Final exam

Chaeeun Shin

Installing packages

```
if(!requireNamespace("tidyverse")) install.packages("tidyverse")
```

```
## Loading required namespace: tidyverse
```

```
if(!requireNamespace("caret")) install.packages("caret")
```

```
## Loading required namespace: caret
```

```
if(!requireNamespace("neuralnet")) install.packages("neuralnet")
```

```
## Loading required namespace: neuralnet
```

```
if(!requireNamespace("keras")) install.packages("keras")
```

```
## Loading required namespace: keras
```

```
if(!requireNamespace("randomForest")) install.packages("randomForest")
```

```
## Loading required namespace: randomForest
```

```
if(!requireNamespace("rpart")) install.packages("rpart")
if(!requireNamespace("rattle")) install.packages("rattle")
```

```
## Loading required namespace: rattle
```

```
if(!requireNamespace("kernlab")) install.packages("kernlab")
```

```
## Loading required namespace: kernlab
```

Loading packages

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
```

```
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'
##
## The following object is masked from 'package:dplyr':
##
##     compute
```

```
library(keras)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(rpart)
library(rattle)
```

```
## Loading required package: bitops
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
##
## Attaching package: 'rattle'
##
## The following object is masked from 'package:randomForest':
##
##     importance
```

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'
##
## The following object is masked from 'package:purrr':
##
##     cross
##
## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(tidyverse)
library(glmnet)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following object is masked from 'package:bitops':
##
##     %&%
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8
```

```
library(leaps)
library(ggplot2)
```

Part I

Loading data, splitting test and training data

```
data <- read.csv('rest.csv')
data <- na.omit(data)
cat("There are", dim(data)[1], "observations left.")
```

```
## There are 4601 observations left.
```

```
set.seed(123)
training.samples <- data$y %>%
  createDataPartition(p=0.75,list=FALSE)
train_data <- data[training.samples,]
test_data <- data[-training.samples,]

data$y <- as.factor(data$y)

nrow(train_data)
```

```
## [1] 3451
```

```
nrow(test_data)
```

```
## [1] 1150
```

Logistic model, prediction

```r
model <- glm(y~., train_data, family=binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

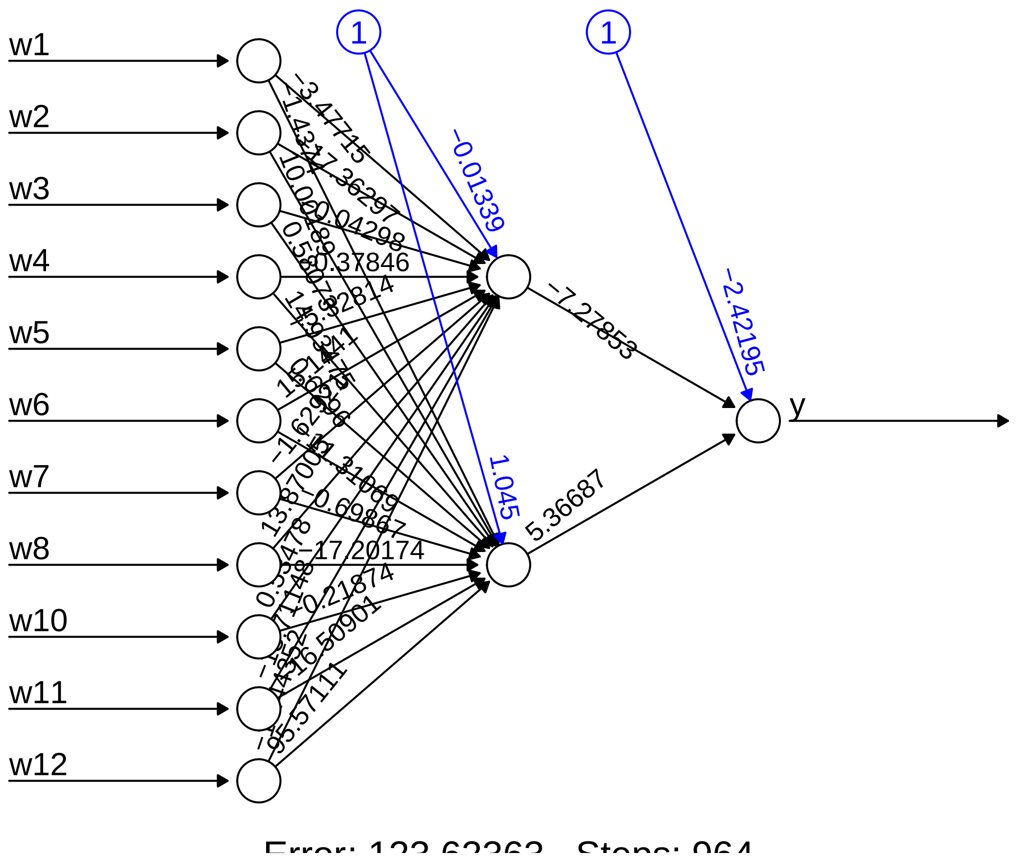
```r
probabilities <- model %>% predict(test_data,type="response")
predicted_class_lm <- ifelse(probabilities>0.5,1,0)

confusionMatrix(factor(predicted_class_lm),factor(test_data$y))
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##            0 658 108
##            1  36 348
##
##                   Accuracy : 0.8748
##                     95% CI : (0.8543, 0.8934)
##        No Information Rate : 0.6035
##        P-Value [Acc > NIR] : < 2.2e-16
##
##                      Kappa : 0.7311
##
##     Mcnemar's Test P-Value : 3.285e-09
##
##                Sensitivity : 0.9481
##                Specificity : 0.7632
##             Pos Pred Value : 0.8590
##             Neg Pred Value : 0.9062
##                 Prevalence : 0.6035
##             Detection Rate : 0.5722
##       Detection Prevalence : 0.6661
##          Balanced Accuracy : 0.8556
##
##           'Positive' Class : 0
##
```

Perception model, sse, 2 neurons

```r
set.seed(123)
model <- neuralnet(y~., data=train_data, hidden=2,err.fct="sse",linear.output=F)
plot(model, rep="best")
```

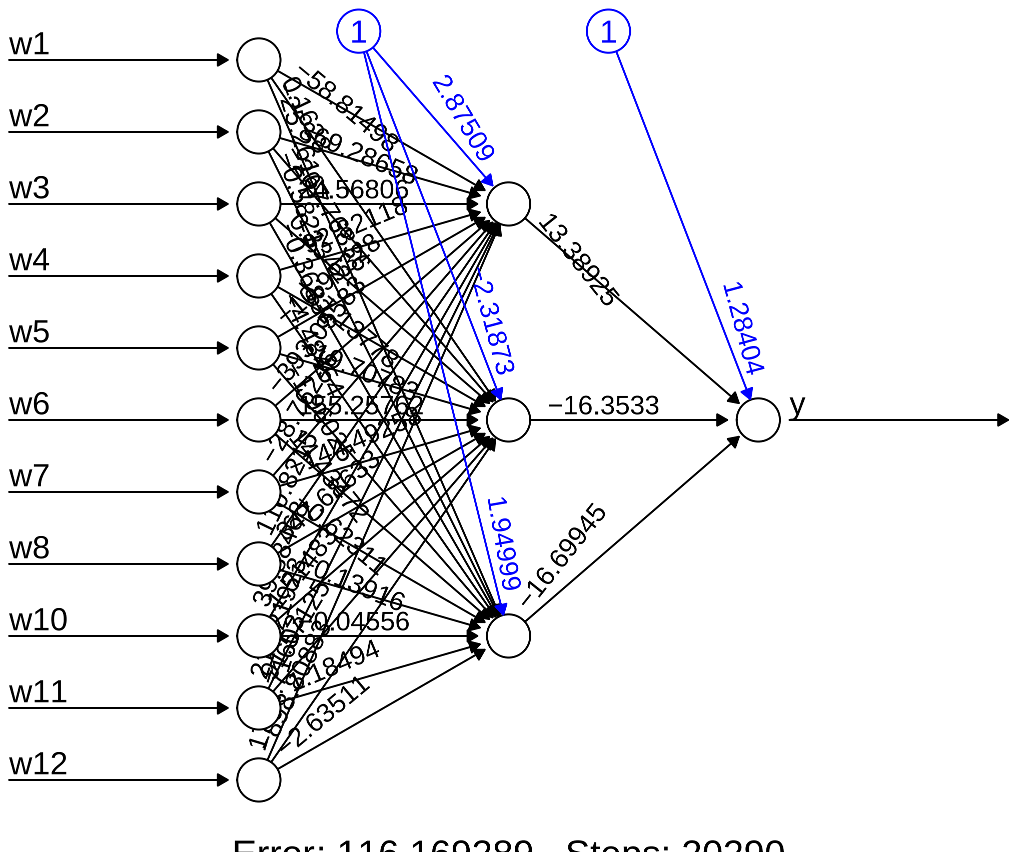


