Credit Card Fraud Detection

Akash Barnwal

April 11, 2017

```r
library(ggplot2)
library(readr)
library(caret)
library(e1071)
library(randomForest)
library(ROSE)
```

```r
# Reading the data file from system

setwd("E:/DataScience/Semester 2/Project")

raw_data <- read.csv("creditcard.csv", header=T, na.strings=c(""))

# The Initial Step in Building Successful Predictive Analytics Solutions


# 1) Depicting a random sample of 100 records from the file to understand the data
raw_data_sample <- raw_data[sample(1:nrow(raw_data), 100, replace=FALSE),]

# Checking no of missing values in data
sapply(raw_data,function(x) sum(is.na(x)))
```

```
##   Time     V1     V2     V3     V4     V5     V6     V7     V8     V9
##      0      0      0      0      0      0      0      0      0      0
##    V10    V11    V12    V13    V14    V15    V16    V17    V18    V19
##      0      0      0      0      0      0      0      0      0      0
##    V20    V21    V22    V23    V24    V25    V26    V27    V28 Amount
##      0      0      0      0      0      0      0      0      0      0
##  Class
##      0
```

Result: We can see that there is no missing values in the data

Checking number of unique values in the data

```r
sapply(raw_data, function(x) length(unique(x)))
```

```
##   Time     V1     V2     V3     V4     V5     V6     V7     V8     V9
## 124592 275663 275663 275663 275663 275663 275663 275663 275663 275663
##    V10    V11    V12    V13    V14    V15    V16    V17    V18    V19
## 275663 275663 275663 275663 275663 275663 275663 275663 275663 275663
##    V20    V21    V22    V23    V24    V25    V26    V27    V28 Amount
## 275663 275663 275663 275663 275663 275663 275663 275663 275663  32767
##  Class
##      2
```

Removing the missing values

```
raw_data <- na.omit(raw_data)
raw_data <- raw_data[,c(1,31,30,2:29)]
```

Observations:

1) The data is positively skewed with fraud trasactions very less as compared to the non fraud ones.
2) The total number of fraud transactions are 492 and non fraud ones are 284315. This sort of skewness makes sense since no of fraud data has to be less as compared to the non fraud data.

Inference drawn:

1) Given such imbalance in the data, an algorithm which doesn't do any analysis will give an accuracy of 99.828%. Hence accuracy is not the correct measure of correctness while classifying transactions as fraud and non fraud.

2) Time features shows the chronological order of the transaction hence its not a significant feature to be kept.

```
raw_data <- raw_data[, !(names(raw_data) == "Time")]
```

Converting class into factor for better analysis

```
raw_data$Class <-as.factor(raw_data$Class)
```

```
library(caret)

data=createDataPartition(raw_data$Class,p=.7,list=FALSE)

train_data=raw_data[data,]

test_data=raw_data[-data,]
```

```
table(train_data$Class)
```

```
##
##      0      1
## 199021    345
```

The no of 0's and 1's in train class are 199021 and 345.

```
table(test_data$Class)
```

```
##
##     0     1
## 85294   147
```

The no of 0's and 1's in train class are 85294 and 147 .

```r
no_fraud_rows <- nrow(test_data[test_data$Class == 1,])

rowsTotal <- nrow(raw_data)

LenTestdata <- rowsTotal - nrow(train_data)

nonFraudRows <- LenTestdata - no_fraud_rows
```

Accuracy of a model that predicts all the cases as non-frauds without Modelling

```r
accuracyBase <- nonFraudRows/LenTestdata
accuracyBase
```

```
## [1] 0.9982795
```

```r
# Using Binomial Logistic Regression Algorithm implementation of the R Package.

# Getting the test data and training data for current iteration

# Modelling the data
model <- glm(formula = Class~., family=binomial(link="logit"), data=train_data)
summary(model)
```

```
##
## Call:
## glm(formula = Class ~ ., family = binomial(link = "logit"), data = train_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.9402  -0.0301  -0.0198  -0.0125   4.5830
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.7255519  0.1799423 -48.491  < 2e-16 ***
## Amount       0.0012184  0.0004636   2.628 0.008590 **
## V1           0.1026567  0.0497946   2.062 0.039246 *
## V2           0.0416185  0.0713763   0.583 0.559836
## V3           0.0658746  0.0541594   1.216 0.223867
## V4           0.7023909  0.0887338   7.916 2.46e-15 ***
## V5           0.1315001  0.0784913   1.675 0.093866 .
## V6          -0.0937067  0.0880466  -1.064 0.287200
## V7          -0.0978496  0.0817219  -1.197 0.231170
## V8          -0.1881909  0.0369450  -5.094 3.51e-07 ***
## V9          -0.2084214  0.1338389  -1.557 0.119410
## V10         -0.7716195  0.1199176  -6.435 1.24e-10 ***
## V11          0.0220391  0.0885397   0.249 0.803424
## V12          0.1130481  0.1037457   1.090 0.275860
## V13         -0.3510736  0.0960833  -3.654 0.000258 ***
## V14         -0.6170102  0.0748089  -8.248  < 2e-16 ***
## V15         -0.0276005  0.0982297  -0.281 0.778727
## V16         -0.2389415  0.1458951  -1.638 0.101471
## V17         -0.0523911  0.0814640  -0.643 0.520147
```

```
## V18         -0.0251791  0.1494970  -0.168 0.866249
## V19          0.0688530  0.1120692   0.614 0.538965
## V20         -0.4631656  0.0987279  -4.691 2.71e-06 ***
## V21          0.3214232  0.0696633   4.614 3.95e-06 ***
## V22          0.5196722  0.1520228   3.418 0.000630 ***
## V23         -0.1329690  0.0781332  -1.702 0.088788 .
## V24          0.1715877  0.1734658   0.989 0.322578
## V25          0.0944275  0.1544009   0.612 0.540820
## V26         -0.0725705  0.2228191  -0.326 0.744657
## V27         -0.9082793  0.1468062  -6.187 6.13e-10 ***
## V28         -0.3984403  0.1318328  -3.022 0.002508 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5077.4  on 199365  degrees of freedom
## Residual deviance: 1629.6  on 199336  degrees of freedom
## AIC: 1689.6
##
## Number of Fisher Scoring iterations: 12
```

Inferences:

1) According to the model output, we can see that the significant variables are V4, V8, V10, V13, V14, V20, V21, V22, V27 and v28.

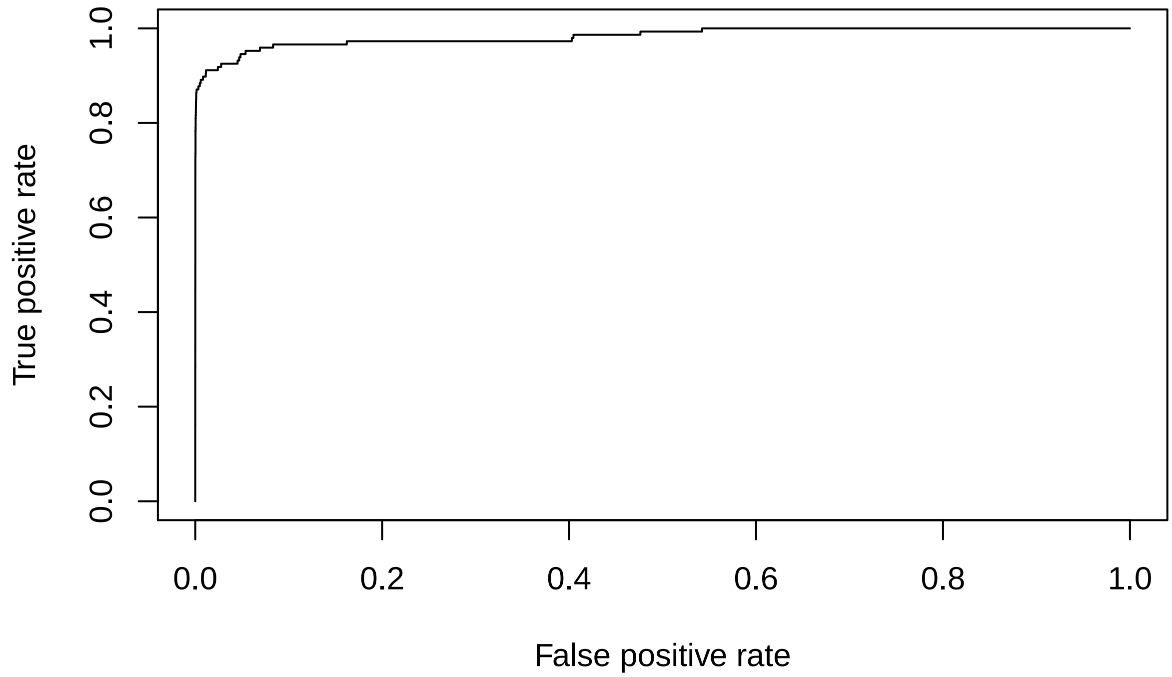
2) The negative coefficient for this predictor suggests that all other variables being equal, the variables with negative coefficient is less likely to have fraud values.

