분류 및 예측 (2)
: 신경망(Neural Network)

R을 활용한 신경망 실습

Neural Network Analysis – 반품고객 예측

상황

국내 홈쇼핑 A사는 최근 소비자의 반품 횟수가 증가됨에 따라 마케팅 부서의 김팀장이 반품고객의 특성을 파악하고자 함.

데이터

홈쇼핑 A사 고객 500명에 대한 성별, 나이, 구매금액, 홈쇼핑 출연자, 반품 여부

분석 과정

① 데이터 준비 → ② 변수 지정 → ③ 훈련 · 테스트자료 분류 → ④ 신경망 분석

Data: Hshopping.txt

No.	변수 이름		변수 설명	변수 유형
	SPSS용	SAS용		
1	ID	ID	고객 고유번호	수치형
2	성별	SEX	1=남자, 2=여자	범주형
3	나이	AGE	나이	수치형
4	구매금액	BUYM	1=10만 원 미만, 2=10~30만 원, 3=30만 원 이상	범주형
5	출연자	ACTOR	1=일반인, 2=유명인	범주형
6	반품 여부	RETURNSYN	0=반품 ×, 1=반품 ○	범주형

Neural Network Analysis using nnet

- Using "nnet", "devtools", "NeuralNetTools" packages
- Related functions
  - nnet() - nnet package
  - predict() - nnet package
  - plot.nnet() - devtools package
  - garson() - NeuralNetTools package

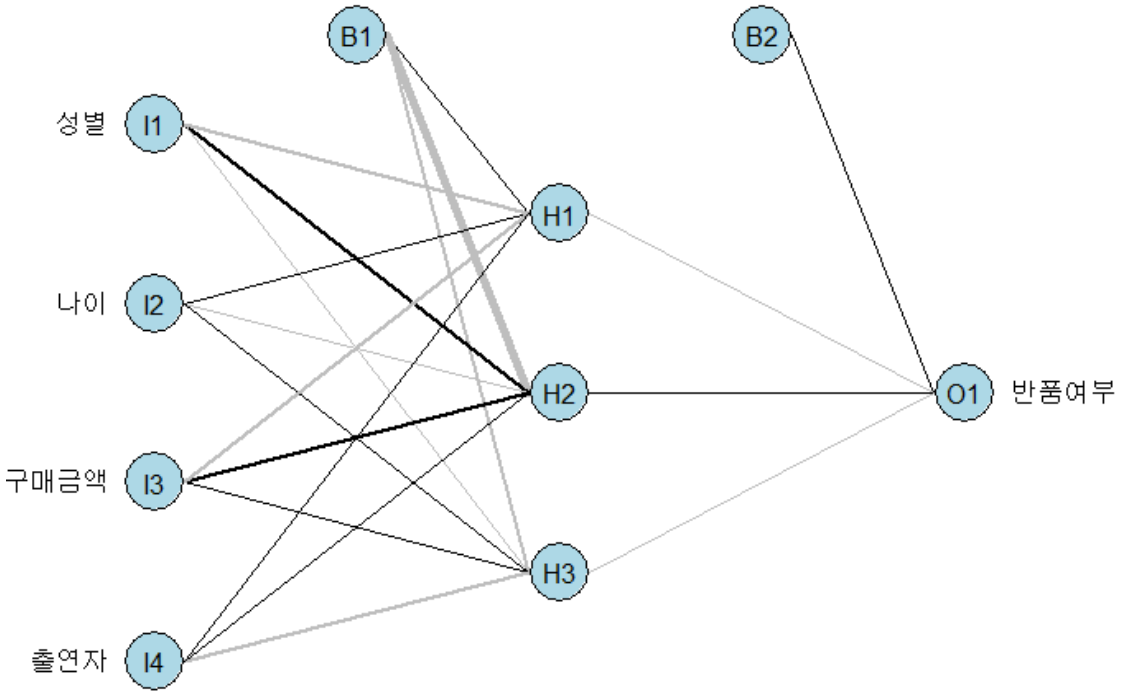
Neural Network Analysis using nnet

```
install.packages("nnet")
library(nnet); library(caret); library(ROCR)

cb <- read.delim("D:/Hshopping.txt", stringsAsFactors=FALSE)
cb$반품여부 <- factor(cb$반품여부)     # 명목형 값 예측일 경우

set.seed(1)
inTrain <- createDataPartition(y=cb$반품여부, p=0.6, list=FALSE)
cb.train <- cb[inTrain,]
cb.test <- cb[-inTrain,]

set.seed(123)
nn_model <- nnet(반품여부 ~ 성별+나이+구매금액+출연자, data=cb.train,
size=3, maxit=1000)        # size: # of hidden nodes

summary(nn_model)
```

```
a 4-3-1 network with 19 weights
options were - entropy fitting
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1
  26.33 -101.85    3.32  -69.49   47.08
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2
-243.83  100.35   -7.51   97.79   20.53
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3
 -86.52  -17.34    5.66   18.53  -74.97
   b->o   h1->o   h2->o   h3->o
   0.57  -35.69   53.31   -2.57
```

