R Exercise: Ensemble Classification

Pilsung Kang

School of Industrial Management Engineering

Korea University

R Exercise: Ensemble Classification

- Data Set: Personal Loan Prediction

Data Description:

ID	Customer ID
Age	Customer's Age in completed years
Experience	#years of professional experience
Income	Annual income of the customer ($000)
ZIPCode	Home Address ZIP code.
Family	Family size (dependents) of the customer
CCAvg	Avg. Spending on Credit Cards per month ($000)
Education	Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
Mortgage	Value of house mortgage if any. ($000)
Personal Loan	Did this customer accept the personal loan offered in the last campaign?
Securities Account	Does the customer have a Securities account with the bank?
CD Account	Does the customer have a Certificate of Deposit (CD) account with the bank?
Online	Does the customer use internet banking facilities?
CreditCard	Does the customer use a credit card issued by UniversalBank?

R Exercise: Ensemble Classification

- Purpose
  - ✓ Compare the classification performances of single classifiers and ensemble models
    - Single classifier: Artificial Neural Network (ANN), Classification Tree
    - Ensemble classifier: Bagging with ANN, Random Forests, AdaBoost with Stump Tree, Gradient Boosting Machine (GBM) with Stump Tree
  - ✓ Experimental Settings
    - Use the first 1,500 examples for training and the remaining 1,000 examples for test
    - Use the best parameter found in the previous R exercise for ANN and Classification Tree
    - Use the same parameter for Bagging with ANN

R Exercise: Ensemble Classification

- Create a performance evaluation function

```
# Performance Evaluation Function ------------------------------------------------
perf_eval <- function(cm){
     # True positive rate: TPR (Recall)
     TPR <- cm[2,2]/sum(cm[2,])
     # Precision
     PRE <- cm[2,2]/sum(cm[,2])
     # True negative rate: TNR
     TNR <- cm[1,1]/sum(cm[1,])
     # Simple Accuracy
     ACC <- (cm[1,1]+cm[2,2])/sum(cm)
     # Balanced Correction Rate
     BCR <- sqrt(TPR*TNR)
     # F1-Measure
     F1 <- 2*TPR*PRE/(TPR+PRE)
     return(c(TPR, PRE, TNR, ACC, BCR, F1))
}
Perf.Table <- matrix(0, nrow = 6, ncol = 6)
rownames(Perf.Table) <- c("ANN", "CART", "Bagging ANN", "AdaBoost", "GBM", "Random
              Forests")
colnames(Perf.Table) <- c("TPR", "Precision", "TNR", "Accuracy", "BCR",
              "F1-Measure")
```

R Exercise: Ensemble Classification

- Initialize the performance matrix & Load the dataset

```r
# Part 1: Classification with Single Model ----------------------------------
# Model 1: Artificial Neural Network ----------------------------------------
# nnet package install
install.packages("nnet", dependencies = TRUE)
library(nnet)

# Load the data & Preprocessing
Ploan <- read.csv("Personal Loan.csv")
input.idx <- c(2,3,4,6,7,8,9,11,12,13,14)
target.idx <- 10
Ploan.input <- Ploan[,input.idx]
Ploan.input.scaled <- scale(Ploan.input, center = TRUE, scale = TRUE)
Ploan.target <- as.factor(Ploan[,target.idx])
Ploan.data.scaled <- data.frame(Ploan.input.scaled, Ploan.target)
```

✓ Column 1 & 5: id and zipcode (irrelevant variables)

✓ Column 10: target variable

✓ Numeric input variables are normalized (mean = 0, stdev = 1)

✓ Convert the target variable type: numeric → factor

R Exercise: Ensemble Classification

- Normalize and split the dataset

```
# Divide the dataset into the training dataset and test dataset
trn.idx <- 1:1500
tst.idx <- 1501:2500

# Input/Target configuration
ANN.trn.input <- Ploan.input.scaled[trn.idx,]
ANN.trn.target <- class.ind(Ploan.target[trn.idx])

ANN.tst.input <- Ploan.input.scaled[tst.idx,]
ANN.tst.target <- class.ind(Ploan.target[tst.idx])
```

