|  |
| --- |
|  |

>library(tidyverse)

> library(neuralnet)

> library(GGally)

> library("nnet")

> library(caret)

>

> # Importing the dataset

> datum= read.csv('Dummy Data With State for ANN Classification.csv')

> vars=c("Segment..Dummy.", "Total.Amount", "Unit.Price")

>

> #PARTIONING

> set.seed(12345)

> datum\_Train <- sample\_frac(tbl = datum, replace = FALSE, size = 0.60)

> datum\_Test <- anti\_join(datum, datum\_Train)

Joining, by = c("ï..Order.Status..Dummy.", "Segment..Dummy.", "Total.Amount", "Unit.Price")

>

> # when y has multiple classes - need to dummify

> trainData <- cbind(datum\_Train[c(vars)],

+ class.ind(datum\_Train$ï..Order.Status..Dummy.))

> names(trainData)=c(vars,

+ paste("Order.Status..Dummy\_", c(1, 2, 3, 4), sep=""))

> validData <- cbind(datum\_Test[c(vars)],

+ class.ind(datum\_Test$ï..Order.Status..Dummy.))

> names(validData)=c(vars,

+ paste("Order.Status..Dummy\_", c(1, 2, 3, 4), sep=""))

>

> #run nn with 2 hidden nodes

> #use hidden= with a vector of integers specifying number of hidden nodes in each layer

> nn <- neuralnet(Order.Status..Dummy\_1 + Order.Status..Dummy\_2 +

+ Order.Status..Dummy\_3 + Order.Status..Dummy\_4 ~

+ Segment..Dummy.+ Total.Amount+ Unit.Price,

+ data = trainData, hidden = 2)

> plot(nn)

> training.prediction=compute(nn, trainData)

> training.class=apply(training.prediction$net.result,1,which.max)

> confusionMatrix(factor(training.class), factor(datum\_Train$ï..Order.Status..Dummy., levels = 1:4))

Confusion Matrix and Statistics

Reference

Prediction 1 2 3 4

1 434 31 10 6

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

Overall Statistics

Accuracy : 0.9023

95% CI : (0.8722, 0.9273)

No Information Rate : 0.9023

P-Value [Acc > NIR] : 0.5387

Kappa : 0

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: 1 Class: 2 Class: 3 Class: 4

Sensitivity 1.0000 0.00000 0.00000 0.00000

Specificity 0.0000 1.00000 1.00000 1.00000

Pos Pred Value 0.9023 NaN NaN NaN

Neg Pred Value NaN 0.93555 0.97921 0.98753

Prevalence 0.9023 0.06445 0.02079 0.01247

Detection Rate 0.9023 0.00000 0.00000 0.00000

Detection Prevalence 1.0000 0.00000 0.00000 0.00000

Balanced Accuracy 0.5000 0.50000 0.50000 0.50000

Warning message:

In confusionMatrix.default(factor(training.class), factor(datum\_Train$ï..Order.Status..Dummy., :

Levels are not in the same order for reference and data. Refactoring data to match.

>

> #testing data

> validation.prediction=compute(nn, validData)

> validation.class=apply(validation.prediction$net.result,1,which.max)

> confusionMatrix(factor(validation.class), factor(datum\_Test$ï..Order.Status..Dummy., levels = 1:4))

Confusion Matrix and Statistics

Reference

Prediction 1 2 3 4

1 268 24 9 11

2 0 0 0 0

3 0 0 0 0

4 0 0 0 0

Overall Statistics

Accuracy : 0.859

95% CI : (0.8153, 0.8956)

No Information Rate : 0.859

P-Value [Acc > NIR] : 0.5401

Kappa : 0

Mcnemar's Test P-Value : NA

Statistics by Class:

Class: 1 Class: 2 Class: 3 Class: 4

Sensitivity 1.000 0.00000 0.00000 0.00000

Specificity 0.000 1.00000 1.00000 1.00000

Pos Pred Value 0.859 NaN NaN NaN

Neg Pred Value NaN 0.92308 0.97115 0.96474

Prevalence 0.859 0.07692 0.02885 0.03526

Detection Rate 0.859 0.00000 0.00000 0.00000

Detection Prevalence 1.000 0.00000 0.00000 0.00000

Balanced Accuracy 0.500 0.50000 0.50000 0.50000

Warning message:

In confusionMatrix.default(factor(validation.class), factor(datum\_Test$ï..Order.Status..Dummy., :

Levels are not in the same order for reference and data. Refactoring data to match.

>