FinalExam

강지훈 (2018123223)

12/17/2020

# 1. 데이터 불러오기  
df = read.csv("4194\_2020fall/WA\_Fn-UseC\_-Telco-Customer-Churn.csv", na.strings ="", stringsAsFactors = TRUE)  
  
#str(df)  
  
# 2. 데이터 ID 열 제거   
df = df[,c(-1)]  
#str(df)  
  
# 3. 결측치 확인과 제거  
sum(is.na(df))

## [1] 11

df = na.omit(df)  
sum(is.na(df))

## [1] 0

# 4. 데이터 factor 화  
df$SeniorCitizen<-factor(df$SeniorCitizen)  
  
# 5. binning을 위한 점검  
#hist(df$MonthlyCharges)  
#hist(df$TotalCharges)  
#hist(df$tenure)  
quantile(df$MonthlyCharges, probs = seq(0, 1, 0.25))

## 0% 25% 50% 75% 100%   
## 18.2500 35.5875 70.3500 89.8625 118.7500

quantile(df$TotalCharges, probs = seq(0, 1, 0.25))

## 0% 25% 50% 75% 100%   
## 18.800 401.450 1397.475 3794.738 8684.800

quantile(df$tenure, probs = seq(0, 1, 0.25))

## 0% 25% 50% 75% 100%   
## 1 9 29 55 72

# 6. bining 후 factor 화 & split data  
df$tenure = factor(round(df$tenure/10))  
df$MonthlyCharges = factor(round(df$MonthlyCharges/10))  
df$TotalCharges = factor(round(df$TotalCharges/1000))  
#str(df)  
#View(df)  
  
set.seed(2)  
train.index = sample(c(1:dim(df)[1]), dim(df)[1]\*0.6 )  
  
train.df = df[train.index,]  
valid.df = df[-train.index,]  
  
dim(train.df)

## [1] 4219 20

dim(valid.df)

## [1] 2813 20

#str(train.df)  
  
# 7. Navie Bayes 적용  
library("e1071")  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

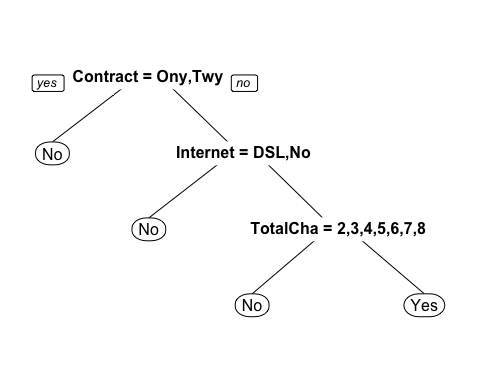
df.nb = naiveBayes(Churn~., data=train.df)  
  
pred.train.class = predict(df.nb, newdata = train.df)  
confusionMatrix(pred.train.class, train.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2183 237  
## Yes 922 877  
##   
## Accuracy : 0.7253   
## 95% CI : (0.7115, 0.7387)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : 0.9435   
##   
## Kappa : 0.4096   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7031   
## Specificity : 0.7873   
## Pos Pred Value : 0.9021   
## Neg Pred Value : 0.4875   
## Prevalence : 0.7360   
## Detection Rate : 0.5174   
## Detection Prevalence : 0.5736   
## Balanced Accuracy : 0.7452   
##   
## 'Positive' Class : No   
##

pred.valid.class = predict(df.nb, newdata = valid.df)  
confusionMatrix(pred.valid.class, valid.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1457 124  
## Yes 601 631  
##   
## Accuracy : 0.7423   
## 95% CI : (0.7257, 0.7584)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : 0.1044   
##   
## Kappa : 0.4531   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.7080   
## Specificity : 0.8358   
## Pos Pred Value : 0.9216   
## Neg Pred Value : 0.5122   
## Prevalence : 0.7316   
## Detection Rate : 0.5180   
## Detection Prevalence : 0.5620   
## Balanced Accuracy : 0.7719   
##   
## 'Positive' Class : No   
##

# 8. Decision Tree 적용  
  
# 8.1 binning 된 기존 데이터 활용  
  
library("rpart")  
library("rpart.plot")  
defalut.ct = rpart(Churn~., data=train.df, method="class")  
prp(defalut.ct)