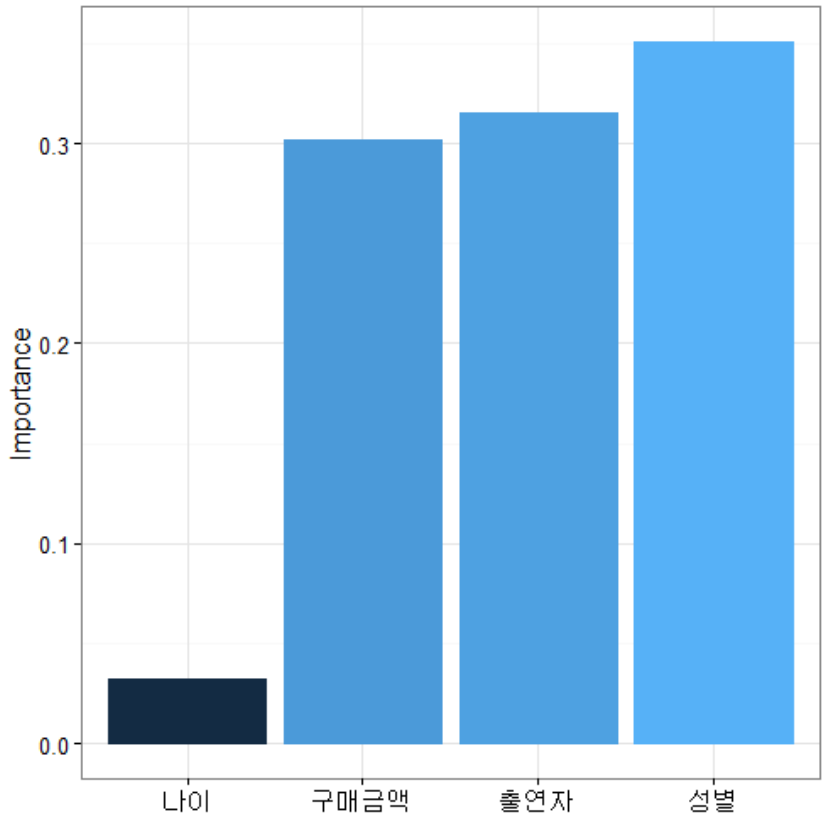
Neural Network Analysis using nnet

```
install.packages("devtools")
library(devtools)
source_url('https://gist.githubusercontent.com/Peque/41a9e20d6687f2f3108d/raw/85e14f3a292e126f1454864427e3a189c2fe33f3/nnet_plot_update.r')
plot.nnet(nn_model)
```



Neural Network Analysis using nnet

```
install.packages("NeuralNetTools")
library(NeuralNetTools)
garson(nn_model)          # 인공신경망 변수중요도 확인(garson 알고리즘)
```



