The Artificial Neural network

Running Neuralnetwork on all 25 variables before feature selection

```
setwd("C:/Users/Administrator/Documents/archive (3)")
ktd <- read.csv("kdd3.csv",header = TRUE )


ktd<-na.omit(ktd)

#nrow(ktd)

#sum(is.na(ktd$pe))

#it is dataset with 25 variables before applying feature selection result
kdn<- ktd
```

```
kdn[9]<- ifelse(kdn$ba == "present", 1, 0)
kdn[20]<- ifelse(kdn$dm == "yes", 1, 0)
kdn[21] <- ifelse(kdn$cad == "yes", 1, 0)

kdn[6] <- ifelse(kdn$rbc == "normal", 1, 0)
kdn[7] <- ifelse(kdn$pc == "normal", 0, 1) ##################changes

kdn[8] <- ifelse(kdn$pcc == "present", 1, 0)

kdn[22] <- ifelse(kdn$appet == "good", 1, 0)
kdn[23] <- ifelse(kdn$pe== "yes", 1, 0)
kdn[24] <- ifelse(kdn$ane == "yes", 1, 0)
kdn[19] <- ifelse(kdn$htn == "yes", 0, 1) ################################change


kdn[25] <- ifelse(kdn$classification == "ckd", 1, 0)


# Scaling data for the NN
maxs <- apply(kdn, 2, max)
mins <- apply(kdn, 2, min)
scaled <- as.data.frame(scale(kdn, center = mins, scale = maxs - mins))
#view(scaled)
# Train-test split
train_ <- scaled[184,]

test_ <- scaled[-184,]
```

```r
set.seed(123)
library(neuralnet) # library to fit neural network

n <- names(train_)
f <- as.formula(paste("classification ~", paste(n[!n %in% "classification"], collapse = " + ")))
nn <- neuralnet(f,data=train_,hidden=c(8,8), act.fct = "logistic", linear.output=T)
plot(nn)
```

```r
#run25 var dataset

nn.results <- compute(nn,test_[,1:25])

results <- data.frame(actual = test_$classification, prediction = nn.results$net.result)

results
```

```
##     actual    prediction
## 4        1 -0.1397666752
## 6        1  0.2749975721
## 10       1  0.0235680276
## 12       1 -0.0665193996
## 13       1 -0.4257101353
## 15       1 -0.7116906873
## 16       1  0.0066480825
## 19       1 -0.2752820480
## 21       1 -0.4531212601
## 23       1  0.1664494474
## 26       1  0.0605432532
## 27       1 -0.4929903183
## 28       1 -0.0396817241
## 32       1 -0.2391111501
## 33       1 -0.5257027442
## 40       1 -0.0163140994
## 43       1 -0.0459745667
## 44       1  0.1893921346
## 47       1  0.1224093957
## 49       1 -0.0023614716
## 59       1 -0.2670168917
## 63       1  0.0765082198
## 70       1 -0.0670098592
## 71       1 -0.3200164096
## 74       1  0.1543874297
## 76       1 -0.1364171352
## 77       1  0.2754993349
## 80       1 -0.1623168165
## 84       1 -0.1070530195
## 90       1 -0.4475841377
## 91       1 -0.1288678241
## 92       1 -0.5832071734
## 93       1 -0.6876501498
## 94       1  0.1059226355
## 97       1 -0.2439856435
## 101      1 -0.0633274539
```

```
## 105      1 -0.0302522850
## 107      1  0.0349205873
## 108      1  0.2284791948
## 110      1  0.1031746305
## 111      1 -0.3014017379
## 126      1 -0.4464668857
## 127      1 -0.4683992997
## 128      1  0.0130196031
## 130      1 -0.3072681737
## 133      1 -0.3656458954
## 144      1  0.0056069193
## 147      1 -0.1407009896
## 153      1 -0.1891974866
## 157      1 -0.0272107807
## 159      1 -0.2634991614
## 162      1 -0.3735783021
## 169      1 -0.0030937986
## 170      1 -0.1558398436
## 171      1 -0.0503394360
## 175      1 -0.1136170371
## 180      1  0.1710880606
## 183      1 -0.4003632493
## 187      1 -0.1905553804
## 188      1 -0.0972637970
## 194      1 -0.4145154968
## 196      1 -0.3850793807
## 197      1  0.1765724151
## 198      1 -0.2311034987
## 208      1 -0.5094871335
## 210      1 -0.4367902209
## 211      1  0.0586714850
## 215      1  0.0259584967
## 221      1 -0.0672130190
## 223      1 -0.2437130600
## 224      