# str(train.df)  
  
defalut.point.pred.train = predict(defalut.ct, train.df, type="class")  
confusionMatrix(defalut.point.pred.train, train.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2897 693  
## Yes 208 421  
##   
## Accuracy : 0.7864   
## 95% CI : (0.7738, 0.7987)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : 1.632e-14   
##   
## Kappa : 0.3614   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9330   
## Specificity : 0.3779   
## Pos Pred Value : 0.8070   
## Neg Pred Value : 0.6693   
## Prevalence : 0.7360   
## Detection Rate : 0.6867   
## Detection Prevalence : 0.8509   
## Balanced Accuracy : 0.6555   
##   
## 'Positive' Class : No   
##

defalut.point.pred.valid = predict(defalut.ct, valid.df, type="class")  
confusionMatrix(defalut.point.pred.valid, valid.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1908 427  
## Yes 150 328  
##   
## Accuracy : 0.7949   
## 95% CI : (0.7795, 0.8097)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : 4.114e-15   
##   
## Kappa : 0.4091   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9271   
## Specificity : 0.4344   
## Pos Pred Value : 0.8171   
## Neg Pred Value : 0.6862   
## Prevalence : 0.7316   
## Detection Rate : 0.6783   
## Detection Prevalence : 0.8301   
## Balanced Accuracy : 0.6808   
##   
## 'Positive' Class : No   
##

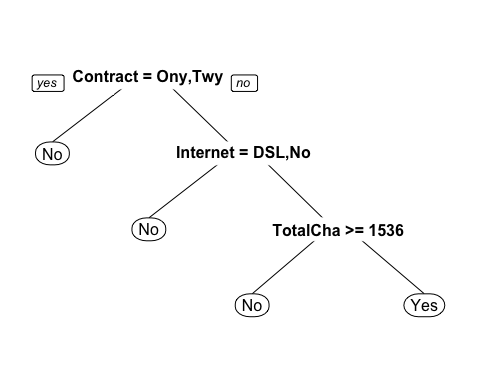
# 8.2 binning 전 Continuous 데이터 활용  
before.df = read.csv("4194\_2020fall/WA\_Fn-UseC\_-Telco-Customer-Churn.csv", na.strings ="", stringsAsFactors = TRUE)  
#str(before.df)  
  
# 8.2.1 데이터 전처리 및 split  
before.df = before.df[,c(-1)]  
sum(is.na(before.df))

## [1] 11

before.df = na.omit(before.df)  
sum(is.na(before.df))

## [1] 0

before.df$SeniorCitizen<-factor(df$SeniorCitizen)  
train.bf = before.df[train.index , ]  
valid.bf = before.df[-train.index, ]  
  
# 8.2.2 Decision Tree 적용  
before.ct = rpart(Churn~., data=train.bf, method="class")  
prp(before.ct)



#str(train.bf)  
  
before.point.pred.train = predict(before.ct, train.bf, type="class")  
confusionMatrix(before.point.pred.train, train.bf$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2897 687  
## Yes 208 427  
##   
## Accuracy : 0.7879   
## 95% CI : (0.7752, 0.8001)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : 2.925e-15   
##   
## Kappa : 0.3669   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9330   
## Specificity : 0.3833   
## Pos Pred Value : 0.8083   
## Neg Pred Value : 0.6724   
## Prevalence : 0.7360   
## Detection Rate : 0.6867   
## Detection Prevalence : 0.8495   
## Balanced Accuracy : 0.6582   
##   
## 'Positive' Class : No   
##

before.point.pred.valid = predict(before.ct, valid.bf, type="class")  
confusionMatrix(before.point.pred.valid, valid.bf$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1905 423  
## Yes 153 332  
##   
## Accuracy : 0.7952   
## 95% CI : (0.7798, 0.81)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : 2.878e-15   
##   
## Kappa : 0.412   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9257   
## Specificity : 0.4397   
## Pos Pred Value : 0.8183   
## Neg Pred Value : 0.6845   
## Prevalence : 0.7316   
## Detection Rate : 0.6772   
## Detection Prevalence : 0.8276   
## Balanced Accuracy : 0.6827   
##   
## 'Positive' Class : No   
##