Classifying the test data on the basis of the above model:

In the steps above, we briefly evaluated the fitting of the model, now we would like to see how the model is doing when predicting y on a new set of data.

By setting the parameter type='response', R will output probabilities in the form of P(y=1|X).

```r
p <- predict(model, newdata=subset(test_data), type="response")

library(ROCR)
# Predicting the cutoff probabilities for the predicted values
pr <- prediction(p, test_data$Class)



perf <- performance(pr, "tpr", "fpr")
plot(perf)
```




```r
# Now we can run the anova() function on the model to analyze the table of deviance
anova(model, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: Class
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                  199365     5077.4
## Amount  1     3.87    199364     5073.5 0.0492521 *
## V1      1   516.30    199363     4557.2 < 2.2e-16 ***
## V2      1   503.80    199362     4053.4 < 2.2e-16 ***
## V3      1   669.49    199361     3383.9 < 2.2e-16 ***
## V4      1   726.28    199360     2657.6 < 2.2e-16 ***
## V5      1    17.11    199359     2640.5 3.535e-05 ***
## V6      1    55.87    199358     2584.6 7.735e-14 ***
## V7      1    73.61    199357     2511.0 < 2.2e-16 ***
## V8      1    32.74    199356     2478.3 1.054e-08 ***
## V9      1    34.97    199355     2443.3 3.352e-09 ***
## V10     1   483.70    199354     1959.6 < 2.2e-16 ***
```

```
## V11      1    54.53    199353      1905.1 1.529e-13 ***
## V12      1    26.20    199352      1878.9 3.079e-07 ***
## V13      1    24.92    199351      1854.0 5.985e-07 ***
## V14      1   125.42    199350      1728.6 < 2.2e-16 ***
## V15      1     0.05    199349      1728.5 0.8264386
## V16      1    37.43    199348      1691.1 9.496e-10 ***
## V17      1     4.45    199347      1686.6 0.0348699 *
## V18      1     0.08    199346      1686.6 0.7806547
## V19      1     0.00    199345      1686.6 0.9826669
## V20      1     5.36    199344      1681.2 0.0205852 *
## V21      1     5.26    199343      1675.9 0.0217625 *
## V22      1    10.49    199342      1665.4 0.0011986 **
## V23      1     4.66    199341      1660.8 0.0308632 *
## V24      1     0.63    199340      1660.1 0.4264016
## V25      1     0.21    199339      1659.9 0.6475920
## V26      1     0.30    199338      1659.6 0.5828409
## V27      1    19.09    199337      1640.5 1.248e-05 ***
## V28      1    10.96    199336      1629.6 0.0009326 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Inference:

The difference between the null deviance and the residual deviance shows how our model is doing against the null model (a model with only the intercept). The wider this gap, the better.

A large p-value here indicates that the model without the variable explains more or less the same amount of variation.

While no exact equivalent to the R2 of linear regression exists, the McFadden R2 index can be used to assess the model fit.

```r
library(pscl)
pR2(model)
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -8.147920e+02 -2.538678e+03  3.447772e+03  6.790487e-01  1.714501e-02
##          r2CU
##  6.818197e-01
```

Compute area under the ROC curve (Output of AUC is cutoff-independent)

```r
auc <- performance(pr, measure = "auc")

auc_logistic <- auc@y.values[[1]]
auc_logistic
```

```
## [1] 0.9833206
```

Inference:

An accurancy of 98.70 shows that the model is very good.

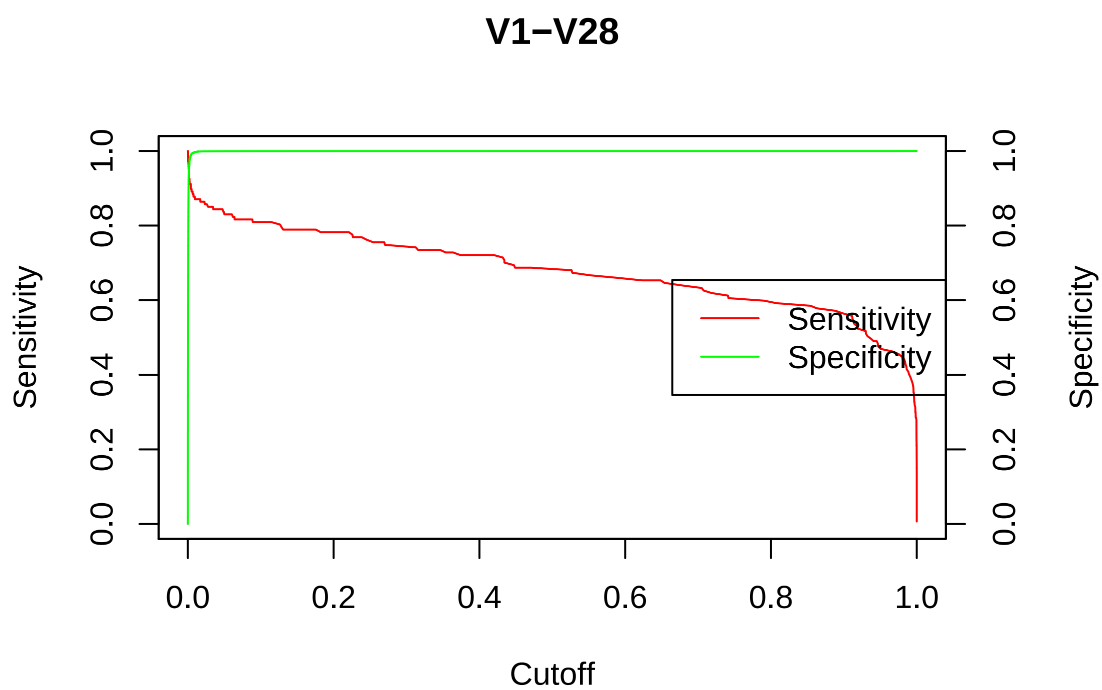
```r
prec_recall <- data.frame(p, test_data$Class)

library(PRROC)
prc <- pr.curve(prec_recall[prec_recall$test_data.Class == 1,]$p, prec_recall[prec_recall$test_data.Clas
prc
```

```
##
##   Precision-recall curve
##
##     Area under curve (Integral):
##      0.7753721
##
##     Area under curve (Davis & Goadrich):
##      0.7753684
##
##     Curve not computed ( can be done by using curve=TRUE )
```

Getting all the values from the prediction model for true positive, true negative, false positive, false negative