1  0.1508444470
## 227      1  0.1783059737
## 239      1  0.0609878454
## 240      1 -0.2909423269
## 241      1 -0.1100030230
## 242      1  0.1067730454
## 244      1 -0.1659823116
## 246      1  0.0080318862
## 247      1 -0.2959528529
## 248      0 -0.0195384501
## 249      0 -0.0765395730
## 250      0 -0.0906012221
## 251      0 -0.1620466099
## 252      0  0.0254497347
## 253      0 -0.1070242676
## 254      0  0.0433852632
## 255      0  0.0186253799
## 256      0 -0.0586408164
## 257      0 -0.1916485646
## 258      0 -0.0760561398
```

```
## 259      0 -0.1420069822
## 260      0 -0.1100510714
## 261      0 -0.1458626706
## 262      0 -0.1148631114
## 263      0 -0.0037242136
## 264      0 -0.0863091137
## 265      0 -0.0405265696
## 267      0 -0.1215416415
## 268      0 -0.1105954461
## 269      0 -0.0135508760
## 270      0 -0.1428097898
## 273      0 -0.1196084255
## 275      0 -0.0862975486
## 276      0 -0.0000551156
## 277      0 -0.1482650825
## 279      0 -0.1397123331
## 280      0 -0.1461067847
## 282      0 -0.1983437653
## 283      0 -0.1076089761
## 284      0  0.0689346056
## 286      0  0.0371452617
## 287      0 -0.0682410340
## 288      0  0.1384928753
## 289      0 -0.1747240030
## 290      0 -0.0948572068
## 292      0 -0.3304821541
## 294      0 -0.1566270363
## 295      0  0.0323552590
## 296      0  0.0294431744
## 297      0  0.0973236569
## 298      0  0.0388232460
## 299      0  0.0045177389
## 302      0 -0.1502725018
## 303      0 -0.1025858925
## 304      0 -0.1773297766
## 305      0  0.1390673207
## 306      0 -0.1236641260
## 307      0  0.1361357087
## 308      0  0.1215819463
## 309      0  0.0631703086
## 311      0 -0.0947373437
## 312      0 -0.0434457763
## 315      0 -0.0464978776
## 316      0 -0.1235276064
## 318      0  0.0335001022
## 319      0  0.0949735493
## 321      0 -0.0974711574
## 323      0 -0.2020781043
## 324      0  0.0479488743
## 325      0  0.0881498366
## 327      0  0.0631152787
## 329      0 -0.0279959299
## 331      0 -0.2141178117
## 333      0  0.0346013738
```

```
## 335      0 -0.0872112932
## 336      0 -0.1154101566
## 337      0 -0.0774236910
## 338      0 -0.0830151340
## 339      0  0.0110314060
## 340      0 -0.0706908446
## 341      0 -0.1288450510
## 342      0  0.1305439573
## 343      0  0.0146097916
## 345      0 -0.0295039423
## 346      0 -0.0893013104
## 347      0  0.1590319575
## 348      0  0.2284479113
## 349      0 -0.1181262667
## 350      0 -0.0454267163
## 351      0  0.0426149899
## 352      0  0.0532044374
## 353      0  0.0253349922
## 354      0 -0.1278100601
## 355      0  0.1125651557
## 356      0 -0.0222959205
## 357      0 -0.0327431227
## 358      0 -0.1173427555
## 359      0 -0.1348756244
## 360      0  0.0190261834
## 362      0 -0.0909770183
## 364      0  0.0119284571
## 365      0 -0.1225097673
## 366      0 -0.0871728307
## 367      0  0.0390066547
## 368      0 -0.0353811564
## 369      0 -0.0886801187
## 370      0  0.1284609494
## 371      0 -0.0094352666
## 372      0 -0.1515885604
## 373      0 -0.0484108184
## 374      0  0.0316330172
## 375      0  0.0065023329
## 377      0 -0.2109723306
## 379      0  0.0579997810
## 380      0 -0.1901701724
## 381      0 -0.1647668116
## 382      0  0.1941553736
## 383      0 -0.0174944285
## 384      0 -0.1973695929
## 385      0 -0.1646185944
## 386      0 -0.0517158488
## 387      0 -0.2178589020
## 388      0 -0.1504768204
## 389      0 -0.1486871178
## 390      0 -0.1471795761
## 391      0 -0.0301321329
## 392      0 -0.0813019273
## 393      0 -0.1368169547
```