# 9.1 기존 데이터에 randomForest 적용  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

rf = randomForest(Churn~., data = train.df, ntree=500, importance = TRUE)  
  
rf.pred.train = predict(rf, train.df)  
confusionMatrix(rf.pred.train, train.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3029 130  
## Yes 76 984  
##   
## Accuracy : 0.9512   
## 95% CI : (0.9442, 0.9575)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8724   
##   
## Mcnemar's Test P-Value : 0.0002219   
##   
## Sensitivity : 0.9755   
## Specificity : 0.8833   
## Pos Pred Value : 0.9588   
## Neg Pred Value : 0.9283   
## Prevalence : 0.7360   
## Detection Rate : 0.7179   
## Detection Prevalence : 0.7488   
## Balanced Accuracy : 0.9294   
##   
## 'Positive' Class : No   
##

rf.pred.valid = predict(rf, valid.df)  
confusionMatrix(rf.pred.valid, valid.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1858 358  
## Yes 200 397  
##   
## Accuracy : 0.8016   
## 95% CI : (0.7864, 0.8162)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4591   
##   
## Mcnemar's Test P-Value : 3.005e-11   
##   
## Sensitivity : 0.9028   
## Specificity : 0.5258   
## Pos Pred Value : 0.8384   
## Neg Pred Value : 0.6650   
## Prevalence : 0.7316   
## Detection Rate : 0.6605   
## Detection Prevalence : 0.7878   
## Balanced Accuracy : 0.7143   
##   
## 'Positive' Class : No   
##

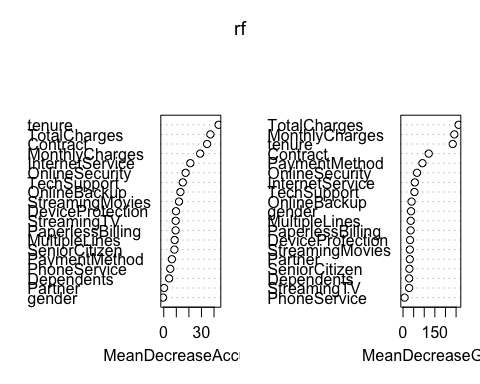
# 9.2 binning 전 Continuous 데이터 활용  
  
rf = randomForest(Churn~., data = train.bf, ntree=500, importance = TRUE)  
rf.pred.train = predict(rf, train.bf)  
confusionMatrix(rf.pred.train, train.bf$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3087 54  
## Yes 18 1060  
##   
## Accuracy : 0.9829   
## 95% CI : (0.9786, 0.9866)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9556   
##   
## Mcnemar's Test P-Value : 3.711e-05   
##   
## Sensitivity : 0.9942   
## Specificity : 0.9515   
## Pos Pred Value : 0.9828   
## Neg Pred Value : 0.9833   
## Prevalence : 0.7360   
## Detection Rate : 0.7317   
## Detection Prevalence : 0.7445   
## Balanced Accuracy : 0.9729   
##   
## 'Positive' Class : No   
##

rf.pred.valid = predict(rf, valid.bf)  
confusionMatrix(rf.pred.valid, valid.bf$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1867 355  
## Yes 191 400  
##   
## Accuracy : 0.8059   
## 95% CI : (0.7908, 0.8204)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4693   
##   
## Mcnemar's Test P-Value : 3.042e-12   
##   
## Sensitivity : 0.9072   
## Specificity : 0.5298   
## Pos Pred Value : 0.8402   
## Neg Pred Value : 0.6768   
## Prevalence : 0.7316   
## Detection Rate : 0.6637   
## Detection Prevalence : 0.7899   
## Balanced Accuracy : 0.7185   
##   
## 'Positive' Class : No   
##

# 10. NN을 위한변수 7개 선택  
varImpPlot(rf)



# MeanDecreaseAccuracy -> 선택  
# Contract, tenure, MonthlyCharges, InternetService, OnlineSecurity  
# TotalCharges, TechSupprot  
  
# MeanDecreaseGINI  
# tenure, MonthlyCharges, Contract, Paymentmethod, OnlineSecurity, InternetService  
# Tech Support  
  
  
# 11. MeanDecreaseAccuracy를 사용한 neuralnet을 위한 데이터 및 nnet 불러오기  
library(nnet)   
library(neuralnet)  
  
NeuralNet = read.csv("4194\_2020fall/WA\_Fn-UseC\_-Telco-Customer-Churn.csv", na.strings ="", stringsAsFactors = TRUE)  
#str(NeuralNet)  
summary(NeuralNet)