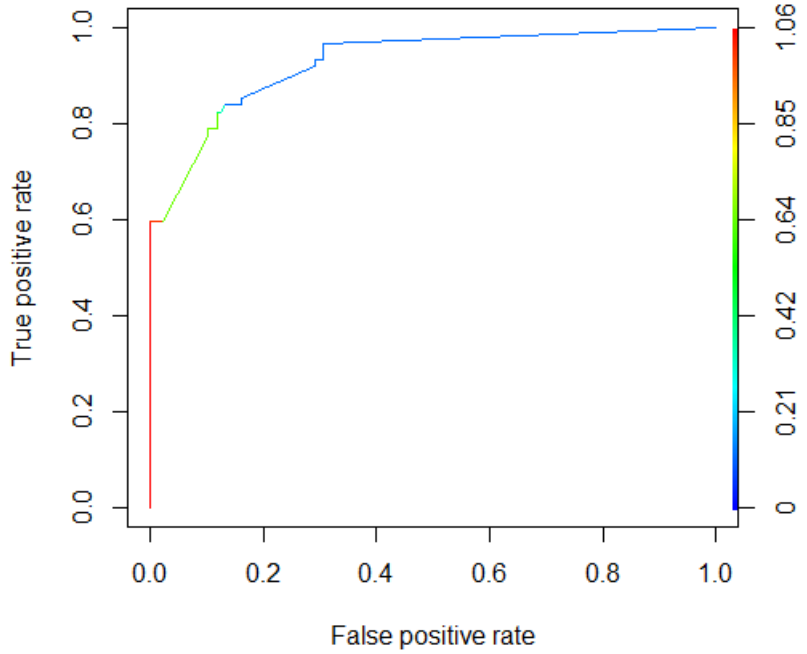
Neural Network Analysis using nnet

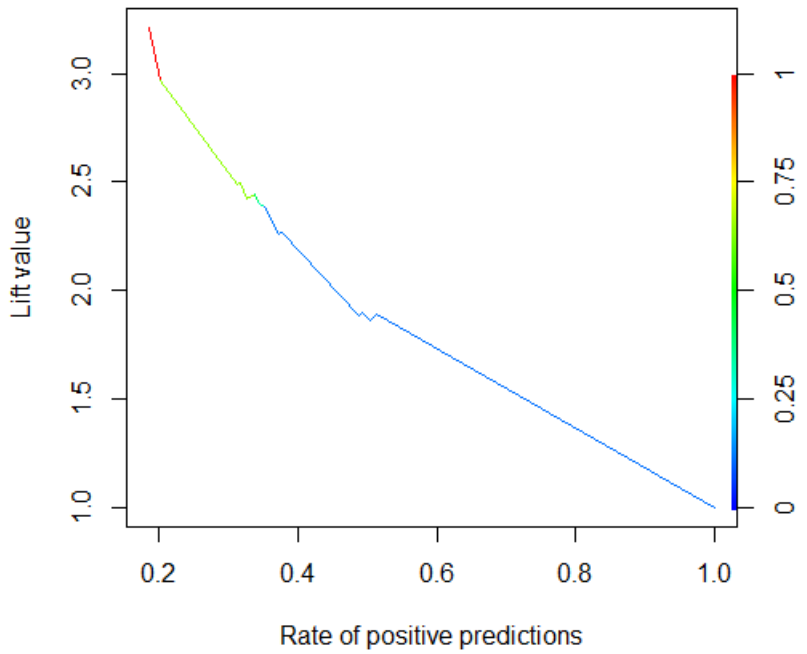
- `confusionMatrix(predict(nn_model, newdata=cb.test, type="class"), cb.test$반품여부)`

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 121  12
         1  16  50

               Accuracy : 0.8592965
                 95% CI : (0.8031029, 0.9044237)
    No Information Rate : 0.6884422
    P-Value [Acc > NIR] : 0.00000001998714

                  Kappa : 0.6776955
 Mcnemar's Test P-Value : 0.5707504

            Sensitivity : 0.8832117
            Specificity : 0.8064516
         Pos Pred Value : 0.9097744
         Neg Pred Value : 0.7575758
             Prevalence : 0.6884422
         Detection Rate : 0.6080402
   Detection Prevalence : 0.6683417
      Balanced Accuracy : 0.8448316