```r
cutoffs <- pr@cutoffs[[1]]
tp <- pr@tp[[1]]
tn <- pr@tn[[1]]
fn <- pr@fn[[1]]
fp <- pr@fp[[1]]

maxTpIndex <- 0
maxSpecSensIndex <- 1
maxSensSpecThreshold <- 0
midCutoffIndex <- 0

# finding the cutoff probability by iterating through the cutoff values in the predicted outcome.

for(i in seq_along(cutoffs)) {
  sensitivity <- tp[i]/no_fraud_rows
  specificity <- tn[i]/nonFraudRows
  sensSpecThreshold <- sensitivity + specificity

  if(sensSpecThreshold > maxSensSpecThreshold)

    {
    maxSensSpecThreshold <- sensSpecThreshold
    maxSpecSensIndex <- i
    }

  if(sensitivity == 1 && maxTpIndex == 0){
    maxTpIndex <- i
  }

  if(cutoffs[i][[1]] < 0.5 && midCutoffIndex == 0) {
    midCutoffIndex <- i
  }
}
```

```
# Graph data for plotting the sensitivity and specificity curves and saving the plot

graph_x = cutoffs
graph_y1 = tp/no_fraud_rows
graph_y2 = tn/nonFraudRows

par(mar=c(6,5,5,6)+0.5)
plot(graph_x,graph_y1,type="l",col="red",yaxt="n",xlab="",ylab="", main="V1-V28")
axis(2)
par(new=TRUE)
plot(graph_x, graph_y2,type="l",col="green",xaxt="n",yaxt="n",xlab="",ylab="")
axis(4)
mtext("Specificity",side=4,line=3)
mtext("Sensitivity",side=2,line=3)
mtext("Cutoff",side=1,line=3)
legend("right",col=c("red","green"),lty=1,legend=c("Sensitivity","Specificity"))
```



Testing the model using concordance- discordance pair test

```
concordance<-function(model){
      # Get all actual observations and their fitted values into a frame
      fitted<-data.frame(cbind(model$y,model$fitted.values))
        colnames(fitted)<-c('respvar','score')
          # Subset only ones
          ones<-fitted[fitted[,1]==1,]
            # Subset only zeros
```

```r
            zeros<-fitted[fitted[,1]==0,]

              pairs_tested<-0
                conc<-0
                  disc<-0
                    ties<-0

                      # Get the values in a for-loop
                      for(i in 1:nrow(ones))
                      {
                            for(j in 1:nrow(zeros))
                            {
                                  pairs_tested<-pairs_tested+1
                                    if(ones[i,2]>zeros[j,2]) {conc<-conc+1}
                                    else if(ones[i,2]==zeros[j,2]){ties<-ties+1}
                                    else {disc<-disc+1}
                                  }
                          }
                      # Calculate concordance, discordance and ties
                      concordance<-conc/pairs_tested
                        discordance<-disc/pairs_tested
                          ties_perc<-ties/pairs_tested
                            return(list("Concordance"=concordance,
                                                                  "Discordance"=discordance,
                                                                  "Tied"=ties_perc,
                                                                  "Pairs"=pairs_tested))
                          }

  concordance(model)
```

```
## $Concordance
## [1] 0.9757272
##
## $Discordance
## [1] 0.02427284
##
## $Tied
## [1] 0
##
## $Pairs
## [1] 68662245
```

Down sampling the positive data samples to avoid data imbalance

```r
data_balanced_under <- ovun.sample(Class ~ ., data = train_data, method = "under", N = 600, seed = 1)$da
table(data_balanced_under$Class)
```

```
##
##   0   1
## 255 345
```

Prediction using different Models

SVM Model

```
model_svm <- train(Class~.,data=data_balanced_under,method="svmRadial",trControl=trainControl(method='c

pred_svm <- predict(model_svm, test_data)
cm_svm <- confusionMatrix(pred_svm,test_data$Class, positive = "0")
cm_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 81006    13
##          1  4288   134
##
##                Accuracy : 0.9497
##                  95% CI : (0.9482, 0.9511)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0555
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.9497
##             Specificity : 0.9116
##          Pos Pred Value : 0.9998
##          Neg Pred Value : 0.0303
##              Prevalence : 0.9983
##          Detection Rate : 0.9481
##    Detection Prevalence : 0.9482
##       Balanced Accuracy : 0.9306
##
##        'Positive' Class : 0
##
```

```
acc_svm <- cm_svm$overall['Accuracy']
acc_svm
```

```
##  Accuracy
## 0.9496612
```

Random Forest

```
model_rf <- train(Class~.,data=data_balanced_under,method="ranger",trControl=trainControl(method='cv'),
pred_rf <- predict(model_rf, test_data)
cm_rf <-confusionMatrix(pred_rf,test_data$Class, positive = "0")
acc_rf <- cm_rf$overall['Accuracy']
acc_rf
```

```
##  Accuracy
## 0.9572102
```

KNN

```
model_knn <- train(Class~.,data=data_balanced_under,method="knn",trControl=trainControl(method='cv'),pre
pred_knn <- predict(model_knn, test_data)
cm_knn <- confusionMatrix(pred_knn,test_data$Class, positive = "0")
acc_knn <- cm_knn$overall['Accuracy']
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 82065    11
##          1  3229   136
##
##                Accuracy : 0.9621
##                  95% CI : (0.9608, 0.9633)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0744
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96214
##             Specificity : 0.92517
##          Pos Pred Value : 0.99987
##          Neg Pred Value : 0.04042
##              Prevalence : 0.99828
##          Detection Rate : 0.96049
##    Detection Prevalence : 0.96062
##       Balanced Accuracy : 0.94366
##
##        'Positive' Class : 0
##
```

Neural net

```
model_nn <- train(Class~.,data=data_balanced_under,method="nnet",trControl=trainControl(method='cv'))
```

```
## # weights:  32
## initial  value 391.296805
## iter  10 value 293.441768
## iter  20 value 240.557865
## iter  30 value 213.154505
## iter  40 value 211.581226
## iter  50 value 201.362830
## iter  60 value 201.269414
## iter  70 value 201.234753
## iter  80 value 200.048079
## iter  90 value 199.939237
## iter 100 value 157.539864
## final  value 157.539864
## stopped after 100 iterations
```

```
## # weights:  94
## initial  value 411.092781
## iter  10 value 150.247502
## iter  20 value 101.234982
## iter  30 value 64.921087
## iter  40 value 48.568832
## iter  50 value 41.224987
## iter  60 value 39.378314
## iter  70 value 36.692682
## iter  80 value 36.127340
## iter  90 value 35.872243
## iter 100 value 35.845105
## final  value 35.845105
## stopped after 100 iterations
## # weights:  156
## initial  value 352.331506
## iter  10 value 110.119543
## iter  20 value 86.926027
## iter  30 value 81.735885
## iter  40 value 80.529523
## iter  50 value 80.512594
## iter  60 value 80.509833
## iter  70 value 78.108555
## iter  80 value 78.107549
## iter  90 value 78.106870
## iter 100 value 78.106582
## final  value 78.106582
## stopped after 100 iterations
## # weights:  32
## initial  value 406.330827
## iter  10 value 235.463037
## iter  20 value 210.106134
## iter  30 value 158.895234
## iter  40 value 107.381958
## iter  50 value 92.082277
## iter  60 value 82.538649
## iter  70 value 76.712124
## iter  80 value 75.123077
## iter  90 value 75.009534
## iter 100 value 75.007855
## final  value 75.007855
## stopped after 100 iterations
## # weights:  94
## initial  value 499.419990
## iter  10 value 226.915665
## iter  20 value 175.206065
## iter  30 value 97.112324
## iter  40 value 66.535971
## iter  50 value 56.149552
## iter  60 value 48.731963
## iter  70 value 45.523552
## iter  80 value 45.018820
## iter  90 value 45.011298
## iter 100 value 45.010999
```