## customerID gender SeniorCitizen Partner Dependents  
## 0002-ORFBO: 1 Female:3488 Min. :0.0000 No :3641 No :4933   
## 0003-MKNFE: 1 Male :3555 1st Qu.:0.0000 Yes:3402 Yes:2110   
## 0004-TLHLJ: 1 Median :0.0000   
## 0011-IGKFF: 1 Mean :0.1621   
## 0013-EXCHZ: 1 3rd Qu.:0.0000   
## 0013-MHZWF: 1 Max. :1.0000   
## (Other) :7037   
## tenure PhoneService MultipleLines InternetService  
## Min. : 0.00 No : 682 No :3390 DSL :2421   
## 1st Qu.: 9.00 Yes:6361 No phone service: 682 Fiber optic:3096   
## Median :29.00 Yes :2971 No :1526   
## Mean :32.37   
## 3rd Qu.:55.00   
## Max. :72.00   
##   
## OnlineSecurity OnlineBackup   
## No :3498 No :3088   
## No internet service:1526 No internet service:1526   
## Yes :2019 Yes :2429   
##   
##   
##   
##   
## DeviceProtection TechSupport   
## No :3095 No :3473   
## No internet service:1526 No internet service:1526   
## Yes :2422 Yes :2044   
##   
##   
##   
##   
## StreamingTV StreamingMovies Contract   
## No :2810 No :2785 Month-to-month:3875   
## No internet service:1526 No internet service:1526 One year :1473   
## Yes :2707 Yes :2732 Two year :1695   
##   
##   
##   
##   
## PaperlessBilling PaymentMethod MonthlyCharges   
## No :2872 Bank transfer (automatic):1544 Min. : 18.25   
## Yes:4171 Credit card (automatic) :1522 1st Qu.: 35.50   
## Electronic check :2365 Median : 70.35   
## Mailed check :1612 Mean : 64.76   
## 3rd Qu.: 89.85   
## Max. :118.75   
##   
## TotalCharges Churn   
## Min. : 18.8 No :5174   
## 1st Qu.: 401.4 Yes:1869   
## Median :1397.5   
## Mean :2283.3   
## 3rd Qu.:3794.7   
## Max. :8684.8   
## NA's :11

# 12. 변수 선택, 결측치 제거  
# Contract, tenure, MonthlyCharges, InternetService, OnlineSecurity  
# TotalCharges, TechSupprot  
  
NN = NeuralNet[,c(6,9,10,13,16,19,20,21)]  
#str(NN)  
  
sum(is.na(NN))

## [1] 11

NN = na.omit(NN)  
sum(is.na(NN))

## [1] 0

# 13. 정규화  
max(NN$TotalCharges)

## [1] 8684.8

NN$tenure = (NN$tenure-min(NN$tenure))/(max(NN$tenure)-min(NN$tenure))  
NN$MonthlyCharges = (NN$MonthlyCharges-min(NN$MonthlyCharges))/(max(NN$MonthlyCharges)-min(NN$MonthlyCharges))  
NN$TotalCharges = (NN$TotalCharges-min(NN$TotalCharges))/(max(NN$TotalCharges)-min(NN$TotalCharges))  
  
  
#NN$tenure  
#NN$MonthlyCharges  
#NN$TotalCharges  
  
#str(NN)  
  
# 14. cbind와 paste를 이용한 더미화  
NN.df1 = cbind(NN,  
 class.ind(NN$InternetService),  
 class.ind(NN$OnlineSecurity),   
 class.ind(NN$TechSupport),  
 class.ind(NN$Contract),  
 class.ind(NN$Churn))  
#NN.df1  
vars = c("tenure", "InternetService", "OnlineSecurity", "TechSupport", "Contract"  
 , "MonthlyCharges", "TotalCharges", "Churn")  
names(NN.df1) = c(vars,  
 paste("Internet\_", c(1, 2, 3), sep=""),  
 paste("Online\_", c(1, 2, 3), sep=""),  
 paste("Tech\_", c(1, 2, 3), sep=""),  
 paste("Contract\_", c(1, 2, 3), sep=""),  
 paste("Churn\_", c("no", "yes"), sep=""))  
  
#NN.df1  
colnames(NN.df1)