       'Positive' Class : 0
```

Neural Network Analysis using nnet

```
nn_pred <- ROCR::prediction(predict(nn_model, newdata=cb.test, type="raw"), cb.test$반품여부)
nn_model.perf1 <- performance(nn_pred, "tpr", "fpr") # ROC-chart
nn_model.perf2 <- performance(nn_pred, "lift", "rpp") # Lift chart
plot(nn_model.perf1, colorize=TRUE); plot(nn_model.perf2, colorize=TRUE)
```





nnet Features

❖ Weight decay

- used to avoid over-fitting through preventing the weights from growing too large
- realized by adding a term to the cost function that penalizes large weights,

$$E(w) = E_0(w) + \frac{1}{2}\lambda \sum_i w_i^2$$

- 사용법: nnet()함수에서 decay 파라미터에 작은 값(ex: 5e-4)을 지정

❖ Initial random weights

- 초기 가중치 설정
- 사용법: nnet() 함수에서 rang 파라미터에 임의의 값(ex: 0.1)를 지정하면 초기 가중치가 [-0.1, 0.1] 사이의 값으로 랜덤하게 설정됨

Neural Network Analysis using neuralnet

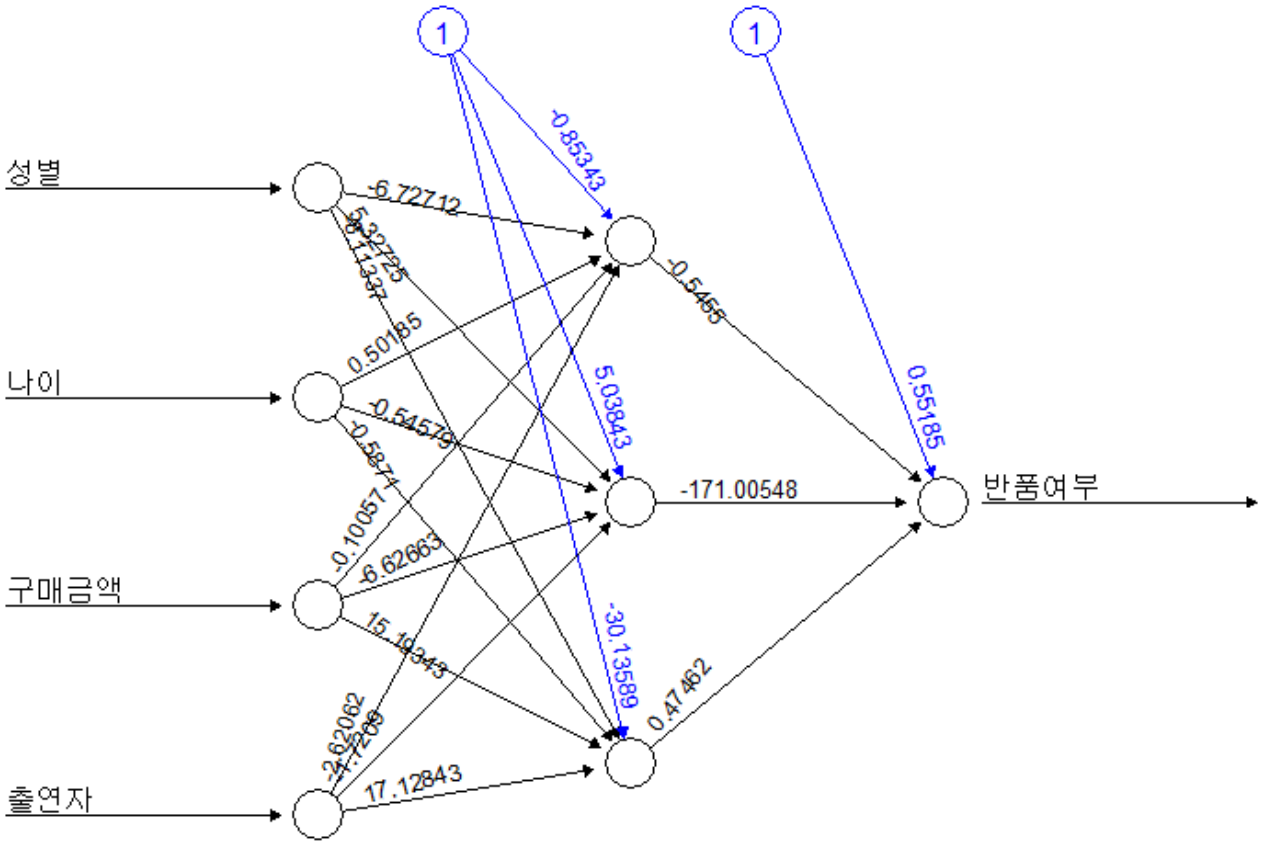
- Using "neuralnet" packages
- Related functions
  - neuralnet() - neuralnet package
  - compute() - neuralnet package
  - gwplot() - neuralnet package

Neural Network Analysis using neuralnet

```
install.packages("neuralnet")
library(neuralnet)

cb <- read.delim("Hshopping.txt", stringsAsFactors=FALSE)
# neuralnet 패키지는 목표변수가 numeric이어야 함.

set.seed(1)
inTrain <- createDataPartition(y=cb$반품여부, p=0.6, list=FALSE)
cb.train <- cb[inTrain,]
cb.test <- cb[-inTrain,]

set.seed(123)
nn2_model <- neuralnet(반품여부 ~ 성별+나이+구매금액+출연자, data=cb.train,
hidden=3, threshold=0.01)
# hidden: # of hidden nodes
# threshold: if change in error at a given step of iteration is
  less than the threshold, the model will stop further optimization.
# linear.output: If act.fct('logistic' or 'tanh') should not be applied
  to the output neurons set linear output to TRUE, otherwise to FALSE
  (default = TRUE)
# stepmax: the maximum steps for the training of the neural network..
```

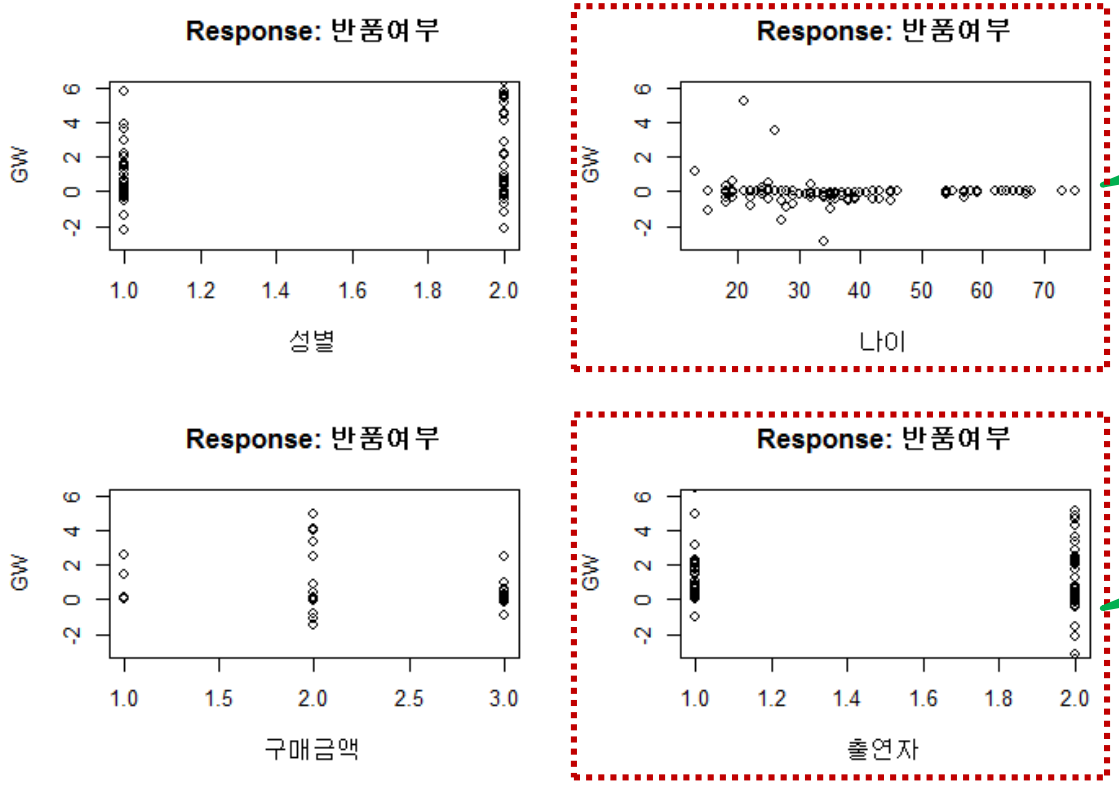
Neural Network Analysis using neuralnet

- `plot(nn2_model)`



Neural Network Analysis using neuralnet