```
## final  value 45.010999
## stopped after 100 iterations
## # weights:  156
## initial  value 382.194052
## iter  10 value 196.367870
## iter  20 value 148.175475
## iter  30 value 77.145318
## iter  40 value 53.884470
## iter  50 value 44.191763
## iter  60 value 42.452878
## iter  70 value 42.003624
## iter  80 value 41.364005
## iter  90 value 40.170604
## iter 100 value 39.302304
## final  value 39.302304
## stopped after 100 iterations
## # weights:  32
## initial  value 373.377886
## iter  10 value 186.798616
## iter  20 value 143.379142
## iter  30 value 111.077971
## iter  40 value 97.686473
## iter  50 value 88.465286
## iter  60 value 83.577919
## iter  70 value 82.580124
## iter  80 value 82.533686
## iter  90 value 82.523959
## iter 100 value 82.517991
## final  value 82.517991
## stopped after 100 iterations
## # weights:  94
## initial  value 360.733883
## iter  10 value 111.020406
## iter  20 value 56.213067
## iter  30 value 27.512589
## iter  40 value 18.800725
## iter  50 value 16.381081
## iter  60 value 16.226696
## iter  70 value 16.100131
## iter  80 value 16.044198
## iter  90 value 16.010642
## iter 100 value 15.559311
## final  value 15.559311
## stopped after 100 iterations
## # weights:  156
## initial  value 407.011051
## iter  10 value 169.574838
## iter  20 value 76.811290
## iter  30 value 40.877956
## iter  40 value 31.128958
## iter  50 value 23.916466
## iter  60 value 22.804289
## iter  70 value 20.862337
## iter  80 value 20.470119
```

```
## iter  90 value 19.990423
## iter 100 value 19.933224
## final  value 19.933224
## stopped after 100 iterations
## # weights:  32
## initial  value 363.812725
## iter  10 value 182.536936
## iter  20 value 161.887236
## iter  30 value 158.754503
## iter  40 value 157.209041
## iter  50 value 156.258425
## iter  60 value 155.618722
## iter  70 value 155.564981
## iter  80 value 155.564329
## iter  80 value 155.564328
## final  value 155.564321
## converged
## # weights:  94
## initial  value 491.000170
## iter  10 value 157.051731
## iter  20 value 74.416578
## iter  30 value 38.045024
## iter  40 value 28.862812
## iter  50 value 26.254143
## iter  60 value 25.306686
## iter  70 value 25.153987
## iter  80 value 25.128030
## iter  90 value 25.121976
## iter 100 value 25.121873
## final  value 25.121873
## stopped after 100 iterations
## # weights:  156
## initial  value 408.340602
## iter  10 value 137.335771
## iter  20 value 90.185840
## iter  30 value 70.924779
## iter  40 value 61.556001
## iter  50 value 27.301480
## iter  60 value 16.822676
## iter  70 value 13.974793
## iter  80 value 12.837205
## iter  90 value 12.004268
## iter 100 value 11.752366
## final  value 11.752366
## stopped after 100 iterations
## # weights:  32
## initial  value 380.384292
## iter  10 value 295.376484
## iter  20 value 165.506210
## iter  30 value 104.818968
## iter  40 value 87.385356
## iter  50 value 74.842022
## iter  60 value 73.000101
## iter  70 value 72.937408
```

```
## final  value 72.937205
## converged
## # weights:  94
## initial  value 384.648916
## iter  10 value 131.103479
## iter  20 value 85.737176
## iter  30 value 69.559428
## iter  40 value 52.640450
## iter  50 value 43.340017
## iter  60 value 38.717053
## iter  70 value 37.420194
## iter  80 value 36.756834
## iter  90 value 36.490869
## iter 100 value 36.488387
## final  value 36.488387
## stopped after 100 iterations
## # weights:  156
## initial  value 474.868376
## iter  10 value 171.679616
## iter  20 value 113.997900
## iter  30 value 76.691602
## iter  40 value 55.771475
## iter  50 value 43.339200
## iter  60 value 41.305333
## iter  70 value 40.791341
## iter  80 value 40.356439
## iter  90 value 40.253648
## iter 100 value 40.247550
## final  value 40.247550
## stopped after 100 iterations
## # weights:  32
## initial  value 381.351642
## iter  10 value 246.272048
## iter  20 value 143.152245
## iter  30 value 122.598987
## iter  40 value 110.236783
## iter  50 value 105.905021
## iter  60 value 103.892975
## iter  70 value 97.991962
## iter  80 value 97.964072
## iter  90 value 97.950852
## iter 100 value 95.622667
## final  value 95.622667
## stopped after 100 iterations
## # weights:  94
## initial  value 362.567503
## iter  10 value 184.046491
## iter  20 value 154.400554
## iter  30 value 134.099498
## iter  40 value 108.708158
## iter  50 value 87.699749
## iter  60 value 71.082253
## iter  70 value 62.505070
## iter  80 value 54.331865
```

```
## iter  90 value 52.909188
## iter 100 value 50.212372
## final  value 50.212372
## stopped after 100 iterations
## # weights:  156
## initial  value 362.608138
## iter  10 value 169.269325
## iter  20 value 110.510455
## iter  30 value 47.399758
## iter  40 value 26.680297
## iter  50 value 24.339198
## iter  60 value 23.947931
## iter  70 value 23.839305
## iter  80 value 23.587934
## iter  90 value 23.480921
## iter 100 value 23.218557
## final  value 23.218557
## stopped after 100 iterations
## # weights:  32
## initial  value 390.799926
## iter  10 value 184.700796
## iter  20 value 102.168128
## iter  30 value 87.956256
## iter  40 value 80.493017
## iter  50 value 80.211976
## iter  60 value 80.163186
## iter  70 value 80.153331
## iter  80 value 80.150789
## final  value 80.150463
## converged
## # weights:  94
## initial  value 357.889600
## iter  10 value 237.066519
## iter  20 value 118.394112
## iter  30 value 83.607335
## iter  40 value 74.617759
## iter  50 value 65.695026
## iter  60 value 58.361215
## iter  70 value 32.901376
## iter  80 value 20.121219
## iter  90 value 17.376991
## iter 100 value 16.156106
## final  value 16.156106
## stopped after 100 iterations
## # weights:  156
## initial  value 384.675022
## iter  10 value 134.058980
## iter  20 value 75.011173
## iter  30 value 34.111042
## iter  40 value 17.700961
## iter  50 value 15.237879
## iter  60 value 15.151650
## iter  70 value 15.146076
## iter  80 value 15.144913
```

```
## final  value 15.144799
## converged
## # weights:  32
## initial  value 366.832817
## iter  10 value 176.458900
## iter  20 value 113.563537
## iter  30 value 96.576082
## iter  40 value 89.022267
## iter  50 value 77.361508
## iter  60 value 73.876153
## iter  70 value 73.518855
## iter  80 value 73.513034
## iter  90 value 73.512599
## final  value 73.512594
## converged
## # weights:  94
## initial  value 448.219073
## iter  10 value 230.326906
## iter  20 value 126.736577
## iter  30 value 115.962554
## iter  40 value 109.137267
## iter  50 value 78.612616
## iter  60 value 56.786159
## iter  70 value 51.598117
## iter  80 value 49.051987
## iter  90 value 46.846937
## iter 100 value 46.189815
## final  value 46.189815
## stopped after 100 iterations
## # weights:  156
## initial  value 373.684514
## iter  10 value 165.363061
## iter  20 value 96.487459
## iter  30 value 64.478526
## iter  40 value 52.041303
## iter  50 value 44.391976
## iter  60 value 40.430314
## iter  70 value 38.794144
## iter  80 value 38.632933
## iter  90 value 38.513471
## iter 100 value 38.504026
## final  value 38.504026
## stopped after 100 iterations
## # weights:  32
## initial  value 469.902726
## iter  10 value 139.038803
## iter  20 value 118.684056
## iter  30 value 117.935002
## iter  40 value 117.921268
## iter  50 value 117.905828
## iter  60 value 117.880904
## iter  70 value 117.851896
## iter  80 value 117.509702
## iter  90 value 116.100574
```