## [1] "tenure" "InternetService" "OnlineSecurity" "TechSupport"   
## [5] "Contract" "MonthlyCharges" "TotalCharges" "Churn"   
## [9] "Internet\_1" "Internet\_2" "Internet\_3" "Online\_1"   
## [13] "Online\_2" "Online\_3" "Tech\_1" "Tech\_2"   
## [17] "Tech\_3" "Contract\_1" "Contract\_2" "Contract\_3"   
## [21] "Churn\_no" "Churn\_yes"

# 15. 더미화 된 변수 중 기존 변수 및 churn 제거  
NN.df2 = NN.df1[, -c(2, 3, 4, 5, 8)]  
colnames(NN.df2)

## [1] "tenure" "MonthlyCharges" "TotalCharges" "Internet\_1"   
## [5] "Internet\_2" "Internet\_3" "Online\_1" "Online\_2"   
## [9] "Online\_3" "Tech\_1" "Tech\_2" "Tech\_3"   
## [13] "Contract\_1" "Contract\_2" "Contract\_3" "Churn\_no"   
## [17] "Churn\_yes"

# 16. split data  
set.seed(2)  
train.index = sample(rownames(NN.df2), dim(NN.df2)[1]\*0.6)  
valid.index = setdiff(row.names(NN.df2), train.index)  
  
train.NN = NN.df2[train.index,]  
valid.NN = NN.df2[valid.index,]  
dim(train.NN)

## [1] 4219 17

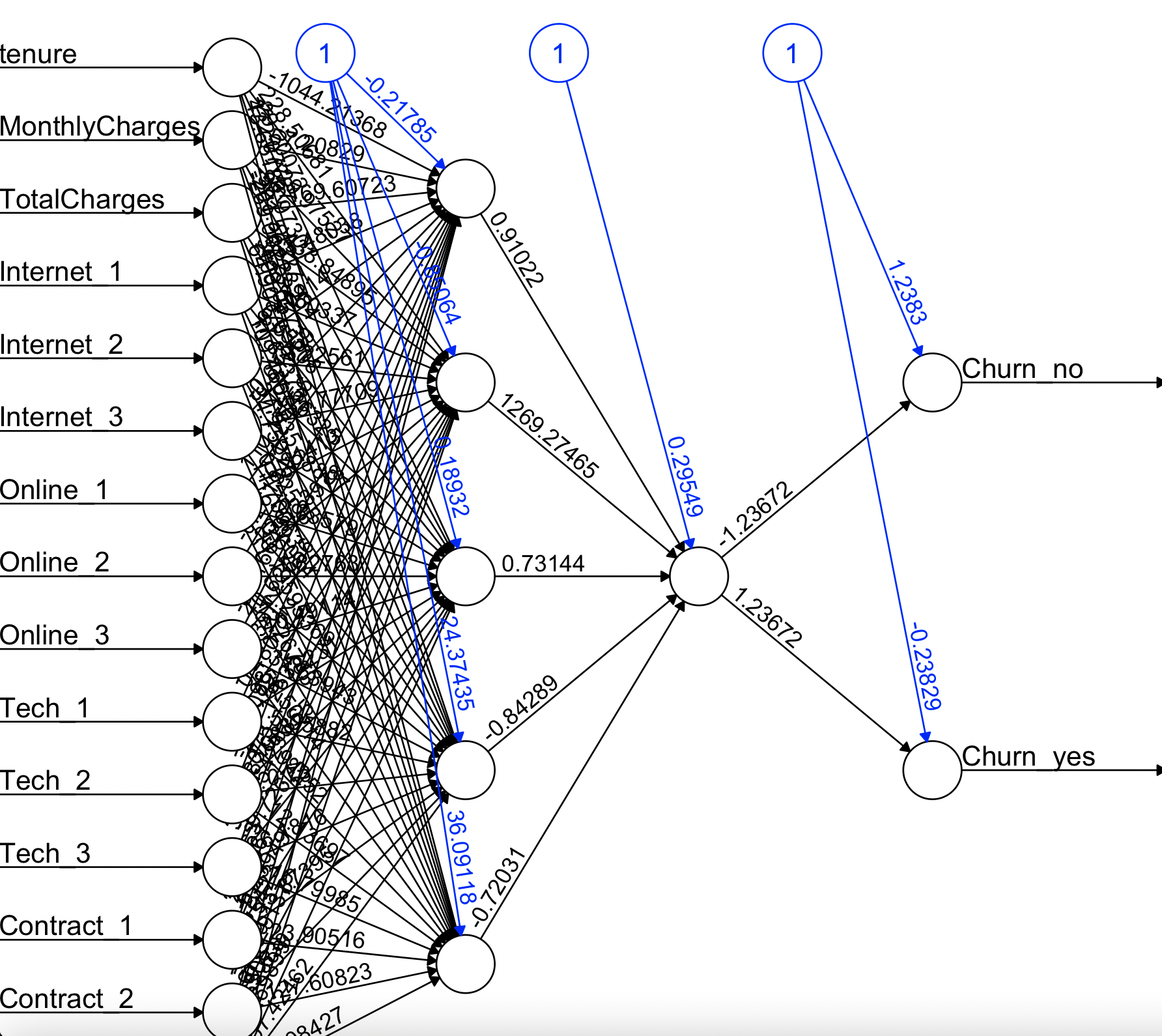
dim(valid.NN)