```
probabilities <- model %>% predict(test_data) %>% as.vector()
predicted_class_sse_2 <- ifelse(probabilities>0.5,1,0)
confusionMatrix(factor(predicted_class_sse_2),factor(test_data$y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 659  71
##          1  35 385
##
##                Accuracy : 0.9078
##                  95% CI : (0.8896, 0.9239)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8048
##
##  Mcnemar's Test P-Value : 0.0006751
##
##             Sensitivity : 0.8443
##             Specificity : 0.9496
##          Pos Pred Value : 0.9167
##          Neg Pred Value : 0.9027
##              Prevalence : 0.3965
##          Detection Rate : 0.3348
##    Detection Prevalence : 0.3652
##        Balanced Accuracy : 0.8969
##
##         'Positive' Class : 1
##
```

Perception model, sse, 3 neurons

```
set.seed(123)
model <- neuralnet(y~., data=train_data, hidden=3,err.fct="sse",linear.output=F)
plot(model, rep="best")
```



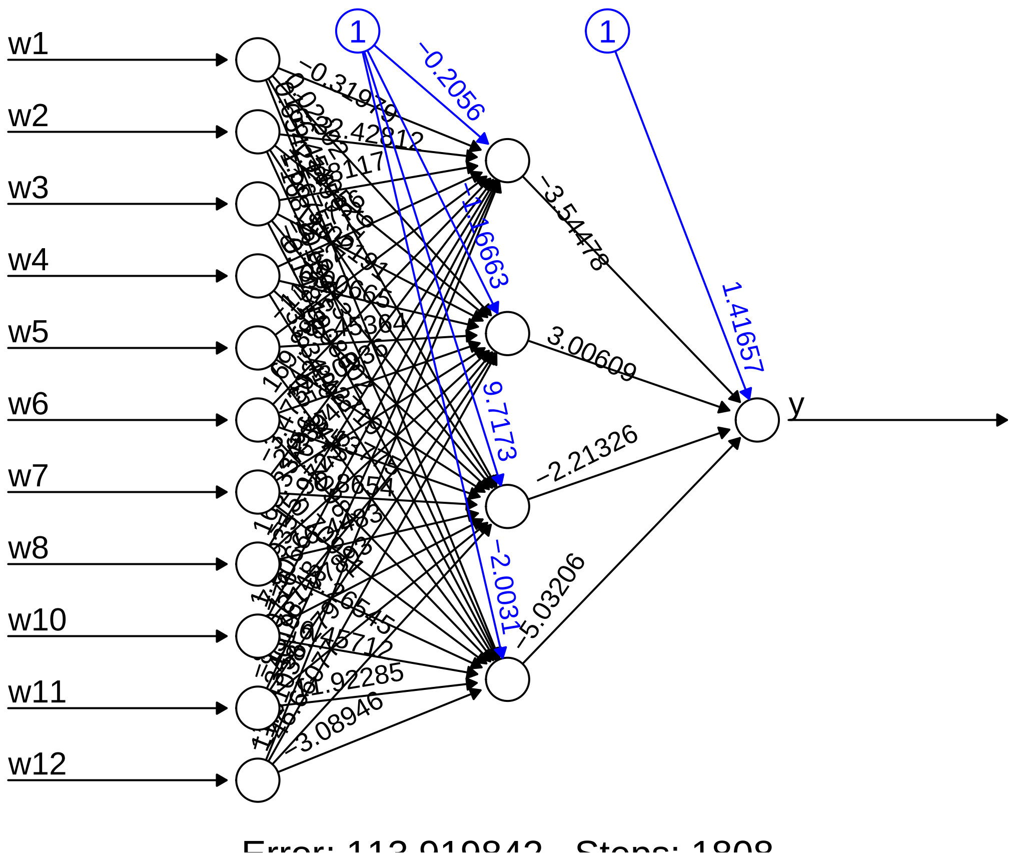
```
probabilities <- model %>% predict(test_data) %>% as.vector()
predicted_class_sse_3 <- ifelse(probabilities>0.5,1,0)
confusionMatrix(factor(predicted_class_sse_3),factor(test_data$y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 650  68
##          1  44 388
##
##                Accuracy : 0.9026
##                  95% CI : (0.884, 0.9191)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7946
```

```
##
##  Mcnemar's Test P-Value : 0.02976
##
##             Sensitivity : 0.8509
##             Specificity : 0.9366
##          Pos Pred Value : 0.8981
##          Neg Pred Value : 0.9053
##              Prevalence : 0.3965
##          Detection Rate : 0.3374
##    Detection Prevalence : 0.3757
##       Balanced Accuracy : 0.8937
##
##        'Positive' Class : 1
##
```

Perception model, sse, 4 neurons

```r
set.seed(123)
model <- neuralnet(y~., data=train_data, hidden=4,err.fct="sse",linear.output=F)
plot(model, rep="best")
```



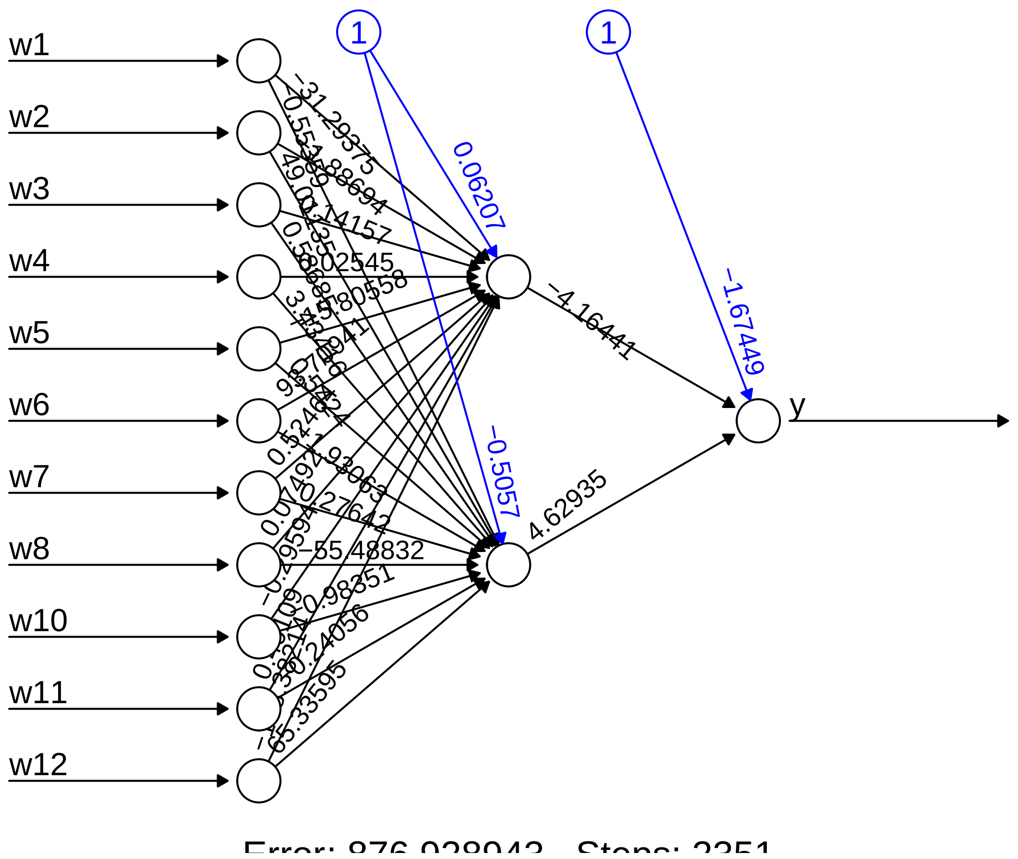
```r
probabilities <- model %>% predict(test_data) %>% as.vector()
predicted_class_sse_4 <- ifelse(probabilities>0.5,1,0)
confusionMatrix(factor(predicted_class_sse_4),factor(test_data$y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 655  72
##          1  39 384
##
##                Accuracy : 0.9035
##                  95% CI : (0.8849, 0.9199)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7958
##
##  Mcnemar's Test P-Value : 0.002387
##
##             Sensitivity : 0.8421
##             Specificity : 0.9438
##          Pos Pred Value : 0.9078
##          Neg Pred Value : 0.9010
##              Prevalence : 0.3965
##          Detection Rate : 0.3339
##    Detection Prevalence : 0.3678
##       Balanced Accuracy : 0.8930
##
##        'Positive' Class : 1
##
```

Perception model with one hidden layer with 2 neuron refers the best accuracy

Perception model, ce, 2 neurons