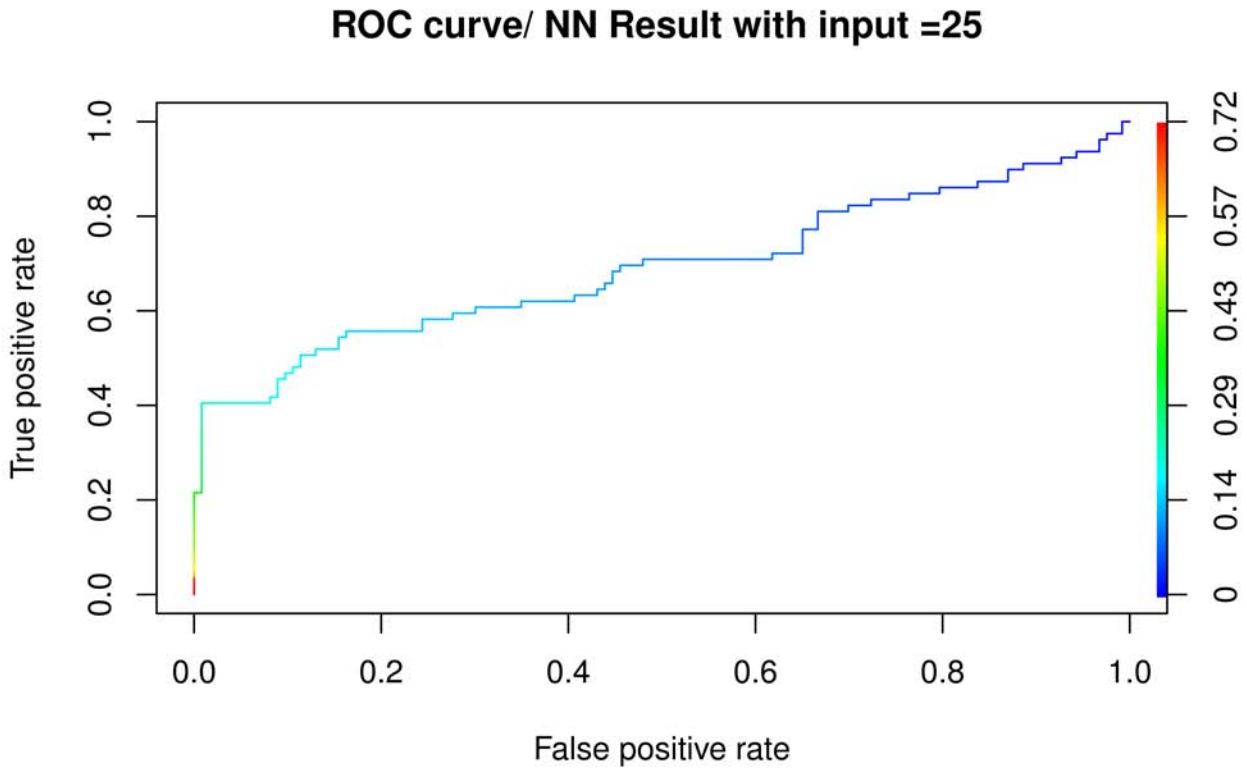
```
## 394      0 -0.1056890417
## 395      0 -0.1688416971
## 396      0 -0.0888405122
## 397      0 -0.1421097078
```

```
roundedresults<-sapply(results,round,digits=0)
roundedresultsdf=data.frame(roundedresults)
attach(roundedresultsdf)
#table(actual,prediction)
```