✓ Use the first 1,500 examples for training and the remaining 1,000 examples for test

✓ Convert the target variable to 1-hot encoding using the class.ind( ) function

R Exercise: Ensemble Classification

- Train a single classifier: ANN

```
# Trainin ANN
ANN.model <- nnet(ANN.trn.input, ANN.trn.target, size = 14,
         decay = 5e-4, maxit = 300)
# Performance evaluation
ANN.prey <- predict(ANN.model, ANN.tst.input)
ANN.cfm <- table(max.col(ANN.tst.target), max.col(ANN.prey))

Perf.Table[1,] <- perf_eval(ANN.cfm)
Perf.Table
```

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
ANN	0.8077	0.8842	0.9877	0.9690	0.8932	0.8442
CART						
Bagging ANN						
AdaBoost						
GBM						
Random Forests						

R Exercise: Ensemble Classification

- Train a single classifier: Classification Tree

```r
# Model 2: Classification Tree -----------------------------------------------
install.packages("party")
library(party)

CART.trn <- data.frame(Ploan.input[trn.idx,], PloanYN = Ploan.target[trn.idx])
CART.tst <- data.frame(Ploan.input[tst.idx,], PloanYN = Ploan.target[tst.idx])

# CART parameters
tree.control = ctree_control(mincriterion = 0.95, minsplit = 10, maxdepth = 0)

# Training the tree
CART.model <- ctree(PloanYN ~ ., data = CART.trn, controls = tree.control)

# Prediction
CART.prey <- predict(CART.model, newdata = CART.tst)
CART.cfm <- table(CART.tst$PloanYN, CART.prey)
Perf.Table[2,] <- perf_eval(CART.cfm)
Perf.Table
```

R Exercise: Ensemble Classification

- Train a single classifier: Classification Tree

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
ANN	0.8077	0.8842	0.9877	0.9690	0.8932	0.8442
CART	0.8173	0.9444	0.9944	0.9760	0.9015	0.8762
Bagging ANN						
AdaBoost						
GBM						
Random Forests						

R Exercise: Ensemble Classification

- Train an ensemble of ANNs

```
# Part 2: Classification with Ensemble Models ------------------------------
# Model 3: Bagging with Neural Network ------------------------------------
# Parallel processing can be possible if your machine has multiple cores/threads

install.packages("caret")
install.packages("doParallel")

library(caret)
library(doParallel)

# Assign the number of cores to be processed in parallel
cl <- makeCluster(1) # the number of cores
registerDoParallel(cl)
```

✓ "caret" package provides Bagging ANN function

✓ "doParallel" package enables multi-core/thread processing

- makeCluster(N): make a cluster with N cores/threads for parallel processing
- registerDoParallel(cl): make "cl" cluster ready

R Exercise: Ensemble Classification

- Train an ensemble of ANNs

```
# Bagging Training
ptm <- proc.time()
Bagging.ANN.model <- avNNet(ANN.trn.input, ANN.trn.target, size = 14, decay = 5e-4,
         repeats = 100, bag = TRUE, allowParallel = TRUE, trace = TRUE)
Bagging.Time <- proc.time() - ptm
Bagging.Time
```