```
set.seed(123)
model <- neuralnet(y~., data=train_data, hidden=2,err.fct="ce",linear.output=F)
plot(model, rep="best")
```

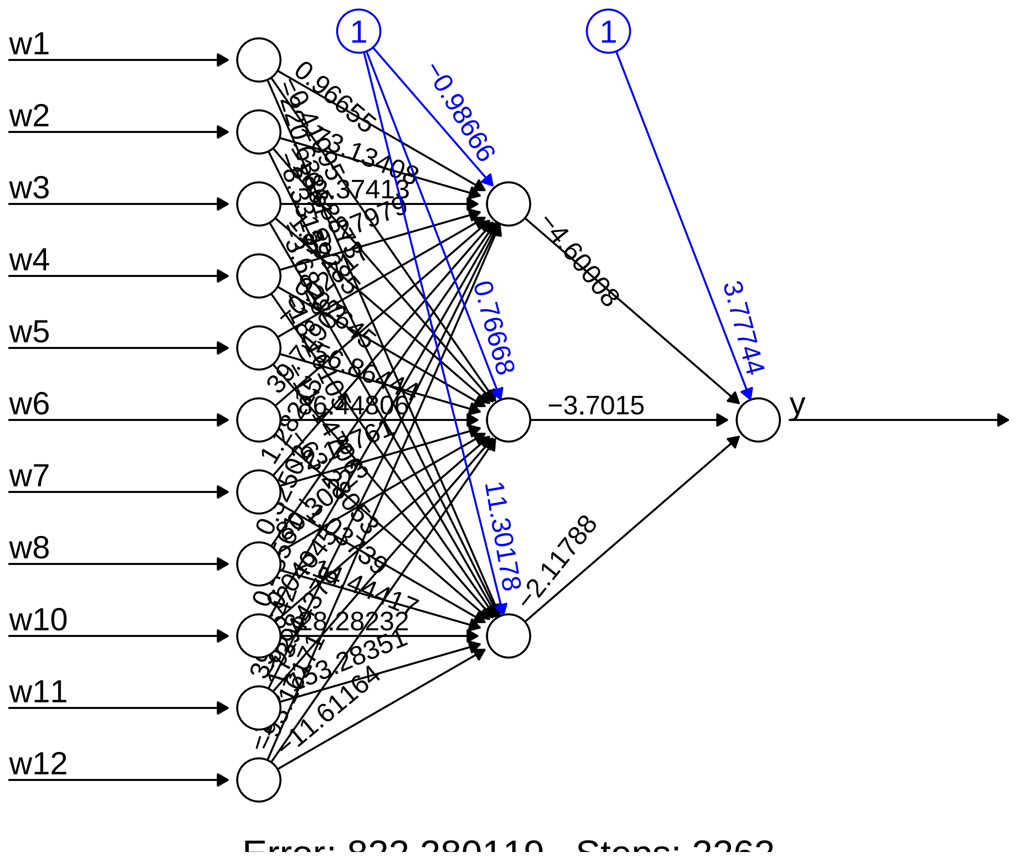


```r
probabilities <- model %>% predict(test_data) %>% as.vector()
predicted_class_ce_2 <- ifelse(probabilities>0.5,1,0)
confusionMatrix(factor(predicted_class_ce_2),factor(test_data$y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 658  73
##          1  36 383
##
##                Accuracy : 0.9052
##                  95% CI : (0.8868, 0.9215)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7992
##
##  Mcnemar's Test P-Value : 0.0005644
##
##             Sensitivity : 0.8399
##             Specificity : 0.9481
##          Pos Pred Value : 0.9141
##          Neg Pred Value : 0.9001
##              Prevalence : 0.3965
##          Detection Rate : 0.3330
##    Detection Prevalence : 0.3643
##        Balanced Accuracy : 0.8940
##
##         'Positive' Class : 1
##
```

Perception model, ce, 3 neurons

```r
set.seed(123)
model <- neuralnet(y~., data=train_data, hidden=3,err.fct="ce",linear.output=F)
plot(model, rep="best")
```



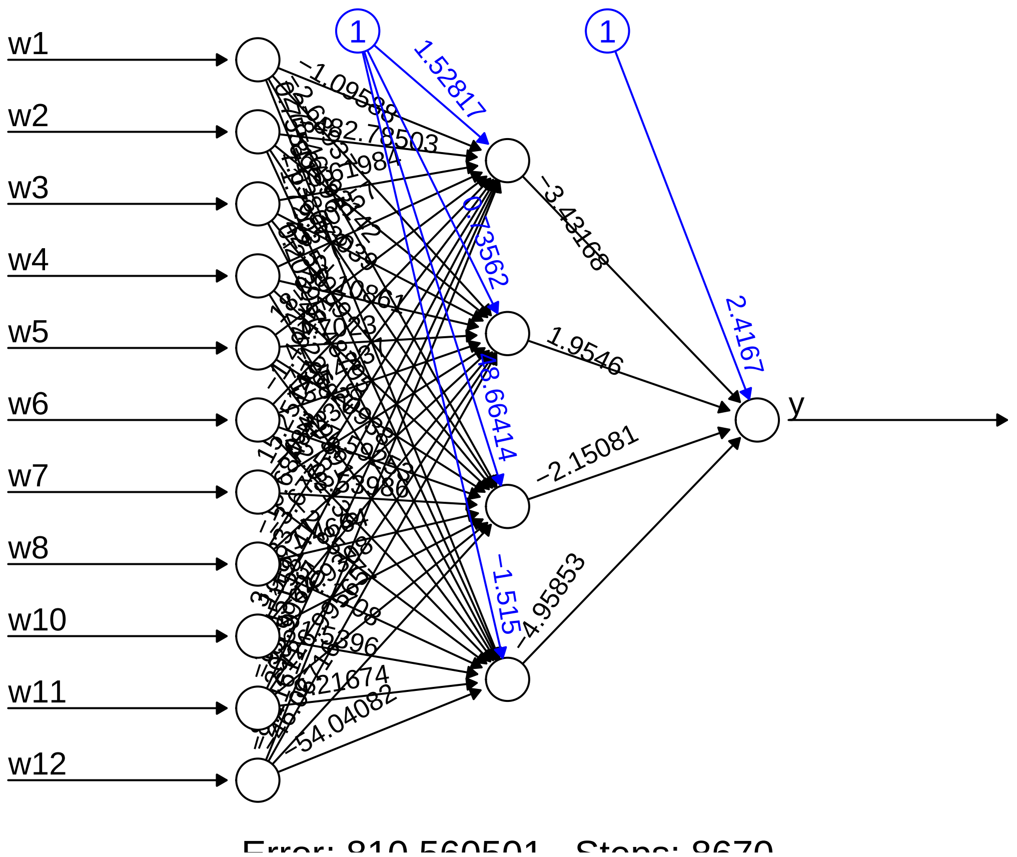
```r
probabilities <- model %>% predict(test_data) %>% as.vector()
predicted_class_ce_3 <- ifelse(probabilities>0.5,1,0)
confusionMatrix(factor(predicted_class_ce_3),factor(test_data$y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 649  68
##          1  45 388
##
##                Accuracy : 0.9017
##                  95% CI : (0.8831, 0.9183)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7929
```

```
##
##  Mcnemar's Test P-Value : 0.03849
##
##             Sensitivity : 0.8509
##             Specificity : 0.9352
##          Pos Pred Value : 0.8961
##          Neg Pred Value : 0.9052
##              Prevalence : 0.3965
##          Detection Rate : 0.3374
##    Detection Prevalence : 0.3765
##       Balanced Accuracy : 0.8930
##
##        'Positive' Class : 1
##
```

Perception model, ce, 4 neurons

```
set.seed(123)
model <- neuralnet(y~., data=train_data, hidden=4,err.fct="ce",linear.output=F)
plot(model, rep="best")
```