## [1] 2813 17

colnames(train.NN)

## [1] "tenure" "MonthlyCharges" "TotalCharges" "Internet\_1"   
## [5] "Internet\_2" "Internet\_3" "Online\_1" "Online\_2"   
## [9] "Online\_3" "Tech\_1" "Tech\_2" "Tech\_3"   
## [13] "Contract\_1" "Contract\_2" "Contract\_3" "Churn\_no"   
## [17] "Churn\_yes"

# 17. neuralnet 적용 및 정확도 확인  
nn1 = neuralnet(Churn\_no + Churn\_yes~   
 tenure + MonthlyCharges + TotalCharges +  
 Internet\_1 + Internet\_2 + Internet\_3  
 + Online\_1 + Online\_2 + Online\_3 +  
 Tech\_1 + Tech\_2 + Tech\_3 +  
 Contract\_1 + Contract\_2 + Contract\_3  
 , data= train.NN, hidden = c(5,1), stepmax = 1e+06)  
plot(nn1)

  
  
training.prediction = compute(nn1, train.NN[,-c(16:17)])  
training.class = apply(training.prediction$net.result, 1, which.max) - 1  
# training.class  
NeuralNet$Churn = as.numeric(factor(NeuralNet$Churn)) - 1  
confusionMatrix(factor(training.class), factor(NeuralNet[train.index,]$Churn))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 2849 575  
## 1 256 539  
##   
## Accuracy : 0.803   
## 95% CI : (0.7907, 0.8149)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.442   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9176   
## Specificity : 0.4838   
## Pos Pred Value : 0.8321   
## Neg Pred Value : 0.6780   
## Prevalence : 0.7360   
## Detection Rate : 0.6753   
## Detection Prevalence : 0.8116   
## Balanced Accuracy : 0.7007   
##   
## 'Positive' Class : 0   
##

validation.prediction = compute(nn1, valid.NN[,-c(16:17)])  
validation.class = apply(validation.prediction$net.result, 1, which.max) -1  
confusionMatrix(factor(validation.class), factor(NeuralNet[valid.index,]$Churn))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 1869 367  
## 1 189 388  
##   
## Accuracy : 0.8023   
## 95% CI : (0.7871, 0.8169)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4561   
##   
## Mcnemar's Test P-Value : 6.074e-14   
##   
## Sensitivity : 0.9082   
## Specificity : 0.5139   
## Pos Pred Value : 0.8359   
## Neg Pred Value : 0.6724   
## Prevalence : 0.7316   
## Detection Rate : 0.6644   
## Detection Prevalence : 0.7949   
## Balanced Accuracy : 0.7110   
##   
## 'Positive' Class : 0   
##

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy (train / valid) | Sensitivity(train / valid) | Specificity(train / valid) |
| Navie Bayes | 0.7253 / 0.7423 | 0.7031 / 0.7080 | 0.7873 / 0.8358 |
| Decision tree | 0.7864 / 0.7949 | 0.9330 / 0.9271 | 0.3779 / 0.4344 |
| RandomForest | 0.9512 / 0.8016 | 0.9755 / 0.9028 | 0.8833 / 0.5258 |
| NeuralNet | 0.803 / 0.8023 | 0.9176 / 0.9082 | 0.4838 / 0.5139 |