✓ avNNet( ): Bagging with ANN

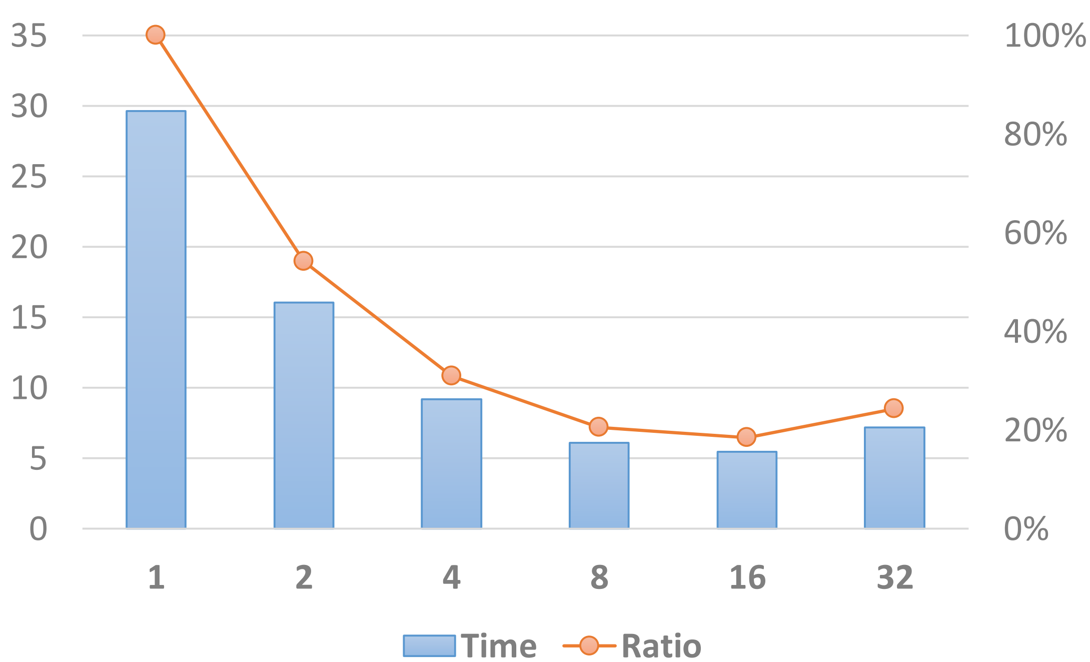
- Arg 1: Input variables of training dataset
- Arg 2: Target variables of training dataset
- Arg 3 & 4: Same as size & decay option of nnet( ) function
- Arg 5: ensemble population size (number of bootstraps)
- Arg 6: option for "sample with replacement" (TRUE: bagging)
- Arg 7: option for parallel processing
- Arg 8: print the progress in the console window

R Exercise: Ensemble Classification

- Train an ensemble of ANNs

```
# Bagging Training
ptm <- proc.time()
Bagging.ANN.model <- avNNet(ANN.trn.input, ANN.trn.target, size = 14, decay = 5e-4,
          repeats = 100, bag = TRUE, allowParallel = TRUE, trace = TRUE)
Bagging.Time <- proc.time() - ptm
Bagging.Time
```

✓ Effect of parallel processing



R Exercise: Ensemble Classification

- Train an ensemble of ANNs

```
# Bagging Test
Bagging.ANN.prey <- predict(Bagging.ANN.model, newdata = ANN.tst.input)
Bagging.ANN.cfm <- table(max.col(ANN.tst.target), max.col(Bagging.ANN.prey))

Perf.Table[3,] <- perf_eval(Bagging.ANN.cfm)
Perf.Table
```

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
ANN	0.8077	0.8842	0.9877	0.9690	0.8932	0.8442
CART	0.8173	0.9444	0.9944	0.9760	0.9015	0.8762
Bagging ANN	0.8173	0.9340	0.9933	0.9750	0.9010	0.8718
AdaBoost						
GBM						
Random Forests						

R Exercise: Ensemble Classification

- Train an AdaBoost with Stump Trees

```
# Model 4: AdaBoost with Stump Tree ------------------------------------
install.packages("ada")
library(ada)

AdaBoost.trn <- CART.trn
AdaBoost.tst <- CART.tst

# Training AdaBoost with Stump Tree (Tree with 1 depth)
ptm <- proc.time()
AdaBoost.model <- ada(AdaBoost.trn[,1:11], AdaBoost.trn[,12], loss = "exponential",
         iter = 100, bag.frac = 0.5, verbose = TRUE)
Boosting.Time <- proc.time() - ptm
```

✓ "ada" package for AdaBoost training

✓ ada( ) function

- Arg 1 & 2: input and target training data
- Arg 3: loss function ("exponential" for classification)
- Arg 4: number of ensemble populations
- Arg 5: boostrap sampling ratio