```
probabilities <- model %>% predict(test_data) %>% as.vector()
predicted_class_ce_4 <- ifelse(probabilities>0.5,1,0)
confusionMatrix(factor(predicted_class_ce_4),factor(test_data$y),positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 651  66
##          1  43 390
##
##                Accuracy : 0.9052
##                  95% CI : (0.8868, 0.9215)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8002
##
##  Mcnemar's Test P-Value : 0.0351
##
##             Sensitivity : 0.8553
##             Specificity : 0.9380
##          Pos Pred Value : 0.9007
##          Neg Pred Value : 0.9079
##              Prevalence : 0.3965
##          Detection Rate : 0.3391
##    Detection Prevalence : 0.3765
##       Balanced Accuracy : 0.8967
##
##        'Positive' Class : 1
##
```

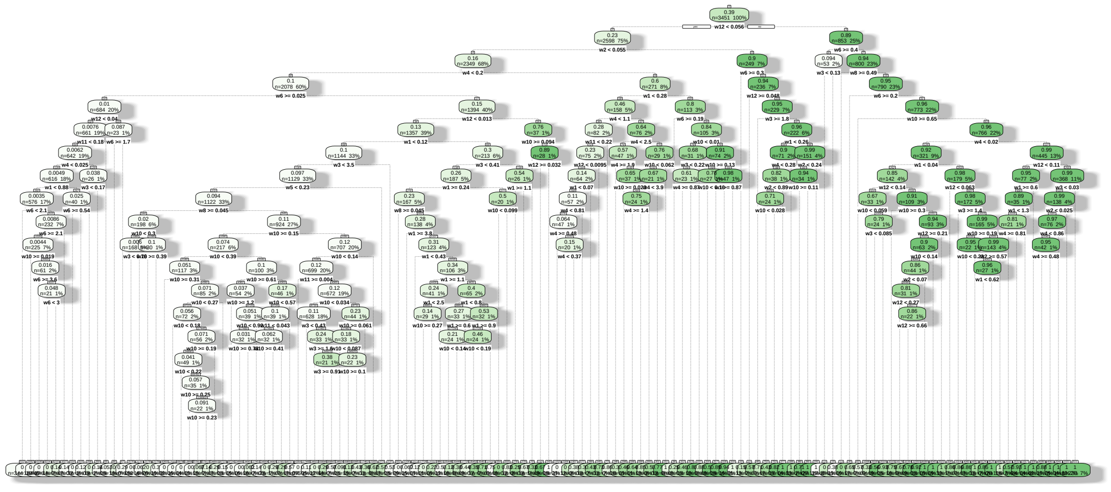
Perception model with 2 and 4 neurons have almost same accuracy. (.9052)

SSE model provides the best accuracy for the test data.

Fully grown tree

```
model <- rpart(y~., data=train_data, control=rpart.control(cp=0))
fancyRpartPlot(model)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2024−May−14 10:15:40 chaeeunshin

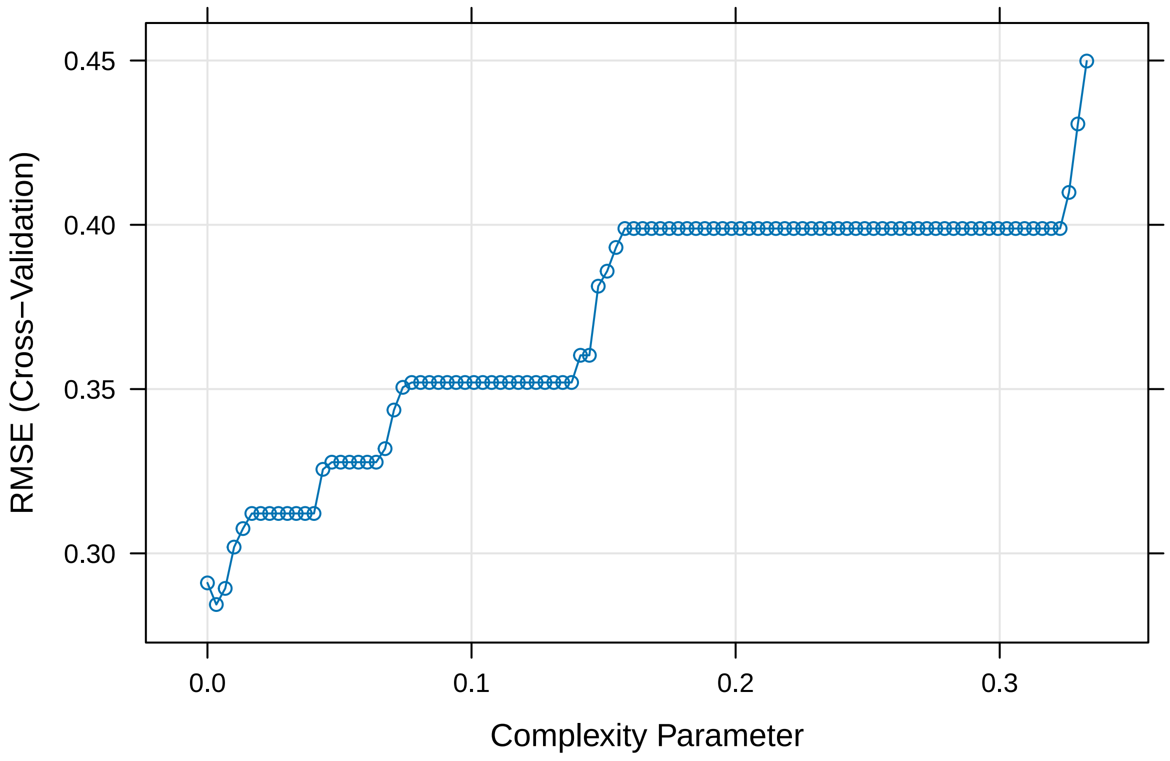
Pruned tree, prediction

```
set.seed(123)
model2 <- train(y~., data=train_data, method="rpart",trControl=trainControl("cv",number=10),tuneLength=
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

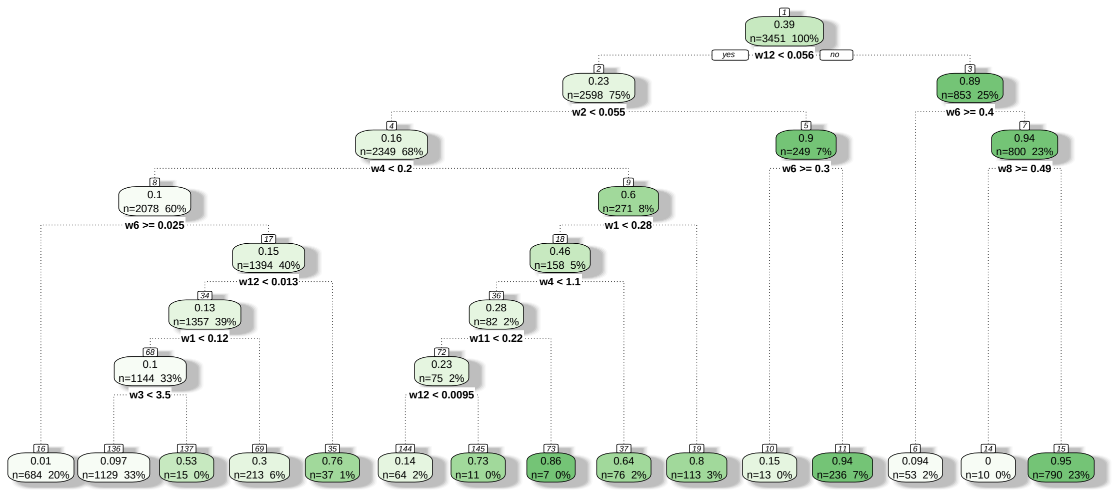
```
plot(model2)
```



```
model2$bestTune
```

```
##            cp
## 2 0.003363161
```

```
fancyRpartPlot(model2$finalModel)
```



Rattle 2024−May−14 10:15:43 chaeeunshin

```r
probabilities <- predict(model2, newdata=test_data)
predicted_class_prunedtree <- ifelse(probabilities>0.5,1,0)

confusionMatrix(factor(predicted_class_prunedtree),factor(test_data$y))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 645  84
##          1  49 372
##
##                Accuracy : 0.8843
##                  95% CI : (0.8644, 0.9023)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7551
##
##  Mcnemar's Test P-Value : 0.003197
##
##             Sensitivity : 0.9294
##             Specificity : 0.8158
##          Pos Pred Value : 0.8848
##          Neg Pred Value : 0.8836
##              Prevalence : 0.6035
##          Detection Rate : 0.5609
##    Detection Prevalence : 0.6339
##       Balanced Accuracy : 0.8726
##
##        'Positive' Class : 0
##
```

randomForest OOB

```
train_data$y <- factor(train_data$y)
test_data$y <- factor(test_data$y)

set.seed(123)
model <- train(y~., data=train_data, method="rf",trControl=trainControl("cv",number=10),importance=TRUE

model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 6
##
##         OOB estimate of  error rate: 8.66%
## Confusion matrix:
##      0    1 class.error
## 0 1984  110  0.05253104
## 1  189 1168  0.13927782
```

Accuracy

