0.9999725 180 16 1 (0.08888888889 0.91111111111)

14) V8< 0.23388 8 3 0 (0.62500000000 0.37500000000) \*

15) V8>=0.23388 172 11 1 (0.06395348837 0.93604651163) \*

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

> #predicting testdata on the decision tree

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

> plot(test[,35], p)

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

p 0 1

0 15 0

1 6 50

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

> Z<-sum(y)

> print(Z\*100)

[1] 91.54929577

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii)) **//FOR SAMPLING #3**

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> #creating decision tree

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

> print(dtm)

n= 280

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

\* denotes terminal node

1) root 280 97 1 (0.34642857143 0.65357142857)

2) V5< 0.5729875 59 2 0 (0.96610169492 0.03389830508) \*

3) V5>=0.5729875 221 40 1 (0.18099547511 0.81900452489)

6) V27>=0.9999725 36 11 0 (0.69444444444 0.30555555556)

12) V1< 0.5 13 0 0 (1.00000000000 0.00000000000) \*

13) V1>=0.5 23 11 0 (0.52173913043 0.47826086957)

26) V25< 0.91266 8 0 0 (1.00000000000 0.00000000000) \*

27) V25>=0.91266 15 4 1 (0.26666666667 0.73333333333) \*

7) V27< 0.9999725 185 15 1 (0.08108108108 0.91891891892)

14) V8< 0.3143275 7 2 0 (0.71428571429 0.28571428571) \*

15) V8>=0.3143275 178 10 1 (0.05617977528 0.94382022472) \*

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

> #predicting testdata on the decision tree

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

> plot(test[,35], p)

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

p 0 1

0 22 5

1 7 37

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

> Z<-sum(y)

> print(Z\*100)

[1] 83.09859155

> 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)

> #predicting testdata on the NeuralNet

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.6206338203

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.37936618

> #creating Perceptron model on training data

> 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 = 0.002, stepmax = 1e+5)

> plot(nn)

> #predicting testdata on the Perceptron

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.3856027261

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.61439727

> x.test= test[,1:34]

> y.test=test[,35]

> #creating NaiveBayes model on training data

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 24 4

1 5 38

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

> accuracy

[1] 87.32394366

> #creating SVM model

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 19 6

1 10 36

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 19 6

1 10 36

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

> accuracy

[1] 77.46478873

> #creating SVM model

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 19 6

1 10 36

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 19 6

1 10 36

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

> accuracy

[1] 77.46478873

> x.test= test[,1:34]

> y.test=test[,35]

> #creating NaiveBayes model on training data

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 24 4

1 5 38

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

> accuracy

[1] 87.32394366

> trainIndex <- sample(1:nrow(scaled), 0.8 \* nrow(scaled)) **//FOR SAMPLING #4**

> train <- scaled[trainIndex, ]

> test <- scaled[-trainIndex, ]

> x.test= test[,1:34]

> y.test=test[,35]

> #creating NaiveBayes model on training data

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 22 6

1 4 39

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

> accuracy

[1] 85.91549296

> #creating Perceptron model on training data

> 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 = 0.002, stepmax = 1e+5)

> plot(nn)

> #predicting testdata on the Perceptron

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.8824306888

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.11756931

> #creating SVM model

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 20 10

1 6 35

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 20 10

1 6 35

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

> accuracy

[1] 77.46478873

> #creating SVM model

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 20 10

1 6 35

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 20 10

1 6 35

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

> accuracy

[1] 77.46478873

> #creating NeuralNet model on training data

> 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)

> #predicting testdata on the NeuralNet

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.6980915825

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.30190842

> trainIndex <- sample(1:nrow(iiii),0.8\*nrow(iiii)) **//FOR SAMPLING #5**

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> #creating decision tree

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

> print(dtm)

n= 280

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

\* denotes terminal node

1) root 280 101 1 (0.36071428571 0.63928571429)

2) V5< 0.6408325 62 2 0 (0.96774193548 0.03225806452) \*

3) V5>=0.6408325 218 41 1 (0.18807339450 0.81192660550)

6) V27>=0.9999725 41 10 0 (0.75609756098 0.24390243902)

12) V22>=0.46473 31 4 0 (0.87096774194 0.12903225806) \*

13) V22< 0.46473 10 4 1 (0.40000000000 0.60000000000) \*

7) V27< 0.9999725 177 10 1 (0.05649717514 0.94350282486) \*

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

> #predicting testdata on the decision tree

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

> plot(test[,35], p)

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

p 0 1

0 17 5

1 8 41

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

> Z<-sum(y)

> print(Z\*100)

[1] 81.69014085

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

> train <- iiii[trainIndex, ]

> test <- iiii[-trainIndex, ]

> #creating decision tree

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

> print(dtm)

n= 280

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

\* denotes terminal node

1) root 280 104 1 (0.37142857143 0.62857142857)

2) V5< 0.5729875 63 2 0 (0.96825396825 0.03174603175) \*

3) V5>=0.5729875 217 43 1 (0.19815668203 0.80184331797)

6) V27>=0.9999725 41 10 0 (0.75609756098 0.24390243902)

12) V1< 0.5 16 0 0 (1.00000000000 0.00000000000) \*

13) V1>=0.5 25 10 0 (0.60000000000 0.40000000000)

26) V8< 0.45309 11 1 0 (0.90909090909 0.09090909091) \*

27) V8>=0.45309 14 5 1 (0.35714285714 0.64285714286) \*

7) V27< 0.9999725 176 12 1 (0.06818181818 0.93181818182)

14) V8< 0.231495 7 2 0 (0.71428571429 0.28571428571) \*

15) V8>=0.231495 169 7 1 (0.04142011834 0.95857988166) \*

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

> #predicting testdata on the decision tree

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

> plot(test[,35], p)

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

p 0 1

0 13 2

1 9 47

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

> Z<-sum(y)

> print(Z\*100)

[1] 84.50704225

> x.test= test[,1:34]

> y.test=test[,35]

> #creating NaiveBayes model on training data

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 17 6

1 5 43

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

> accuracy

[1] 84.50704225

> x.test= test[,1:34]

> y.test=test[,35]

> #creating NaiveBayes model on training data

> nav.model<-naiveBayes(as.factor(V35)~. ,data=train, threshold=0)

> #predicting testdata on the Model

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

> #calculating accuracy (predicted data VS actual data) using confusionMatrix

> d= table(predicted, y.test)

> print(d)

y.test

predicted 0 1

0 17 6

1 5 43

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

> accuracy

[1] 84.50704225

> #creating Perceptron model on training data

> 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 = 0.002, stepmax = 1e+5)

> plot(nn)

> #predicting testdata on the Perceptron

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.4121768645

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.58782314

> #creating SVM model

> 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

> #predicting testdata on the SVM model

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

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

predicted 0 1

0 13 12

1 9 37

> #using ConfusionMatrix to calculate accuracy

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

> print(d)

predicted 0 1

0 13 12

1 9 37

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

> accuracy

[1] 70.42253521

> #creating NeuralNet model on training data

> 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)

> #predicting testdata on the NeuralNet

> 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)

> #calcultaing Mean Square Error to find accuracy

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

> MSE.nn

[1] 0.6903382597

> accuracy<-(100-MSE.nn)

> accuracy

[1] 99.30966174