R Exercise: Ensemble Classification

- Train an AdaBoost with Stump Trees

```r
print(AdaBoost.model)
```

```
> print(AdaBoost.model)
Call:
ada(AdaBoost.trn[, 1:11], y = AdaBoost.trn[, 12], loss = "exponential",
    iter = 100, bag.frac = 0.5, verbose = TRUE)

Loss: exponential Method: discrete   Iteration: 100

Final Confusion Matrix for Data:
          Final Prediction
True value    0    1
         0 1348    0
         1    3  149

Train Error: 0.002

Out-Of-Bag Error:  0.006  iteration= 88

Additional Estimates of number of iterations:

train.err1 train.kap1
        95         95
```

R Exercise: Ensemble Classification

- Train an AdaBoost with Stump Trees

```
# Prediction
AdaBoost.prey <- predict(AdaBoost.model, AdaBoost.tst[,1:11])
AdaBoost.cfm <- table(AdaBoost.tst$PloanYN, AdaBoost.prey)

Perf.Table[4,] <- perf_eval(AdaBoost.cfm)
Perf.Table
```

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
ANN	0.8077	0.8842	0.9877	0.9690	0.8932	0.8442
CART	0.8173	0.9444	0.9944	0.9760	0.9015	0.8762
Bagging ANN	0.8173	0.9340	0.9933	0.9750	0.9010	0.8718
AdaBoost	0.8942	0.9394	0.9933	0.9830	0.9427	0.9163
GBM						
Random Forests						

R Exercise: Ensemble Classification

- Train a GBM with Stump Trees

```r
# Model 5: Gradient Boosting Machine ---------------------------------------
install.packages("gbm")
library(gbm)

GBM.trn <- data.frame(Ploan.input[trn.idx,], PloanYN = Ploan[trn.idx,target.idx])
GBM.tst <- data.frame(Ploan.input[tst.idx,], PloanYN = Ploan[tst.idx,target.idx])
```

  ✓ "gbm" package provide GBM function

R Exercise: Ensemble Classification

- Train a GBM with Stump Trees

```r
# Training the GBM
ptm <- proc.time()
GBM.model <- gbm.fit(GBM.trn[,1:11], GBM.trn[,12], distribution = "bernoulli",
              n.trees = 1000, shrinkage = 0.02,
              bag.fraction = 0.8, nTrain = 1000)
GBM.Time <- proc.time() - ptm
summary(GBM.model)
```

✓ gbm.fit( ): Gradient boosting machine training function

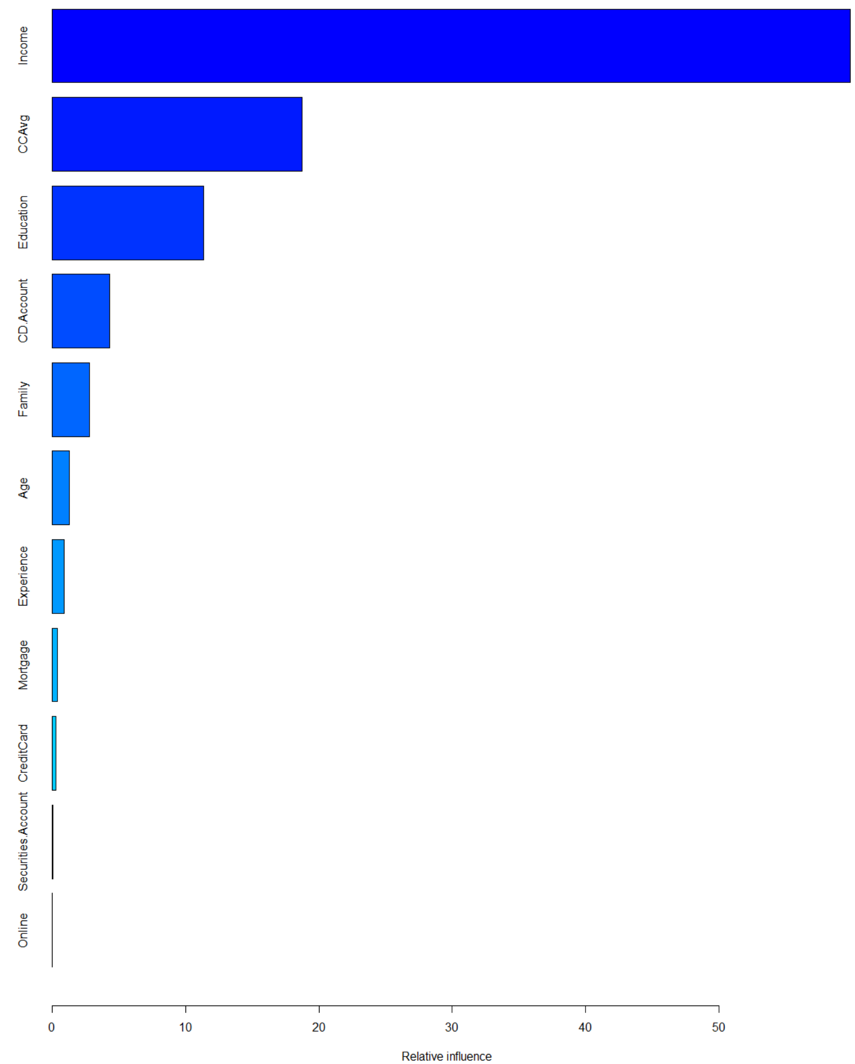
  - Arg 1 & 2: Input & Target variables of training dataset
  - Arg 3: model category ("bernoulli" is used for classification model)
  - Arg 4: Ensemble population (Note: it is better to set a larger number for GBM than AdaBoost and Random Forest)
  - Arg 5: Shrinkage factor to prevent overfitting
  - Arg 6: Bagging sampling ratio to prevent overfitting
  - Arg 7: Maximum number of training examples to prevent overfitting