```
(1984+1168)/(1984+110+189+1168)
```

```
## [1] 0.9133584
```

Sensitivity

```
1168/(1168+189)
```

```
## [1] 0.8607222
```

Specificity

```
1984/(1984+110)
```

```
## [1] 0.947469
```

randomForest prediction

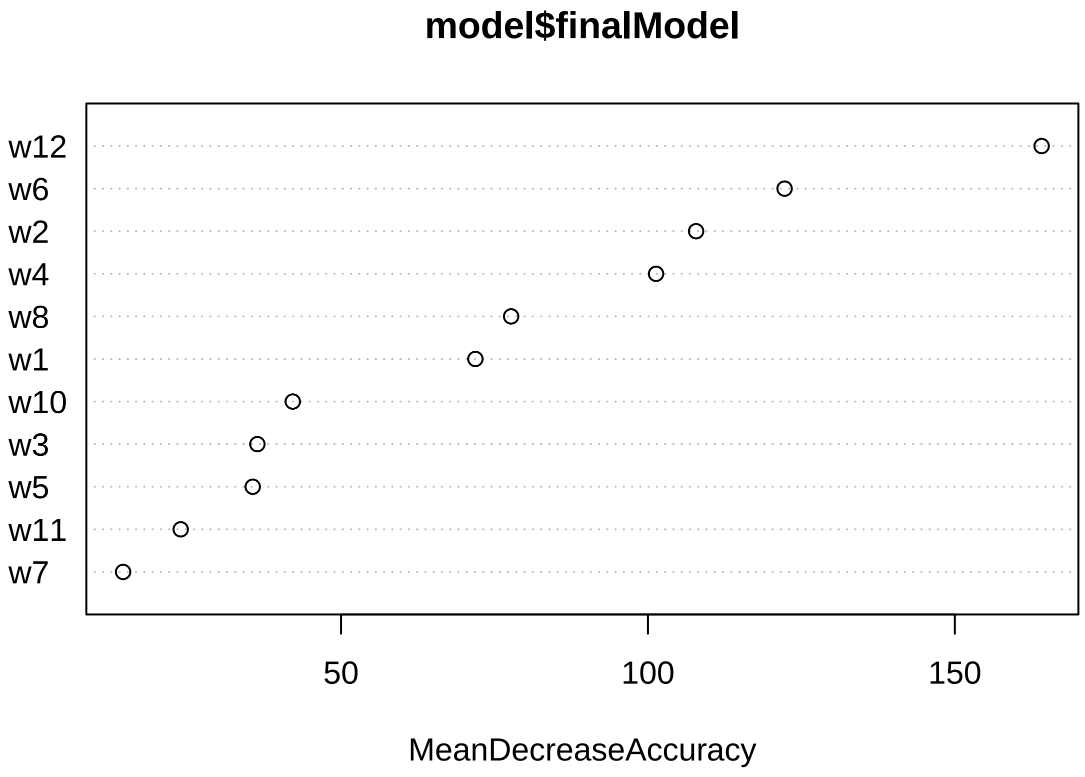
```
predicted_class_rf <- model %>% predict(test_data)
confusionMatrix(predicted_class_rf, test_data$y, positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 652  74
##          1  42 382
##
##                Accuracy : 0.8991
##                  95% CI : (0.8803, 0.9159)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7867
##
```

```
##  Mcnemar's Test P-Value : 0.003999
##
##             Sensitivity : 0.8377
##             Specificity : 0.9395
##          Pos Pred Value : 0.9009
##          Neg Pred Value : 0.8981
##              Prevalence : 0.3965
##          Detection Rate : 0.3322
##    Detection Prevalence : 0.3687
##       Balanced Accuracy : 0.8886
##
##        'Positive' Class : 1
##
```

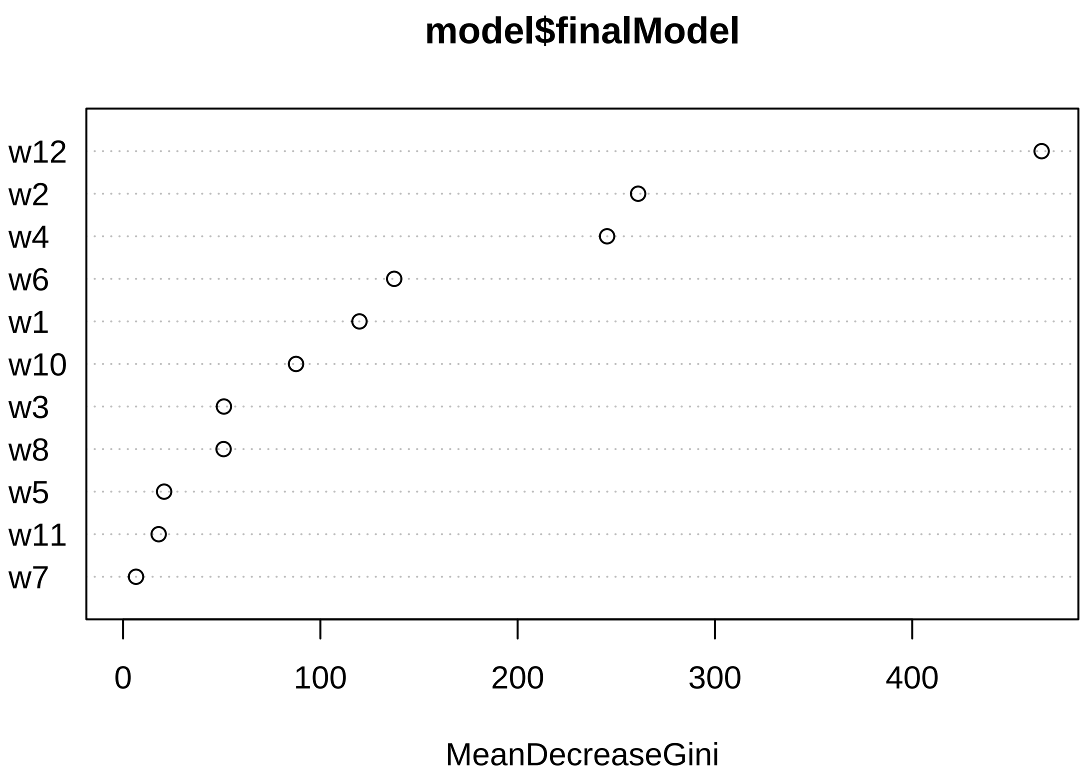
Plot MeanDecreaseAccuracy

```
varImpPlot(model$finalModel, type=1)
```



Plot MeanDecreaseGini

```
varImpPlot(model$finalModel, type=2)
```

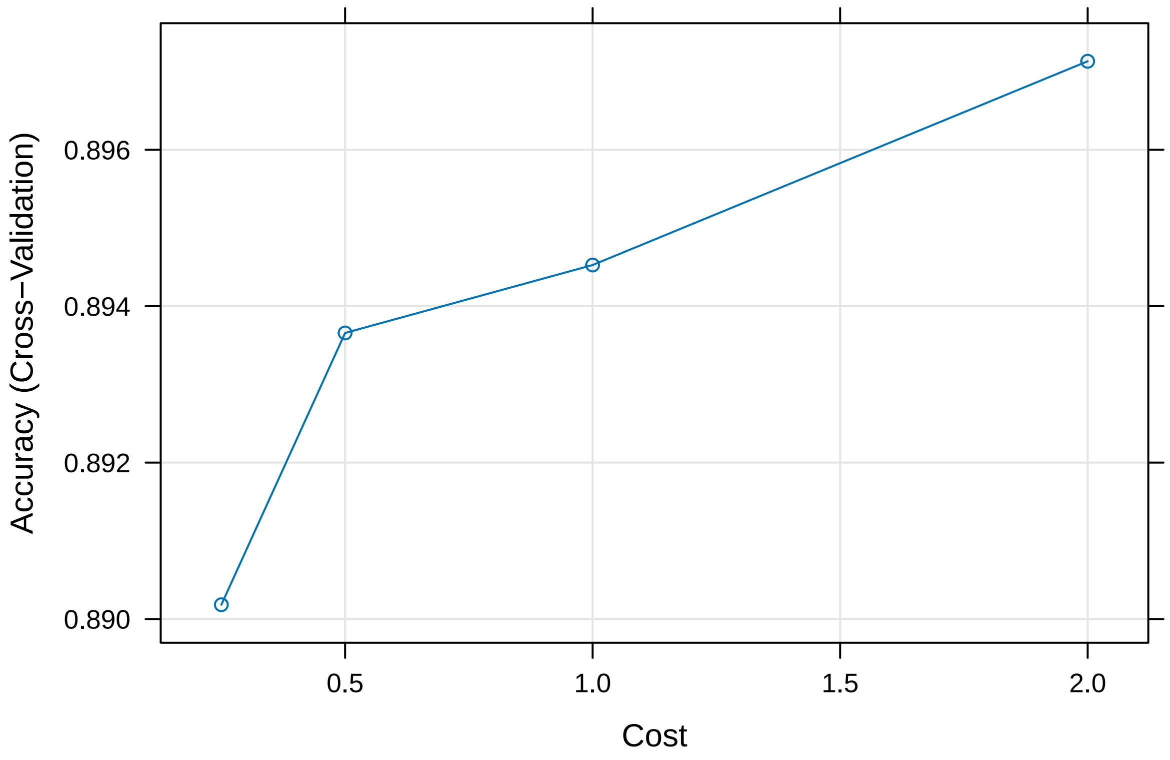