```
## iter 100 value 105.097582
## final  value 105.097582
## stopped after 100 iterations
## # weights:  94
## initial  value 376.251946
## iter  10 value 190.012965
## iter  20 value 94.128855
## iter  30 value 75.825051
## iter  40 value 66.799691
## iter  50 value 60.816166
## iter  60 value 58.945316
## iter  70 value 54.409848
## iter  80 value 51.002902
## iter  90 value 47.239904
## iter 100 value 43.341522
## final  value 43.341522
## stopped after 100 iterations
## # weights:  156
## initial  value 553.778154
## iter  10 value 160.607560
## iter  20 value 77.655230
## iter  30 value 43.547747
## iter  40 value 35.079139
## iter  50 value 31.642370
## iter  60 value 26.985406
## iter  70 value 25.188420
## iter  80 value 23.580023
## iter  90 value 17.281362
## iter 100 value 8.193755
## final  value 8.193755
## stopped after 100 iterations
## # weights:  32
## initial  value 342.742245
## iter  10 value 195.287783
## iter  20 value 186.418130
## iter  30 value 171.286264
## iter  40 value 171.148744
## iter  50 value 171.131178
## iter  60 value 171.129026
## final  value 171.128902
## converged
## # weights:  94
## initial  value 436.191439
## iter  10 value 155.455278
## iter  20 value 89.921224
## iter  30 value 62.926665
## iter  40 value 46.101286
## iter  50 value 36.349243
## iter  60 value 30.542882
## iter  70 value 22.786164
## iter  80 value 19.520708
## iter  90 value 18.357636
## iter 100 value 18.221853
## final  value 18.221853
```

```
## stopped after 100 iterations
## # weights:  156
## initial  value 369.059464
## iter  10 value 150.487963
## iter  20 value 90.324264
## iter  30 value 49.292250
## iter  40 value 27.182214
## iter  50 value 21.272650
## iter  60 value 17.543219
## iter  70 value 15.924834
## iter  80 value 14.566175
## iter  90 value 14.154773
## iter 100 value 14.145598
## final  value 14.145598
## stopped after 100 iterations
## # weights:  32
## initial  value 419.150776
## iter  10 value 313.870820
## iter  20 value 235.244316
## iter  30 value 177.702117
## iter  40 value 116.621511
## iter  50 value 100.313773
## iter  60 value 95.779679
## iter  70 value 90.841313
## iter  80 value 79.735963
## iter  90 value 74.074979
## iter 100 value 71.483788
## final  value 71.483788
## stopped after 100 iterations
## # weights:  94
## initial  value 410.963757
## iter  10 value 243.134966
## iter  20 value 103.992888
## iter  30 value 75.815040
## iter  40 value 58.371684
## iter  50 value 51.690293
## iter  60 value 49.495505
## iter  70 value 48.768525
## iter  80 value 42.950199
## iter  90 value 38.566732
## iter 100 value 37.527713
## final  value 37.527713
## stopped after 100 iterations
## # weights:  156
## initial  value 446.369411
## iter  10 value 174.136351
## iter  20 value 136.258323
## iter  30 value 95.919570
## iter  40 value 70.035578
## iter  50 value 52.404670
## iter  60 value 45.791778
## iter  70 value 43.567849
## iter  80 value 39.408461
## iter  90 value 36.062899
```

```
## iter 100 value 34.260702
## final  value 34.260702
## stopped after 100 iterations
## # weights:  32
## initial  value 401.390434
## iter  10 value 232.375338
## iter  20 value 149.413304
## iter  30 value 95.659530
## iter  40 value 85.540563
## iter  50 value 84.047284
## iter  60 value 82.259901
## iter  70 value 82.007792
## iter  80 value 81.967930
## iter  90 value 81.963202
## iter 100 value 81.961208
## final  value 81.961208
## stopped after 100 iterations
## # weights:  94
## initial  value 396.866842
## iter  10 value 185.239696
## iter  20 value 157.523567
## iter  30 value 151.358058
## iter  40 value 151.216025
## iter  50 value 151.194485
## iter  60 value 151.172485
## iter  70 value 151.156707
## iter  80 value 151.149968
## iter  90 value 151.143782
## iter 100 value 151.138393
## final  value 151.138393
## stopped after 100 iterations
## # weights:  156
## initial  value 339.893804
## iter  10 value 111.020468
## iter  20 value 72.348135
## iter  30 value 28.283818
## iter  40 value 12.829332
## iter  50 value 11.199379
## iter  60 value 11.163002
## iter  70 value 11.138338
## iter  80 value 11.110121
## iter  90 value 10.953102
## iter 100 value 10.919973
## final  value 10.919973
## stopped after 100 iterations
## # weights:  32
## initial  value 400.681376
## iter  10 value 130.405429
## iter  20 value 114.400953
## iter  30 value 111.510426
## iter  40 value 111.247012
## iter  50 value 103.352911
## iter  60 value 103.165396
## iter  70 value 99.673866
```

```
## iter  80 value 99.672742
## iter  90 value 95.544090
## iter 100 value 91.894309
## final  value 91.894309
## stopped after 100 iterations
## # weights:  94
## initial  value 406.452820
## iter  10 value 159.210185
## iter  20 value 86.745041
## iter  30 value 58.649360
## iter  40 value 44.871223
## iter  50 value 38.523598
## iter  60 value 35.968389
## iter  70 value 35.728565
## iter  80 value 34.975593
## iter  90 value 34.964263
## iter 100 value 34.962145
## final  value 34.962145
## stopped after 100 iterations
## # weights:  156
## initial  value 460.000624
## iter  10 value 139.072908
## iter  20 value 95.163005
## iter  30 value 44.844245
## iter  40 value 36.999666
## iter  50 value 33.142980
## iter  60 value 27.154882
## iter  70 value 22.595908
## iter  80 value 21.185979
## iter  90 value 18.526503
## iter 100 value 17.880049
## final  value 17.880049
## stopped after 100 iterations
## # weights:  32
## initial  value 397.002368
## iter  10 value 276.827574
## iter  20 value 261.350774
## iter  30 value 258.332151
## iter  40 value 204.507359
## iter  50 value 142.392837
## iter  60 value 123.575409
## iter  70 value 116.836379
## iter  80 value 93.168463
## iter  90 value 81.224813
## iter 100 value 73.083611
## final  value 73.083611
## stopped after 100 iterations
## # weights:  94
## initial  value 372.037273
## iter  10 value 188.747546
## iter  20 value 95.244354
## iter  30 value 64.571764
## iter  40 value 51.859032
## iter  50 value 45.884085
```

```
## iter  60 value 44.051649
## iter  70 value 43.117210
## iter  80 value 42.804846
## iter  90 value 42.564206
## iter 100 value 42.419537
## final  value 42.419537
## stopped after 100 iterations
## # weights:  156
## initial  value 432.258755
## iter  10 value 148.590938
## iter  20 value 94.851329
## iter  30 value 71.567616
## iter  40 value 49.667262
## iter  50 value 43.158304
## iter  60 value 41.335456
## iter  70 value 38.733182
## iter  80 value 37.300664
## iter  90 value 36.775854
## iter 100 value 35.915607
## final  value 35.915607
## stopped after 100 iterations
## # weights:  32
## initial  value 368.425891
## iter  10 value 196.761856
## iter  20 value 170.585009
## iter  30 value 170.380465
## iter  40 value 169.693474
## iter  50 value 168.931910
## iter  60 value 168.897567
## iter  70 value 168.888609
## iter  80 value 168.884749
## iter  90 value 168.882512
## iter 100 value 168.880990
## final  value 168.880990
## stopped after 100 iterations
## # weights:  94
## initial  value 371.097231
## iter  10 value 179.458315
## iter  20 value 65.368950
## iter  30 value 42.210582
## iter  40 value 32.527304
## iter  50 value 30.661305
## iter  60 value 23.383955
## iter  70 value 22.274959
## iter  80 value 22.199317
## iter  90 value 22.184773
## iter 100 value 22.176942
## final  value 22.176942
## stopped after 100 iterations
## # weights:  156
## initial  value 349.482970
## iter  10 value 110.003266
## iter  20 value 67.854221
## iter  30 value 51.185172
```