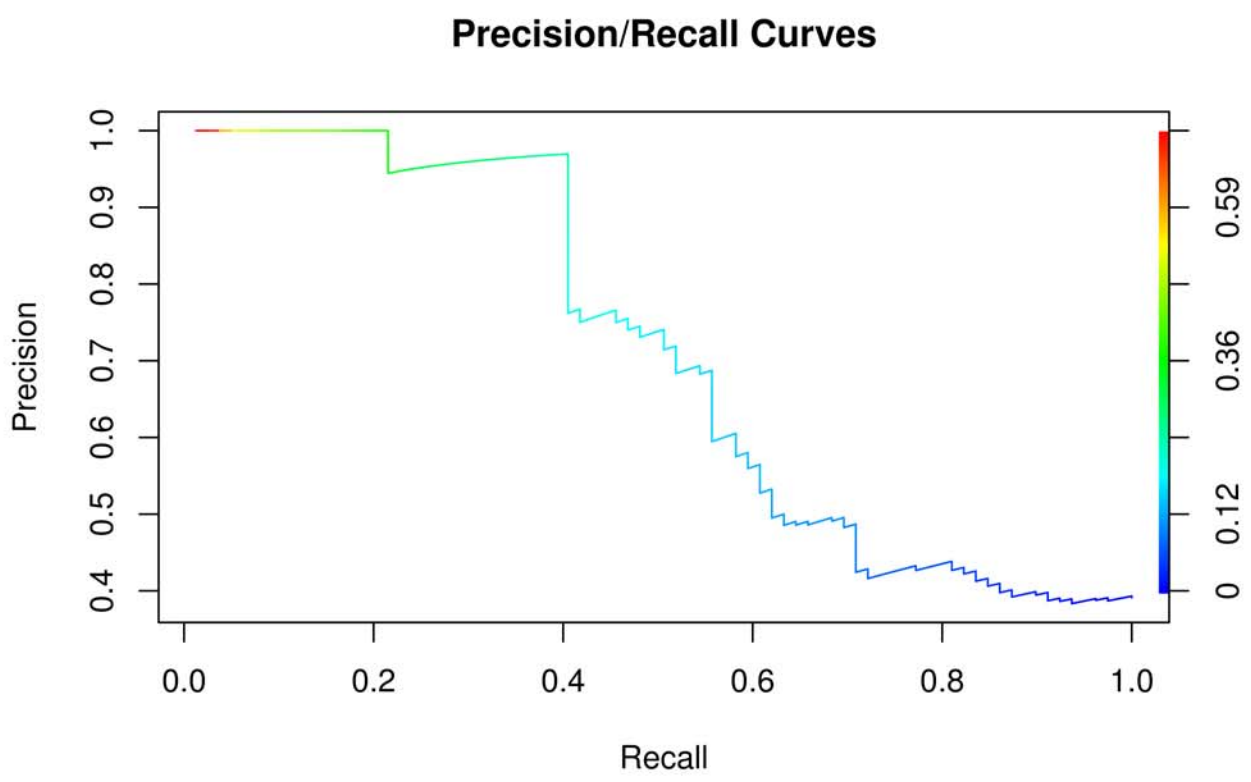
```
detach(package:neuralnet,unload = T)
#dplyr::select(neuralnet)

pr<-nn.results$net.result
pr<-abs(pr)


library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.2.3
```

```
nn.pred = prediction(pr, test_$classification)

pref <- performance(nn.pred, "tpr", "fpr")


pref2 <- performance(nn.pred, "prec", "rec")
plot(pref,colorize=T, main="ROC curve/ NN Result with input =25 " )
```




```
plot(pref2,colorize=T, main="Precision/Recall Curves")
```




```
auc(test_$classification, pr)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response, predictor, auc = TRUE, ...): Deprecated use a
## matrix as predictor. Unexpected results may be produced, please pass a numeric
## vector.
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.6904
```

```
roc_score = roc(test_[, 25], pr)  # AUC score
```

```
## Setting levels: control = 0, case = 1
```