```
varImp(model,type=2)
```

```
## rf variable importance
##
##     Overall
## w12 100.000
## w2   55.443
## w4   52.016
## w6   28.509
## w1   24.673
## w10  17.670
## w3    9.704
## w8    9.668
## w5    3.101
## w11   2.502
## w7    0.000
```

Support vector machine

```
set.seed(123)
model <- train(y~., data=train_data, method="svmRadial",trControl=trainControl("cv",number=10),tuneLeng
plot(model)
```



```
model$bestTune
```

```
##       sigma C
## 4 0.4852082 2
```

Support vector machine prediction

```
predicted_class_svmPoly <- predict(model, newdata=test_data)
confusionMatrix(predicted_class_svmPoly, test_data$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 647  75
##          1  47 381
##
##                Accuracy : 0.8939
##                  95% CI : (0.8747, 0.9111)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.776
##
##  Mcnemar's Test P-Value : 0.01451
##
##             Sensitivity : 0.9323
##             Specificity : 0.8355
##          Pos Pred Value : 0.8961
##          Neg Pred Value : 0.8902
##              Prevalence : 0.6035
##          Detection Rate : 0.5626
##    Detection Prevalence : 0.6278
##       Balanced Accuracy : 0.8839
##
##        'Positive' Class : 0
##
```

Ensemble classifier prediction

```
pred <- cbind(predicted_class_sse_2, predicted_class_rf, predicted_class_prunedtree)
pred.m <- apply(pred,1,function(x) names(which.max(table(x))))

confusionMatrix(factor(pred.m), test_data$y, positive='1')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 650  80
##          1  44 376
##
##                Accuracy : 0.8922
##                  95% CI : (0.8728, 0.9095)
##     No Information Rate : 0.6035
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7716
##
##  Mcnemar's Test P-Value : 0.001672
##
##             Sensitivity : 0.8246
##             Specificity : 0.9366
##          Pos Pred Value : 0.8952
##          Neg Pred Value : 0.8904
##              Prevalence : 0.3965
##          Detection Rate : 0.3270
##    Detection Prevalence : 0.3652
##       Balanced Accuracy : 0.8806
##
##        'Positive' Class : 1
##
```

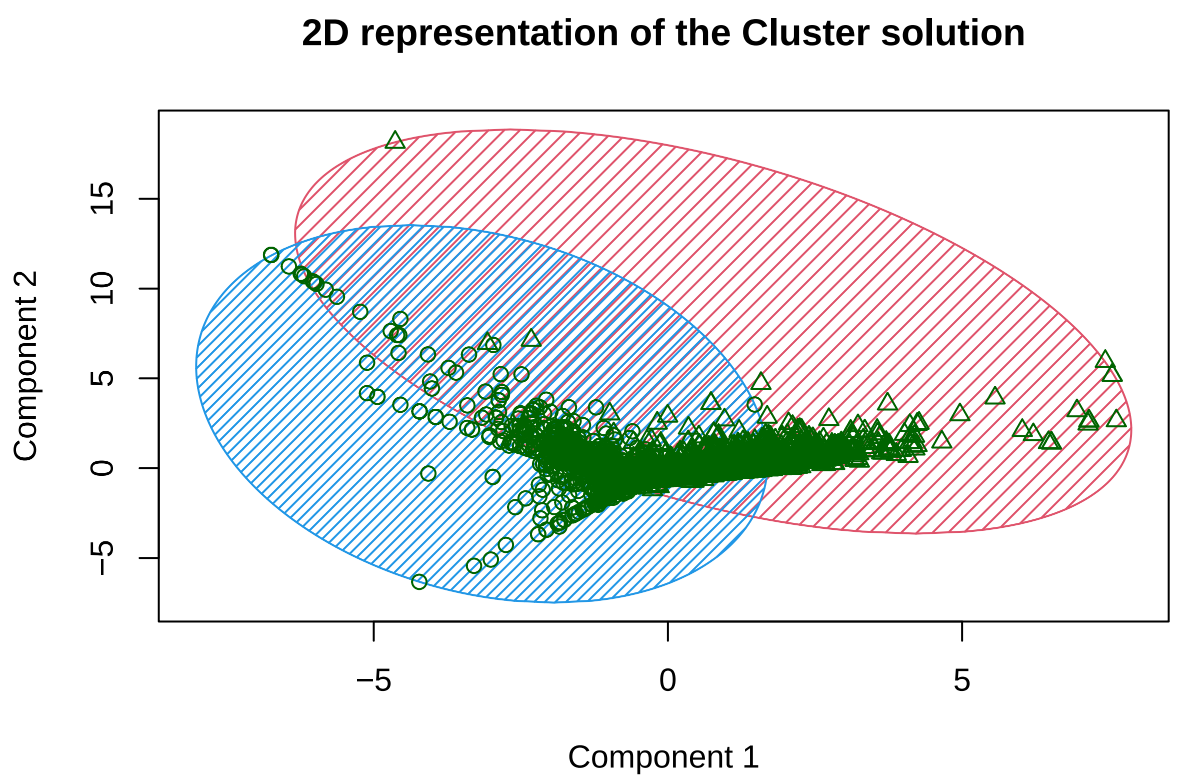
Ensemble classifier is better than the five individual classifier.

Part II

```
data <- read.csv("rest.csv")
data <- na.omit(data)
```

K-mean

```
k.means.fit <- kmeans(data,2)
library(cluster)
clusplot(data,k.means.fit$cluster,main="2D representation of the Cluster solution",color=TRUE,shade=TRU
```



These two components explain 28.38 % of the point variability.

```
table(k.means.fit$cluster,data$y)
```

```
## 
##         0    1
##    1 2422    7
##    2  366 1806
```

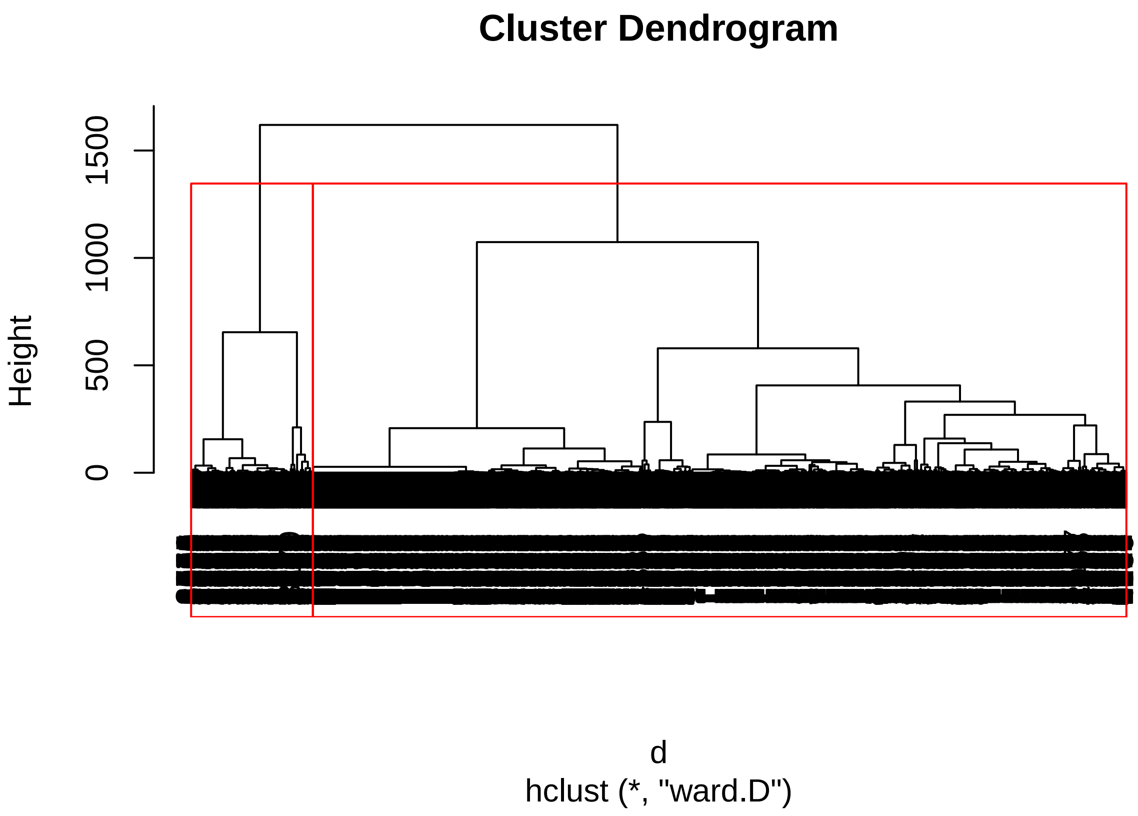
Accuracy