```
par(mfrow=c(2,2))
gwplot(nn2_model, selected.covariate = "성별", min=-3, ,max=6)
gwplot(nn2_model, selected.covariate = "나이", min=-3, ,max=6)
gwplot(nn2_model, selected.covariate = "구매금액", min=-3, ,max=6)
gwplot(nn2_model, selected.covariate = "출연자", min=-3, ,max=6)
par(mfrow=c(1,1))
```



일반화 가중치의 분산이 0에 가까움 -> 결과에 미치는 영향이 없음.

일반화 가중치의 분산이 1보다 크기 때문에 비선형 효과가 있음.

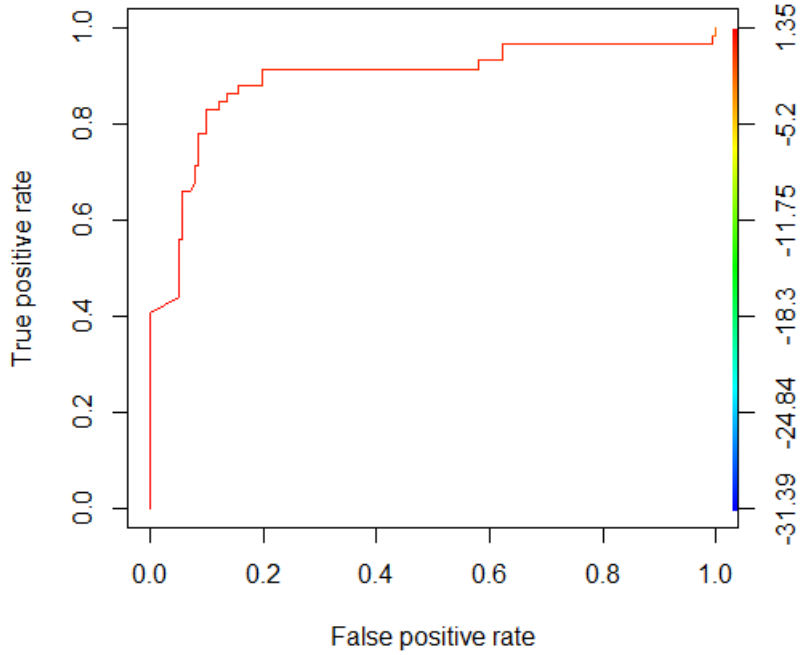
Neural Network Analysis using neuralnet

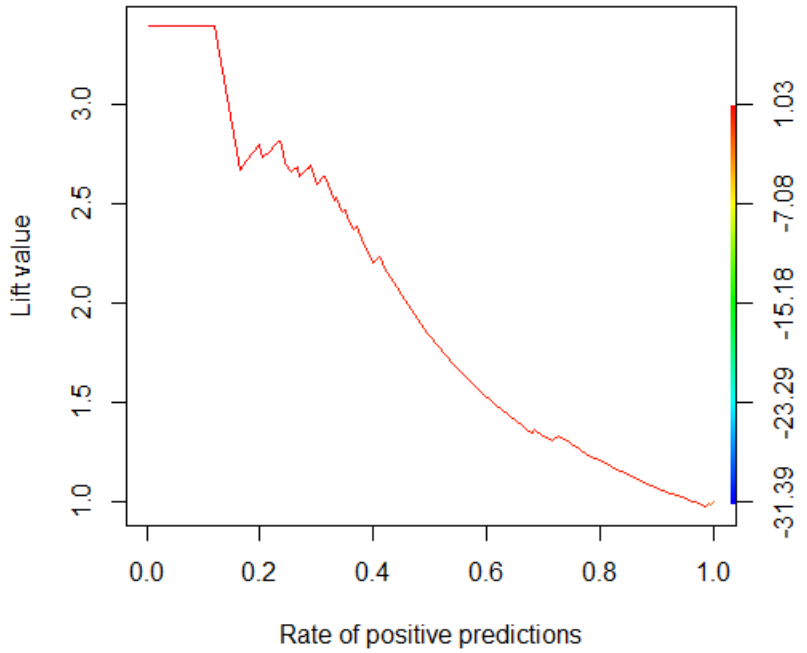
- `cb.test$nn2_pred_prob <- compute(nn2_model, covariate=cb.test[, c(2:5)])$net.result`
- `cb.test$nn2_pred <- ifelse(cb.test$nn2_pred_prob > 0.5, 1, 0)`
- `confusionMatrix(cb.test$nn2_pred, cb.test$반품여부)`

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 128  13
         1  13  46

               Accuracy : 0.87
                 95% CI : (0.8153477, 0.9132916)
    No Information Rate : 0.705
    P-Value [Acc > NIR] : 0.00000002942981

                  Kappa : 0.6874624
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9078014
            Specificity : 0.7796610
         Pos Pred Value : 0.9078014
         Neg Pred Value : 0.7796610
             Prevalence : 0.7050000
         Detection Rate : 0.6400000
   Detection Prevalence : 0.7050000
      Balanced Accuracy : 0.8437312

       'Positive' Class : 0
```