```
## iter  40 value 48.291800
## iter  50 value 34.192566
## iter  60 value 21.834155
## iter  70 value 18.817056
## iter  80 value 15.601220
## iter  90 value 15.195641
## iter 100 value 13.553316
## final  value 13.553316
## stopped after 100 iterations
## # weights:  32
## initial  value 369.764096
## iter  10 value 118.104530
## iter  20 value 83.214659
## iter  30 value 73.852002
## iter  40 value 62.123509
## iter  50 value 56.388476
## iter  60 value 51.071989
## iter  70 value 47.416164
## iter  80 value 46.962668
## iter  90 value 46.508647
## iter 100 value 46.235590
## final  value 46.235590
## stopped after 100 iterations
## # weights:  94
## initial  value 418.238205
## iter  10 value 142.315862
## iter  20 value 57.848970
## iter  30 value 34.913767
## iter  40 value 21.905929
## iter  50 value 13.127472
## iter  60 value 9.157382
## iter  70 value 8.458650
## iter  80 value 8.101984
## iter  90 value 7.894909
## iter 100 value 7.623284
## final  value 7.623284
## stopped after 100 iterations
## # weights:  156
## initial  value 375.635546
## iter  10 value 163.328465
## iter  20 value 79.565757
## iter  30 value 47.218157
## iter  40 value 40.446852
## iter  50 value 34.865932
## iter  60 value 30.412647
## iter  70 value 27.725773
## iter  80 value 22.635016
## iter  90 value 18.569862
## iter 100 value 17.249112
## final  value 17.249112
## stopped after 100 iterations
## # weights:  32
## initial  value 379.920290
## iter  10 value 123.379364
```

```
## iter  20 value 93.625775
## iter  30 value 80.668237
## iter  40 value 67.162208
## iter  50 value 63.702287
## iter  60 value 62.041350
## iter  70 value 61.986253
## final  value 61.985900
## converged
## # weights:  94
## initial  value 368.900877
## iter  10 value 138.127784
## iter  20 value 79.971471
## iter  30 value 59.069750
## iter  40 value 54.179162
## iter  50 value 53.222301
## iter  60 value 53.032051
## iter  70 value 52.821821
## iter  80 value 52.625461
## iter  90 value 52.533270
## final  value 52.533061
## converged
## # weights:  156
## initial  value 360.590009
## iter  10 value 108.747785
## iter  20 value 69.061006
## iter  30 value 59.770913
## iter  40 value 56.949209
## iter  50 value 51.812922
## iter  60 value 46.657292
## iter  70 value 44.732139
## iter  80 value 44.266758
## iter  90 value 43.755506
## iter 100 value 43.317227
## final  value 43.317227
## stopped after 100 iterations
## # weights:  32
## initial  value 368.407135
## iter  10 value 193.017270
## iter  20 value 175.211972
## iter  30 value 157.340659
## iter  40 value 139.266933
## iter  50 value 132.611576
## iter  60 value 129.842849
## iter  70 value 129.764244
## iter  80 value 129.754401
## iter  90 value 104.785428
## iter 100 value 80.662330
## final  value 80.662330
## stopped after 100 iterations
## # weights:  94
## initial  value 384.149933
## iter  10 value 116.927695
## iter  20 value 102.344135
## iter  30 value 87.938631
```

```
## iter  40 value 76.662969
## iter  50 value 65.611768
## iter  60 value 60.358808
## iter  70 value 56.782286
## iter  80 value 56.616146
## iter  90 value 56.451715
## iter 100 value 55.759890
## final  value 55.759890
## stopped after 100 iterations
## # weights:  156
## initial  value 535.472939
## iter  10 value 148.155474
## iter  20 value 97.144273
## iter  30 value 46.013780
## iter  40 value 31.169416
## iter  50 value 25.396085
## iter  60 value 22.901186
## iter  70 value 19.759923
## iter  80 value 18.709347
## iter  90 value 18.512746
## iter 100 value 18.122200
## final  value 18.122200
## stopped after 100 iterations
## # weights:  32
## initial  value 398.596942
## iter  10 value 151.830313
## iter  20 value 135.352765
## iter  30 value 134.807701
## iter  40 value 134.796354
## iter  50 value 134.795515
## iter  60 value 134.794932
## final  value 134.794880
## converged
## # weights:  94
## initial  value 456.543739
## iter  10 value 201.468881
## iter  20 value 140.240502
## iter  30 value 82.194210
## iter  40 value 62.368442
## iter  50 value 59.867478
## iter  60 value 55.649279
## iter  70 value 49.268213
## iter  80 value 46.374568
## iter  90 value 46.279441
## iter 100 value 46.245012
## final  value 46.245012
## stopped after 100 iterations
## # weights:  156
## initial  value 409.900104
## iter  10 value 143.845312
## iter  20 value 83.985881
## iter  30 value 49.928482
## iter  40 value 43.906002
## iter  50 value 40.405045
```

```
## iter  60 value 36.165376
## iter  70 value 25.533509
## iter  80 value 19.163559
## iter  90 value 13.861665
## iter 100 value 9.577084
## final  value 9.577084
## stopped after 100 iterations
## # weights:  32
## initial  value 438.722304
## iter  10 value 295.665942
## iter  20 value 152.567668
## iter  30 value 140.132053
## iter  40 value 124.938360
## iter  50 value 117.473355
## iter  60 value 94.112051
## iter  70 value 77.532902
## iter  80 value 72.306792
## iter  90 value 68.024815
## iter 100 value 66.517229
## final  value 66.517229
## stopped after 100 iterations
## # weights:  94
## initial  value 389.565421
## iter  10 value 133.953619
## iter  20 value 97.424478
## iter  30 value 70.019634
## iter  40 value 55.680047
## iter  50 value 53.088348
## iter  60 value 48.462265
## iter  70 value 46.127149
## iter  80 value 44.685685
## iter  90 value 44.412404
## iter 100 value 44.405070
## final  value 44.405070
## stopped after 100 iterations
## # weights:  156
## initial  value 429.214432
## iter  10 value 111.744039
## iter  20 value 81.545903
## iter  30 value 65.344544
## iter  40 value 49.282342
## iter  50 value 43.298217
## iter  60 value 38.899944
## iter  70 value 35.633189
## iter  80 value 32.342061
## iter  90 value 31.613076
## iter 100 value 31.097094
## final  value 31.097094
## stopped after 100 iterations
## # weights:  32
## initial  value 406.982992
## iter  10 value 254.922475
## iter  20 value 248.450082
## iter  30 value 248.206261
```