```
(370+7)/(366+2422+1806+7)
```

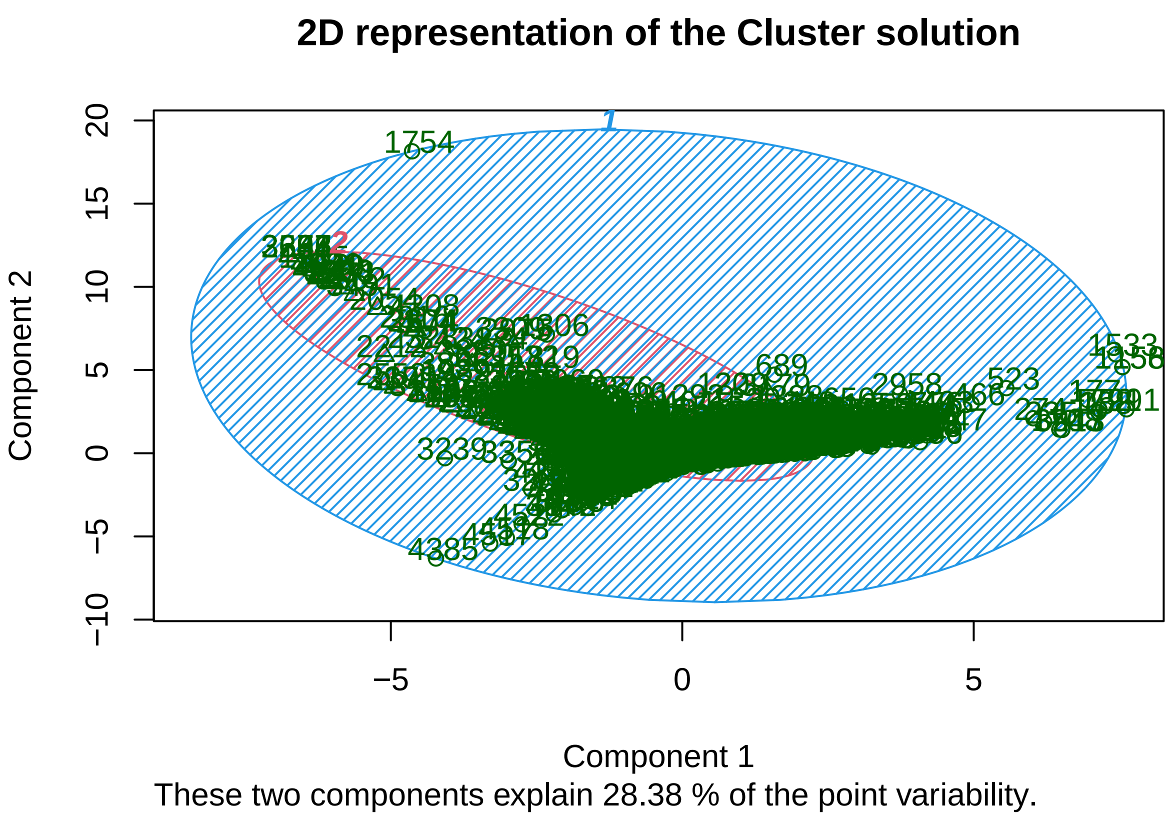
```
## [1] 0.08193871
```

Hierarchical: Ward

```
d <- dist(data,method="euclidean")
H.fit <- hclust(d,method="ward.D")
plot(H.fit)
rect.hclust(H.fit,k=2,border="red")
```



```
groups <- cutree(H.fit,k=2)
clusters.ward <- factor(groups,levels=1:2,labels=c(0,1))
clusplot(data,groups,main="2D representation of the Cluster solution",color=TRUE,shade=TRUE,labels=2,li
```



```
table(data$y,clusters.ward)
```

```
##    clusters.ward
##        0    1
##   0 2191  597
##   1 1811    2
```

Accuracy

```
(2191+2)/(2191+597+1811+2)
```

```
## [1] 0.4766355
```

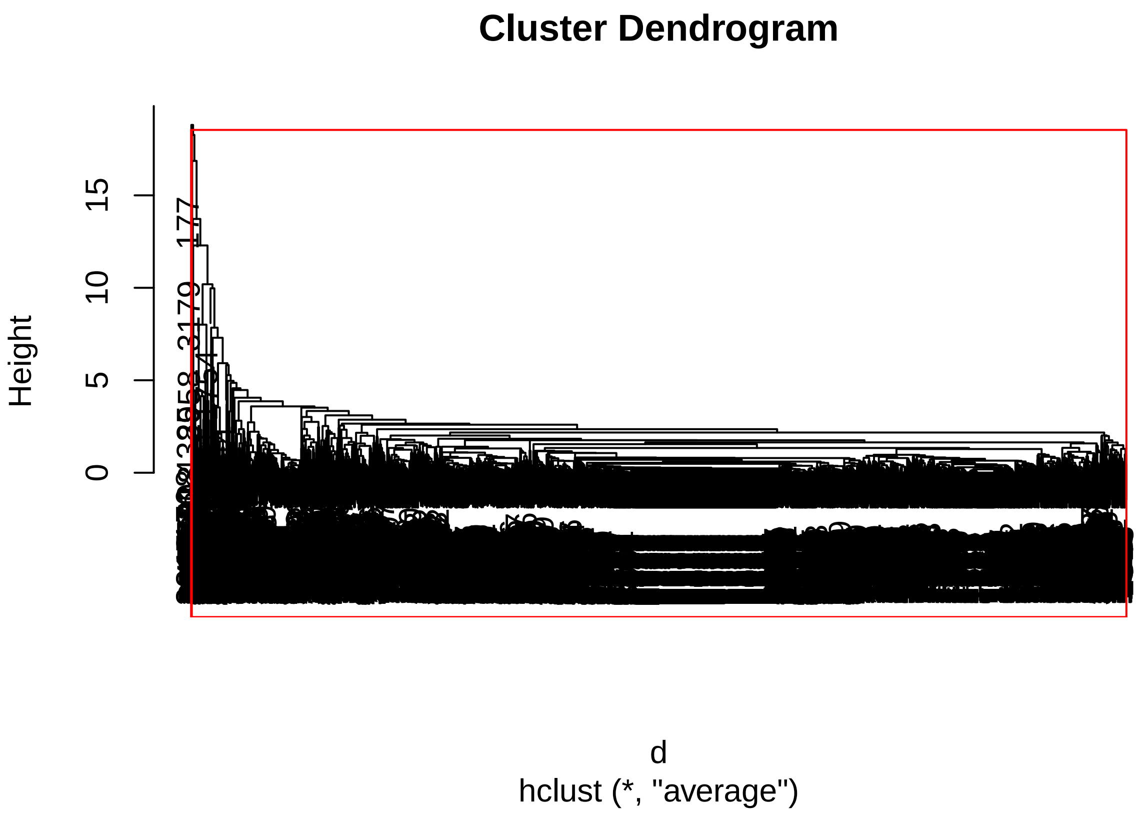
Confusion matrix to compare the clustering results of the K-means and the Ward's method