Cf.) Decision tree와 랜덤포레스트에서 continuous 변수 그대로 사용했을 시 비교

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy (train / valid) | Sensitivity(train / valid) | Specificity(train / valid) |
| Decision tree(factor) | 0.7864 / 0.7949 | 0.9330 / 0.9271 | 0.3779 / 0.4344 |
| Decision tree(con) | 0.7879 / 0.7952 | 0.9330/ 0.9257 | 0.3833 / 0.4397 |
| RandomForest(Factor) | 0.9512 / 0.8016 | 0.9755 / 0.9028 | 0.8833 / 0.5258 |
| RandomForest(Con) | 0.9829 / 0.8059 | 0.9942 / 0.9072 | 0.9515 / 0.5298 |

# 보너스: Accuracy 높이기 위한 전략  
  
# 1. RandomForest에서 Ntree 값 조정

# 기존 데이터에 randomForest 적용 # Ntree 500 -> 400  
library(randomForest)  
rf = randomForest(Churn~., data = train.df, ntree=400, importance = TRUE)  
  
# 0.9526(500) -> 0.9512(400)  
rf.pred.train = predict(rf, train.df)  
confusionMatrix(rf.pred.train, train.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3031 132  
## Yes 74 982  
##   
## Accuracy : 0.9512   
## 95% CI : (0.9442, 0.9575)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.8722   
##   
## Mcnemar's Test P-Value : 7.146e-05   
##   
## Sensitivity : 0.9762   
## Specificity : 0.8815   
## Pos Pred Value : 0.9583   
## Neg Pred Value : 0.9299   
## Prevalence : 0.7360   
## Detection Rate : 0.7184   
## Detection Prevalence : 0.7497   
## Balanced Accuracy : 0.9288   
##   
## 'Positive' Class : No   
##

# 0.8013(500) -> 0.8031 (400)  
rf.pred.valid = predict(rf, valid.df)  
confusionMatrix(rf.pred.valid, valid.df$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1855 351  
## Yes 203 404  
##   
## Accuracy : 0.8031   
## 95% CI : (0.7879, 0.8176)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4653   
##   
## Mcnemar's Test P-Value : 4.226e-10   
##   
## Sensitivity : 0.9014   
## Specificity : 0.5351   
## Pos Pred Value : 0.8409   
## Neg Pred Value : 0.6656   
## Prevalence : 0.7316   
## Detection Rate : 0.6594   
## Detection Prevalence : 0.7842   
## Balanced Accuracy : 0.7182   
##   
## 'Positive' Class : No   
##

# binning 전 Continuous 데이터 활용 # Ntree 500 -> 400  
  
rf = randomForest(Churn~., data = train.bf, ntree=400, importance = TRUE)  
  
# 0.9839(500) -> 0.9834(400)  
rf.pred.train = predict(rf, train.bf)  
confusionMatrix(rf.pred.train, train.bf$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 3090 55  
## Yes 15 1059  
##   
## Accuracy : 0.9834   
## 95% CI : (0.9791, 0.987)  
## No Information Rate : 0.736   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9568   
##   
## Mcnemar's Test P-Value : 3.141e-06   
##   
## Sensitivity : 0.9952   
## Specificity : 0.9506   
## Pos Pred Value : 0.9825   
## Neg Pred Value : 0.9860   
## Prevalence : 0.7360   
## Detection Rate : 0.7324   
## Detection Prevalence : 0.7454   
## Balanced Accuracy : 0.9729   
##   
## 'Positive' Class : No   
##

# 0.8091(500) -> 0.8041(400)  
rf.pred.valid = predict(rf, valid.bf)  
confusionMatrix(rf.pred.valid, valid.bf$Churn)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 1864 357  
## Yes 194 398  
##   
## Accuracy : 0.8041   
## 95% CI : (0.789, 0.8186)  
## No Information Rate : 0.7316   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.4646   
##   
## Mcnemar's Test P-Value : 5.148e-12   
##   
## Sensitivity : 0.9057   
## Specificity : 0.5272   
## Pos Pred Value : 0.8393   
## Neg Pred Value : 0.6723   
## Prevalence : 0.7316   
## Detection Rate : 0.6626   
## Detection Prevalence : 0.7895   
## Balanced Accuracy : 0.7164   
##   
## 'Positive' Class : No   
##