Neural Network Analysis using neuralnet

- `nn2_pred <- ROCR::prediction(cb.test$nn2_pred_prob, cb.test$반품여부)`
- `nn2_model.perf1 <- performance(nn2_pred, "tpr", "fpr") # ROC-chart`
- `nn2_model.perf2 <- performance(nn2_pred, "lift", "rpp") # Lift chart`
- `plot(nn2_model.perf1, colorize=TRUE); plot(nn2_model.perf2, colorize=TRUE)`



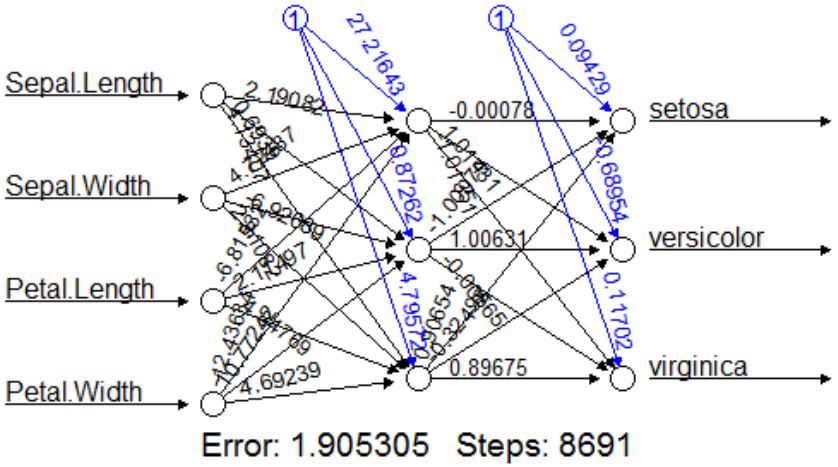


Multinomial Classification using neuralnet

```
data(iris)
formula <- as.formula(paste('Species ~', paste(names(iris)[-length(iris)], collapse='+')))
# neuralnet does not support the '.' notation in the formula.
m2 <- neuralnet(formula, iris, hidden=3, linear.output=FALSE)
# fails !
```

- In neuralnet package, a factor is not accepted as target.
- You have to expand the factor Species to three binary variables first. This works best with the function class.ind() from the nnet package.

```
trainData <- cbind(iris[, 1:4], class.ind(iris$Species))
m2 <- neuralnet(setosa + versicolor + virginica ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, trainData, hidden=3)
```