R Exercise: Ensemble Classification

- Variable importance of GBM
  - ✓ Income > CCAvg > Education

```
> summary(GBM.model)
                                 var     rel.inf
Income                        Income 59.85850376
CCAvg                          CCAvg 18.71848734
Education                  Education 11.35142524
CD.Account                CD.Account  4.29000525
Family                        Family  2.79626171
Age                              Age  1.31475346
Experience                Experience  0.92462237
Mortgage                    Mortgage  0.39585780
CreditCard                CreditCard  0.30314083
Securities.Account Securities.Account  0.04694223
Online                        Online  0.00000000
```

  - ✓ Variable importance of Income (rank 1) is three times higher than that of CCAvg (rank 2)



R Exercise: Ensemble Classification

- GBM Performance

```
# Prediction
GBM.prey <- predict(GBM.model, GBM.tst[,1:11], type = "response")
GBM.prey <- round(GBM.prey)
GBM.cfm <- table(GBM.prey, GBM.tst$PloanYN)
Perf.Table[5,] <- perf_eval(GBM.cfm)
Perf.Table
```

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
ANN	0.8077	0.8842	0.9877	0.9690	0.8932	0.8442
CART	0.8173	0.9444	0.9944	0.9760	0.9015	0.8762
Bagging ANN	0.8173	0.9340	0.9933	0.9750	0.9010	0.8718
AdaBoost	0.8942	0.9394	0.9933	0.9830	0.9427	0.9163
GBM	0.9350	0.6923	0.9653	0.9630	0.9501	0.7956
Random Forests						

R Exercise: Ensemble Classification

- Train a Random Forest

```
# Model 6: Random Forest ----------------------------------------
install.packages("randomForest")
library(randomForest)

RF.trn <- CART.trn
RF.tst <- CART.tst
```

✓ "randomForest" package provides a function to train Random Forest model

✓ Because Random Forest use the decision tree as a base leaner, we use the same dataset used for CART model

R Exercise: Ensemble Classification

- Train a Random Forest

```
# Training the Random Forest
ptm <- proc.time()
RF.model <- randomForest(PloanYN ~ ., data = RF.trn, ntree = 100,
         importance = TRUE, do.trace = TRUE)
RF.Time <- proc.time() - ptm

# Check the result
print(RF.model)
plot(RF.model)
```

✓ randomForest( ): function for training Random Forest

- Arg 1: Formula
- Arg 2: Training dataset
- Arg 3: Number of trees in the ensemble model
- Arg 4: Option for variable importance computation
- Arg 5: print the progress in the console window