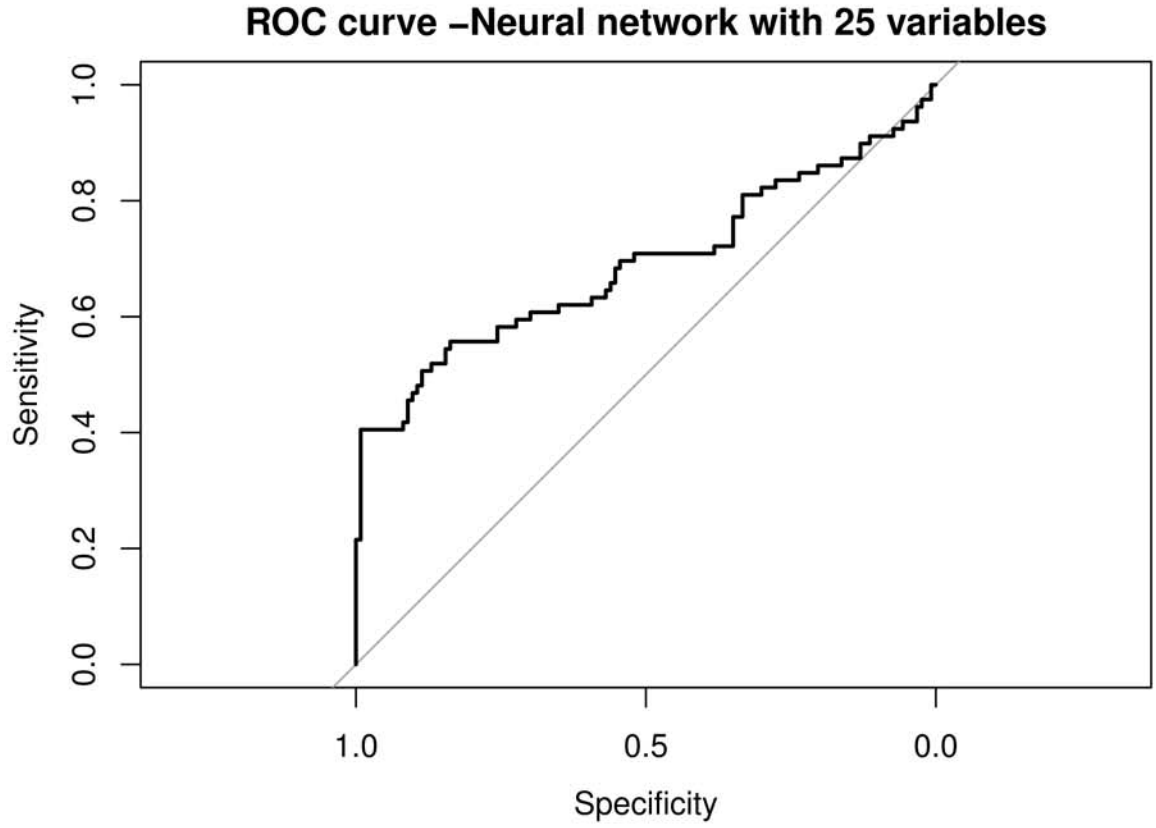
```
## Warning in roc.default(test_[, 25], pr): Deprecated use a matrix as predictor.
## Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```

```
roc_score
```

```
##
## Call:
## roc.default(response = test_[, 25], predictor = pr)
##
## Data: pr in 123 controls (test_[, 25] 0) < 79 cases (test_[, 25] 1).
## Area under the curve: 0.6904
```

```
plot(roc_score,colorsize=T,main="ROC curve -Neural network with 25 variables")
```




```
#confusion matrix
Result<-confusionMatrix(test_$classification, pr)
Result
```

```
##      [,1] [,2]
## [1,]  123   74
## [2,]    0    5
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.