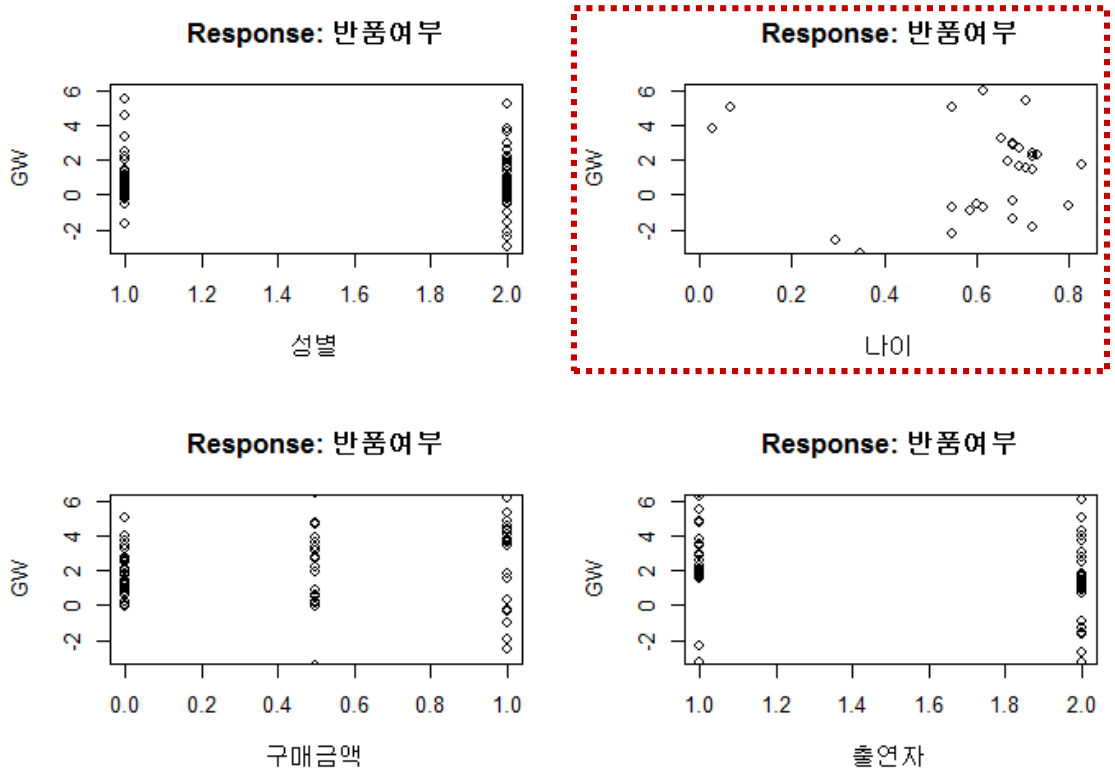


Input Normalization in Neural Networks

```
normalize <- function (x) {
   normalized = (x - min(x)) / (max(x) - min(x))
   return(normalized)
}

cb <- read.delim("Hshopping.txt", stringsAsFactors=FALSE)

cb$나이 <- normalize(cb$나이)
cb$구매금액 <- normalize(cb$구매금액)

set.seed(1)
inTrain <- createDataPartition(y=cb$반품여부, p=0.6, list=FALSE)
cb.train <- cb[inTrain,]
cb.test <- cb[-inTrain,]

set.seed(123)
nn3_model <- neuralnet(반품여부 ~ 성별+나이+구매금액+출연자,
data=cb.train, hidden=3, threshold=0.01)
```

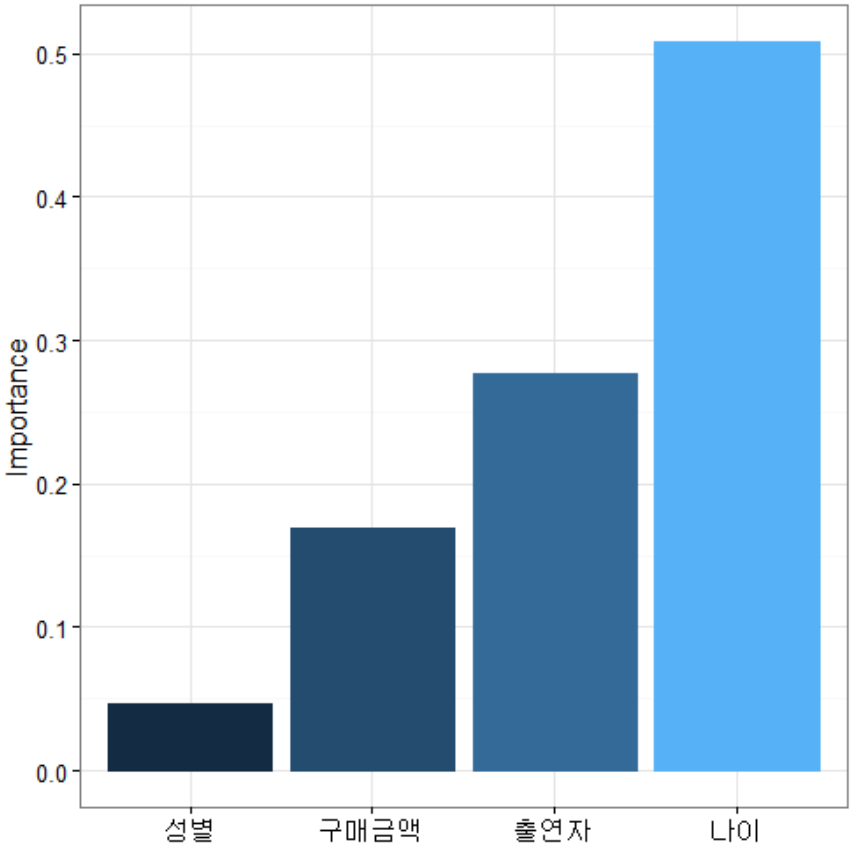
Input Normalization in Neural Networks

```
par(mfrow=c(2,2))
gwplot(nn3_model, selected.covariate = "성별", min=-3, ,max=6)
gwplot(nn3_model, selected.covariate = "나이", min=-3, ,max=6)
gwplot(nn3_model, selected.covariate = "구매금액", min=-3, ,max=6)
gwplot(nn3_model, selected.covariate = "출연자", min=-3, ,max=6)
par(mfrow=c(1,1))
```