```
table(k.means.fit$cluster,clusters.ward)
```

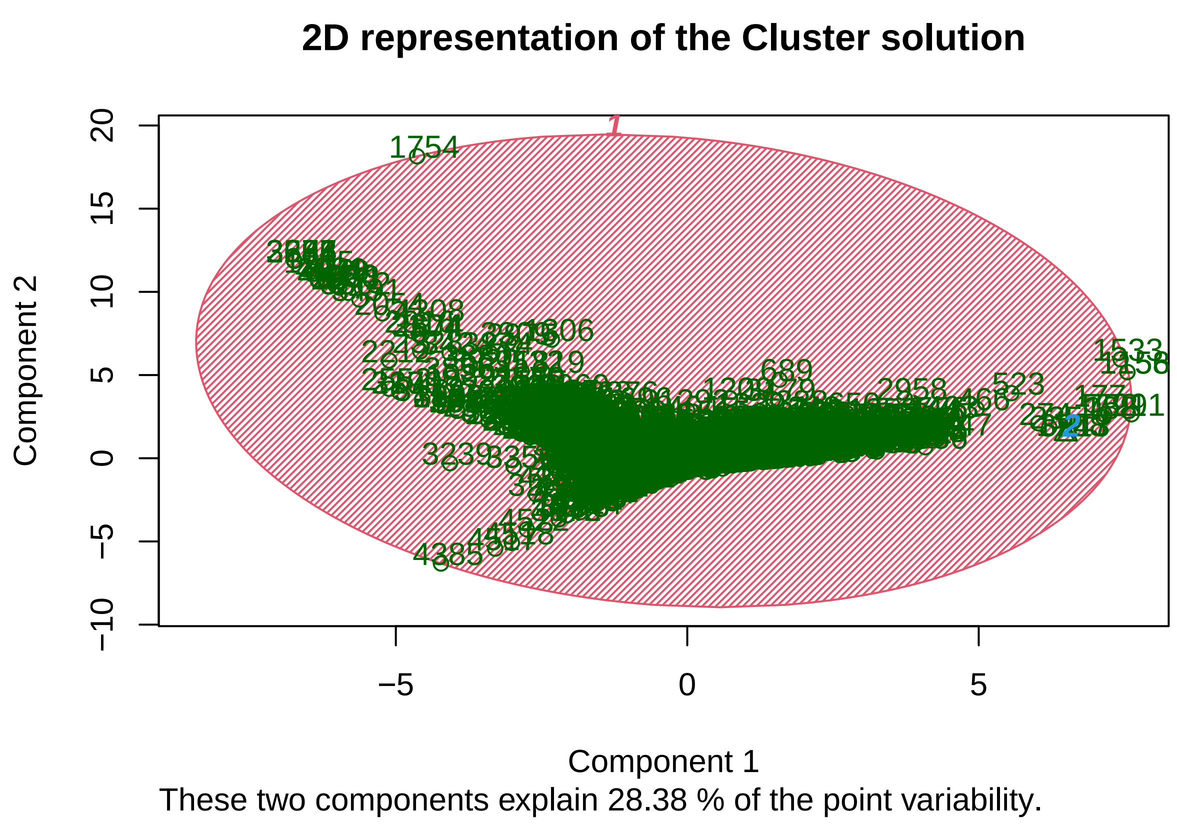
```
##    clusters.ward
##        0    1
##   1 1830  599
##   2 2172    0
```

Hierarchical: Average

```
H.fit <- hclust(d,method="average")
plot(H.fit)
rect.hclust(H.fit,k=2,border="red")
```



```
groups <- cutree(H.fit,k=2)
clusters.average <- factor(groups,levels=1:2,labels=c(0,1))
clusplot(data,groups,main="2D representation of the Cluster solution",color=TRUE,shade=TRUE,labels=2,li
```



```r
table(data$y,clusters.average)
```

```
##    clusters.average
##        0    1
##   0 2786    2
##   1 1812    1
```

Accuracy

```r
(2786+1)/(1812+2+1+2786)
```

```
## [1] 0.6057379
```

H.Average>H.Ward>K.means based on their accuracy on confusion matrix

```r
library(devtools)
```

```
## Loading required package: usethis
```

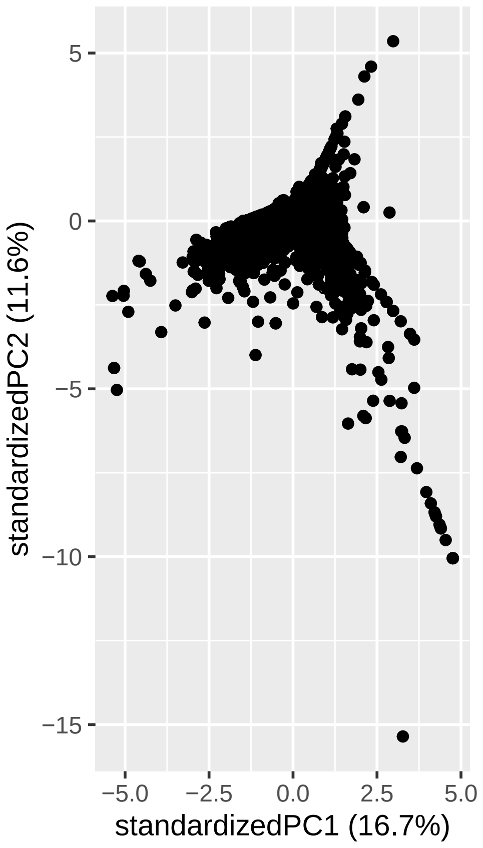
```r
library(ggbiplot)
```

```
##
## Attaching package: 'ggbiplot'
## The following object is masked from 'package:rattle':
##
##     wine
```

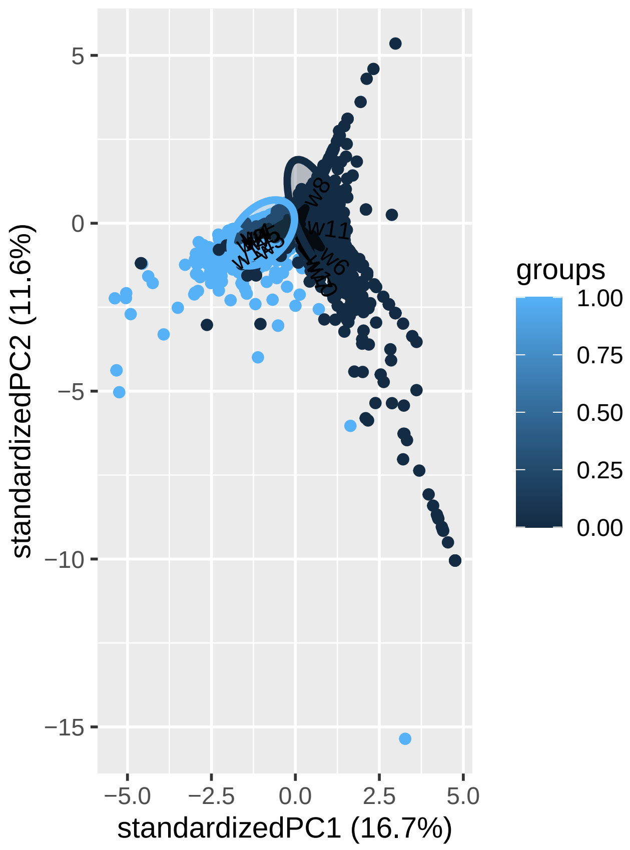
```r
data.pca <- prcomp(data,center=TRUE,scale. = TRUE)
summary(data.pca)
```

```
## Importance of components:
##                             PC1     PC2     PC3      PC4      PC5      PC6      PC7
## Standard deviation     1.4173 1.1818 1.0253 1.00469 0.99147 0.97932 0.97048
## Proportion of Variance 0.1674 0.1164 0.0876 0.08412 0.08192 0.07992 0.07849
## Cumulative Proportion  0.1674 0.2838 0.3714 0.45550 0.53742 0.61734 0.69583
##                            PC8     PC9    PC10    PC11   PC12
## Standard deviation     0.95448 0.91729 0.89380 0.79649 0.6814
## Proportion of Variance 0.07592 0.07012 0.06657 0.05287 0.0387
## Cumulative Proportion  0.77174 0.84186 0.90844 0.96130 1.0000
```

```
ggbiplot(data.pca)
```



```
ggbiplot(data.pca, ellipse = TRUE, groups = data$y)
```



```
data.pca$rotation[,1]
```

```
##         w1         w2         w3         w4         w5         w6         w7
## -0.2869951 -0.3584463 -0.1554061 -0.2959100 -0.1809565  0.3582133  0.2296587
##         w8        w10        w11        w12          y
##  0.1416651  0.1996604  0.0917179 -0.2843641 -0.5607341
```

The first PC contains 0.1674 of whole information of the original variable.

#Part III

```
data <- read.csv('restt.csv')
data <- na.omit(data)
ggplot(data,aes(x=Date,y=Close)) + geom_line()
```

```
## `geom_line()`: Each group consists of only one observation.
## i Do you need to adjust the group aesthetic?
```

```r
data$min_lagged <- lag(data$Low)
data$max_lagged <- log(data$High)
data$Close_norm <- (data$Close-data$min_lagged)/(data$max_lagged-data$min_lagged)

model_data <- matrix(data$Close_norm[-1])

knitr::kable(tail(model_data,10))
```

[491,]	-0.0067433
[492,]	0.0071986
[493,]	0.0180125
[494,]	-0.0343329
[495,]	-0.0672942
[496,]	-0.0497275
[497,]	-0.0436322
[498,]	-0.0254107
[499,]	-0.0335002
[500,]	-0.0047339

```r
train_data <- head(model_data,-10)
test_data <- tail(model_data,10)
cat(dim(train_data)[1], 'days are divided into the training set')
```

```
## 490 days are divided into the training set
```