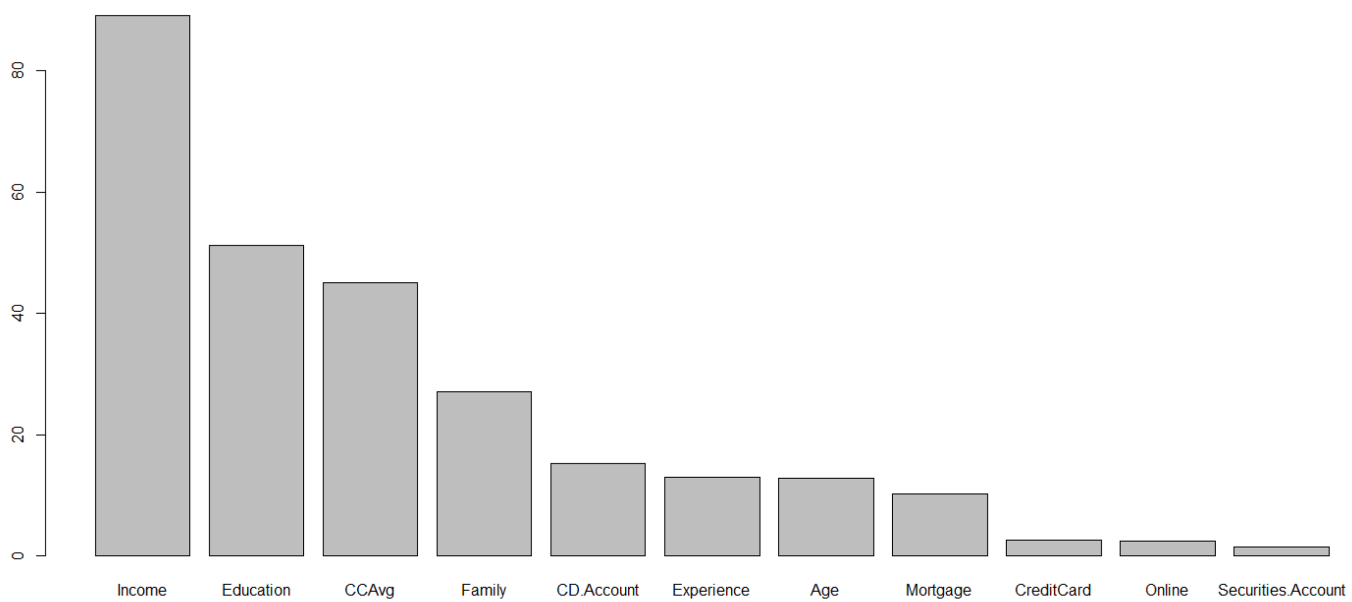
R Exercise: Ensemble Classification

- Random Forest: variable importance

```
# Variable importance
Var.imp <- importance(RF.model)
barplot(Var.imp[order(Var.imp[,4], decreasing = TRUE),4])
```

✓ Top 5 important variables for Random Forest

- Income > Education > CCAvg > Family > CD.Account



R Exercise: Ensemble Classification

- Random Forest: performance

```
# Prediction
RF.prey <- predict(RF.model, newdata = RF.tst, type = "class")
RF.cfm <- table(RF.prey, RF.tst$PloanYN)
Perf.Table[6,] <- perf_eval(RF.cfm)
Perf.Table
```

	TPR	Precision	TNR	Accuracy	BCR	F1-Measure
ANN	0.8077	0.8842	0.9877	0.9690	0.8932	0.8442
CART	0.8173	0.9444	0.9944	0.9760	0.9015	0.8762
Bagging ANN	0.8173	0.9340	0.9933	0.9750	0.9010	0.8718
AdaBoost	0.8942	0.9394	0.9933	0.9830	0.9427	0.9163
GBM	0.9350	0.6923	0.9653	0.9630	0.9501	0.7956
Random Forests	0.9778	0.8462	0.9824	0.9820	0.9801	0.9072


ANY
questions?