Input Normalization in Neural Networks

- `garson(nn3_model)`



Input Normalization in Neural Networks

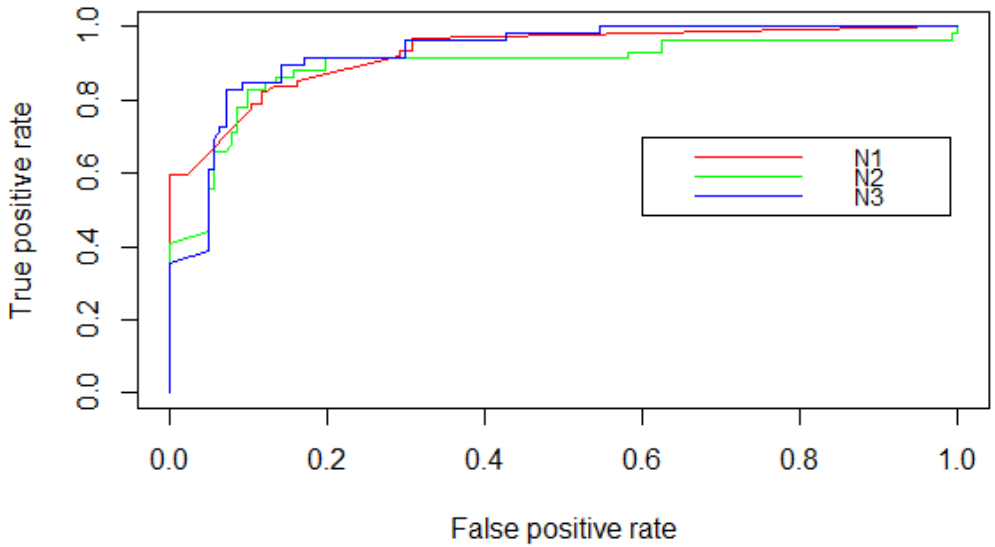
- `cb.test$nn3_pred_prob <- compute(nn3_model, covariate=cb.test[, c(2:5)])$net.result`
- `cb.test$nn3_pred <- ifelse(cb.test$nn3_pred_prob > 0.5, 1, 0)`
- `confusionMatrix(cb.test$nn3_pred, cb.test$반품여부)`

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 131  12
         1  10  47

               Accuracy : 0.89
                 95% CI : (0.8382038, 0.9297668)
    No Information Rate : 0.705
    P-Value [Acc > NIR] : 0.0000000003254392

                  Kappa : 0.7329125
 Mcnemar's Test P-Value : 0.8311704

            Sensitivity : 0.9290780
            Specificity : 0.7966102
         Pos Pred Value : 0.9160839
         Neg Pred Value : 0.8245614
             Prevalence : 0.7050000
         Detection Rate : 0.6550000
   Detection Prevalence : 0.7150000
      Balanced Accuracy : 0.8628441

       'Positive' Class : 0
```

0.87 -> 0.89로 성능이 개선됨

Model Comparison

```
nn3_pred <- ROCR::prediction(cb.test$nn3_pred_prob, cb.test$반품여부)
nn3_model.perf1 <- performance(nn3_pred, "tpr", "fpr") # ROC-chart

plot(nn_model.perf1, col="red")
plot(nn2_model.perf1, col="green", add=T)
plot(nn3_model.perf1, col="blue", add=T)
legend(0.6,0.7,c("N1","N2","N3"),cex=0.9,col=c("red","green","blue"),lty=1)
```




```
performance(nn_pred, "auc")@y.values[[1]]; performance(nn2_pred,
"auc")@y.values[[1]]; performance(nn3_pred, "auc")@y.values[[1]]
[1] 0.9286555215
[1] 0.8942781584
[1] 0.9308210121
```