```
## iter  40 value 247.269110
## iter  50 value 235.732049
## iter  60 value 201.726397
## iter  70 value 173.573384
## iter  80 value 164.879274
## iter  90 value 164.867180
## iter 100 value 163.046763
## final  value 163.046763
## stopped after 100 iterations
## # weights:  94
## initial  value 379.974862
## iter  10 value 170.728321
## iter  20 value 76.911973
## iter  30 value 45.505216
## iter  40 value 33.399922
## iter  50 value 27.792958
## iter  60 value 25.524806
## iter  70 value 25.408897
## iter  80 value 25.340088
## iter  90 value 25.226158
## iter 100 value 25.167851
## final  value 25.167851
## stopped after 100 iterations
## # weights:  156
## initial  value 470.659040
## iter  10 value 196.465946
## iter  20 value 54.598161
## iter  30 value 37.528759
## iter  40 value 27.659807
## iter  50 value 23.302960
## iter  60 value 22.864173
## iter  70 value 21.240233
## iter  80 value 17.884248
## iter  90 value 17.473890
## iter 100 value 17.441140
## final  value 17.441140
## stopped after 100 iterations
## # weights:  32
## initial  value 364.509064
## iter  10 value 255.559147
## iter  20 value 247.882732
## iter  30 value 247.819431
## final  value 247.819397
## converged
## # weights:  94
## initial  value 515.507267
## iter  10 value 268.677074
## iter  20 value 242.401774
## iter  30 value 235.698029
## iter  40 value 191.395747
## iter  50 value 190.374352
## iter  60 value 156.099435
## iter  70 value 147.120093
## iter  80 value 132.096526
```

```
## iter  90 value 125.217285
## iter 100 value 123.468350
## final  value 123.468350
## stopped after 100 iterations
## # weights:  156
## initial  value 434.501055
## iter  10 value 102.686695
## iter  20 value 65.349000
## iter  30 value 27.835431
## iter  40 value 18.836561
## iter  50 value 15.882097
## iter  60 value 13.052346
## iter  70 value 10.692133
## iter  80 value 10.442712
## iter  90 value 10.431907
## iter 100 value 10.431123
## final  value 10.431123
## stopped after 100 iterations
## # weights:  32
## initial  value 373.966552
## iter  10 value 285.109280
## iter  20 value 271.596214
## iter  30 value 167.159614
## iter  40 value 151.681456
## iter  50 value 144.383130
## iter  60 value 124.805686
## iter  70 value 116.348200
## iter  80 value 110.736256
## iter  90 value 107.971217
## iter 100 value 101.189333
## final  value 101.189333
## stopped after 100 iterations
## # weights:  94
## initial  value 367.222085
## iter  10 value 154.451816
## iter  20 value 95.662777
## iter  30 value 76.951954
## iter  40 value 67.980086
## iter  50 value 60.847134
## iter  60 value 53.532453
## iter  70 value 50.177483
## iter  80 value 48.583270
## iter  90 value 48.380001
## iter 100 value 48.197181
## final  value 48.197181
## stopped after 100 iterations
## # weights:  156
## initial  value 337.920080
## iter  10 value 176.588783
## iter  20 value 96.927155
## iter  30 value 61.446765
## iter  40 value 46.589436
## iter  50 value 42.354049
## iter  60 value 39.732181
```

```
## iter  70 value 39.083045
## iter  80 value 38.964083
## iter  90 value 38.890295
## iter 100 value 38.359903
## final  value 38.359903
## stopped after 100 iterations
## # weights:  32
## initial  value 368.576902
## iter  10 value 179.035286
## iter  20 value 149.242982
## iter  30 value 130.810647
## iter  40 value 110.371139
## iter  50 value 98.761246
## iter  60 value 97.319913
## iter  70 value 97.291200
## iter  80 value 97.284355
## iter  90 value 97.278759
## iter 100 value 97.275958
## final  value 97.275958
## stopped after 100 iterations
## # weights:  94
## initial  value 411.526069
## iter  10 value 175.957679
## iter  20 value 90.344100
## iter  30 value 48.840737
## iter  40 value 43.722388
## iter  50 value 38.650867
## iter  60 value 36.987036
## iter  70 value 35.995633
## iter  80 value 34.883229
## iter  90 value 34.652110
## iter 100 value 34.463272
## final  value 34.463272
## stopped after 100 iterations
## # weights:  156
## initial  value 389.845878
## iter  10 value 165.596682
## iter  20 value 102.729465
## iter  30 value 44.388611
## iter  40 value 18.623006
## iter  50 value 6.765994
## iter  60 value 5.747427
## iter  70 value 5.686552
## iter  80 value 5.661885
## iter  90 value 5.637227
## iter 100 value 5.621104
## final  value 5.621104
## stopped after 100 iterations
## # weights:  32
## initial  value 368.686832
## iter  10 value 268.165208
## iter  20 value 263.015979
## iter  30 value 258.906656
## iter  40 value 256.454368
```

```
## iter  50 value 223.597253
## iter  60 value 170.657055
## iter  70 value 137.072034
## iter  80 value 137.025056
## iter  90 value 120.861351
## iter 100 value 113.075913
## final  value 113.075913
## stopped after 100 iterations
## # weights:  94
## initial  value 381.458931
## iter  10 value 107.794682
## iter  20 value 70.758528
## iter  30 value 47.186222
## iter  40 value 40.668749
## iter  50 value 38.204966
## iter  60 value 36.006329
## iter  70 value 29.783342
## iter  80 value 25.649121
## iter  90 value 25.453482
## iter 100 value 25.452687
## final  value 25.452687
## stopped after 100 iterations
## # weights:  156
## initial  value 366.601524
## iter  10 value 113.557084
## iter  20 value 84.750743
## iter  30 value 76.058810
## iter  40 value 72.953005
## iter  50 value 62.727939
## iter  60 value 47.093633
## iter  70 value 41.427252
## iter  80 value 37.886039
## iter  90 value 36.549877
## iter 100 value 36.117289
## final  value 36.117289
## stopped after 100 iterations
## # weights:  32
## initial  value 406.605448
## iter  10 value 141.492253
## iter  20 value 110.160400
## iter  30 value 100.417415
## iter  40 value 92.064246
## iter  50 value 81.681268
## iter  60 value 77.582809
## iter  70 value 76.967313
## iter  80 value 76.949078
## iter  80 value 76.949077
## iter  80 value 76.949077
## final  value 76.949077
## converged
## # weights:  94
## initial  value 376.068293
## iter  10 value 165.446195
## iter  20 value 127.670877
```

```
## iter  30 value 115.737376
## iter  40 value 113.873040
## iter  50 value 94.070956
## iter  60 value 86.738924
## iter  70 value 82.454396
## iter  80 value 71.431760
## iter  90 value 66.999065
## iter 100 value 66.675487
## final  value 66.675487
## stopped after 100 iterations
## # weights:  156
## initial  value 402.637910
## iter  10 value 159.923723
## iter  20 value 90.060312
## iter  30 value 63.553393
## iter  40 value 52.824443
## iter  50 value 49.219923
## iter  60 value 47.694047
## iter  70 value 46.483612
## iter  80 value 45.078554
## iter  90 value 44.263089
## iter 100 value 43.504871
## final  value 43.504871
## stopped after 100 iterations
## # weights:  32
## initial  value 392.956133
## iter  10 value 367.505941
## iter  20 value 363.857919
## iter  30 value 353.109100
## iter  40 value 349.598499
## iter  50 value 302.678665
## iter  60 value 181.857900
## iter  70 value 158.775582
## iter  80 value 155.359349
## iter  90 value 150.754659
## iter 100 value 147.433523
## final  value 147.433523
## stopped after 100 iterations
## # weights:  94
## initial  value 424.685027
## iter  10 value 175.722571
## iter  20 value 97.999043
## iter  30 value 75.206737
## iter  40 value 55.231776
## iter  50 value 35.849265
## iter  60 value 30.687824
## iter  70 value 29.560538
## iter  80 value 29.392844
## iter  90 value 29.034265
## iter 100 value 28.953696
## final  value 28.953696
## stopped after 100 iterations
## # weights:  156
## initial  value 616.934974
```

```
## iter  10 value 123.991722
## iter  20 value 52.282738
## iter  30 value 39.512640
## iter  40 value 36.639299
## iter  50 value 34.809313
## iter  60 value 33.236670
## iter  70 value 30.891213
## iter  80 value 29.013933
## iter  90 value 26.878512
## iter 100 value 25.283967
## final  value 25.283967
## stopped after 100 iterations
## # weights:  32
## initial  value 372.626830
## iter  10 value 290.192844
## iter  20 value 284.637613
## iter  30 value 284.620357
## iter  30 value 284.620355
## iter  30 value 284.620355
## final  value 284.620355
## converged
## # weights:  94
## initial  value 402.392217
## iter  10 value 109.848921
## iter  20 value 58.214746
## iter  30 value 45.126848
## iter  40 value 18.309261
## iter  50 value 14.171463
## iter  60 value 13.164761
## iter  70 value 12.809963
## iter  80 value 12.731341
## iter  90 value 12.610235
## iter 100 value 12.330900
## final  value 12.330900
## stopped after 100 iterations
## # weights:  156
## initial  value 418.133202
## iter  10 value 174.818981
## iter  20 value 144.726770
## iter  30 value 119.378789
## iter  40 value 82.446617
## iter  50 value 72.330524
## iter  60 value 68.617067
## iter  70 value 64.756569
## iter  80 value 63.356803
## iter  90 value 63.232818
## iter 100 value 63.175827
## final  value 63.175827
## stopped after 100 iterations
## # weights:  32
## initial  value 368.727715
## iter  10 value 301.741045
## iter  20 value 175.919604
## iter  30 value 132.133708
```

```
## iter  40 value 120.161834
## iter  50 value 96.890498
## iter  60 value 86.228806
## iter  70 value 76.832577
## iter  80 value 74.320273
## iter  90 value 72.077676
## iter 100 value 71.529700
## final  value 71.529700
## stopped after 100 iterations
## # weights:  94
## initial  value 394.831815
## iter  10 value 165.776263
## iter  20 value 79.349397
## iter  30 value 70.233263
## iter  40 value 53.476692
## iter  50 value 45.483144
## iter  60 value 44.530284
## iter  70 value 44.340652
## iter  80 value 44.186544
## iter  90 value 43.826159
## iter 100 value 43.528722
## final  value 43.528722
## stopped after 100 iterations
## # weights:  156
## initial  value 434.366184
## iter  10 value 213.176834
## iter  20 value 76.878537
## iter  30 value 54.304429
## iter  40 value 45.411723
## iter  50 value 41.854828
## iter  60 value 39.813091
## iter  70 value 37.615314
## iter  80 value 36.652510
## iter  90 value 36.303687
## iter 100 value 35.063781
## final  value 35.063781
## stopped after 100 iterations
## # weights:  32
## initial  value 500.417464
## iter  10 value 237.190372
## iter  20 value 130.319310
## iter  30 value 117.993177
## iter  40 value 111.832729
## iter  50 value 110.428810
## iter  60 value 107.922285
## iter  70 value 107.818523
## iter  80 value 107.790931
## iter  90 value 107.789127
## iter 100 value 107.787382
## final  value 107.787382
## stopped after 100 iterations
## # weights:  94
## initial  value 362.254260
## iter  10 value 129.422866
```

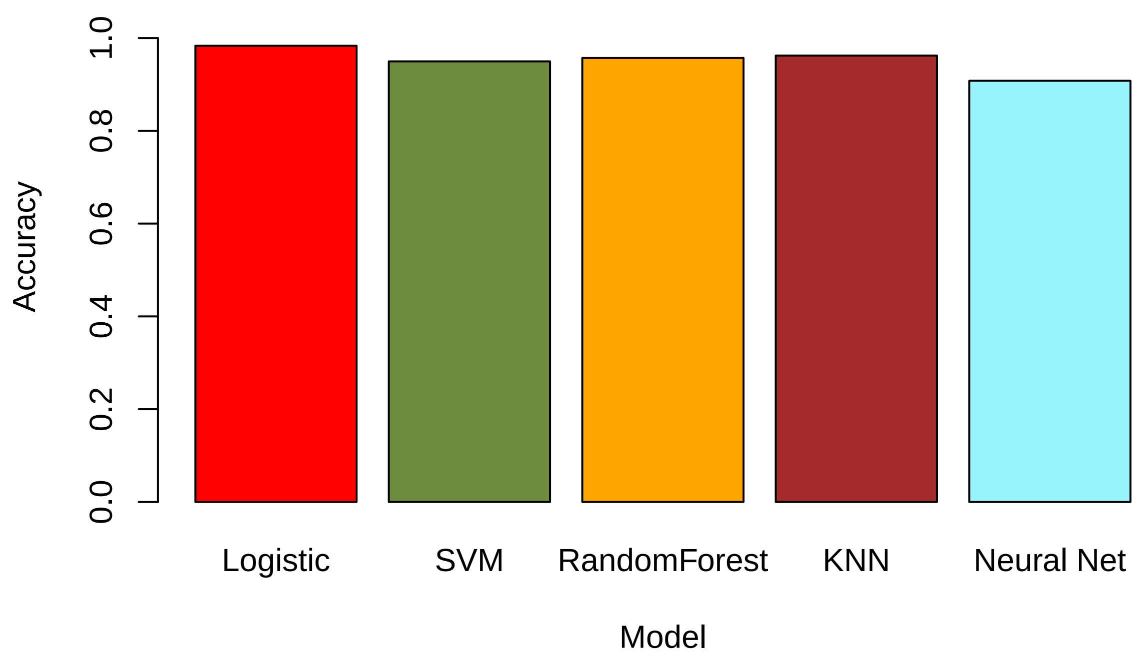
```
## iter  20 value 63.015096
## iter  30 value 46.588527
## iter  40 value 44.928191
## iter  50 value 43.607774
## iter  60 value 41.642488
## iter  70 value 40.919685
## iter  80 value 39.423362
## iter  90 value 38.968560
## iter 100 value 38.489605
## final  value 38.489605
## stopped after 100 iterations
## # weights:  156
## initial  value 342.215814
## iter  10 value 109.673903
## iter  20 value 67.135467
## iter  30 value 53.213453
## iter  40 value 46.785683
## iter  50 value 37.583483
## iter  60 value 31.656413
## iter  70 value 30.252040
## iter  80 value 25.924905
## iter  90 value 24.804350
## iter 100 value 24.094123
## final  value 24.094123
## stopped after 100 iterations
## # weights:  156
## initial  value 395.437761
## iter  10 value 132.633576
## iter  20 value 85.504456
## iter  30 value 62.391904
## iter  40 value 49.111987
## iter  50 value 46.718676
## iter  60 value 45.916960
## iter  70 value 43.872237
## iter  80 value 40.884105
## iter  90 value 40.020430
## iter 100 value 39.305818
## final  value 39.305818
## stopped after 100 iterations
```

```
pred_nn <- predict(model_nn, test_data)
cm_nn <- confusionMatrix(pred_nn,test_data$Class, positive = "0")
acc_nn <- cm_nn$overall['Accuracy']
cm_nn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 77435     6
##          1  7859   141
##
##                Accuracy : 0.9079
##                  95% CI : (0.906, 0.9099)
```

```
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0313
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.90786
##             Specificity : 0.95918
##          Pos Pred Value : 0.99992
##          Neg Pred Value : 0.01763
##              Prevalence : 0.99828
##          Detection Rate : 0.90630
##    Detection Prevalence : 0.90637
##       Balanced Accuracy : 0.93352
##
##        'Positive' Class : 0
##
```

```
accuracy_list = c(auc_logistic, acc_svm, acc_rf, acc_knn, acc_nn)
accuracy_list_names <- c("Logistic", "SVM","RandomForest","KNN","Neural Net")

# Plotting the accuracy of different models model_svm
colors <- c("red", "darkolivegreen4","orange","brown","cadetblue1")
barplot(accuracy_list, names.arg = accuracy_list_names, col = colors,ylim=c(0,1.1),xlab = "Model",ylab=
```



We can see that the best performing model is the Logistic Regression.