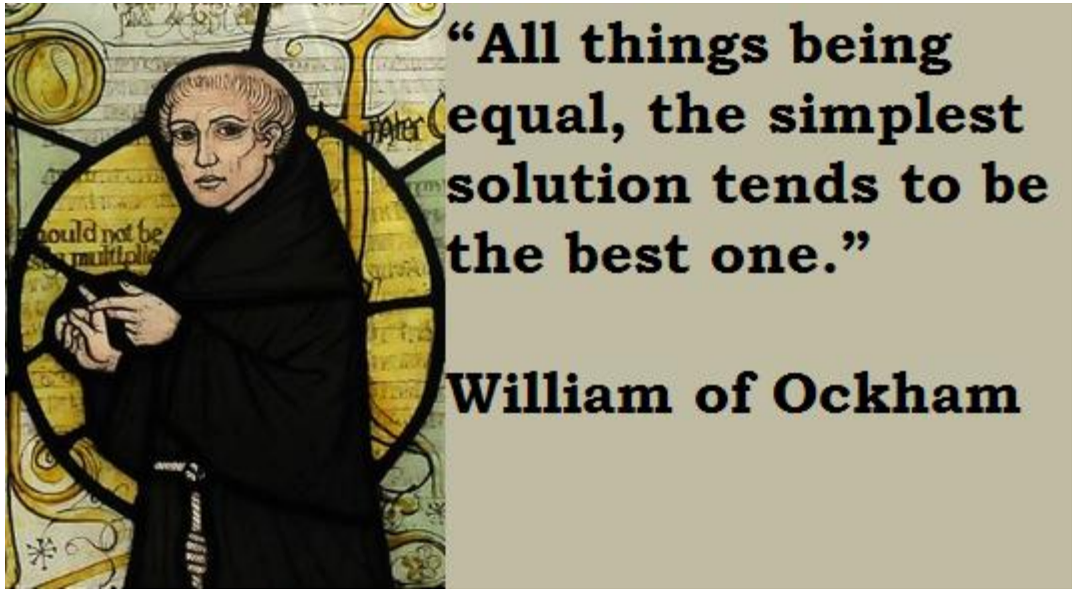


Dimensionality Reduction

Pilsung Kang
School of Industrial Management Engineering
Korea University

AGENDA

R Exercise: Data Set

- Personal Loan
  - ✓ Purpose: identify future customer who will use the personal loan service based on his/her demographic information and banking service history

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ID	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal L	Securities	CD Accou	Online	CreditCard
2	1	25	1	49	91107	4	1.6	1	0	0	1	0	0	0
3	2	45	19	34	90089	3	1.5	1	0	0	1	0	0	0
4	3	39	15	11	94720	1	1	1	0	0	0	0	0	0
5	4	35	9	100	94112	1	2.7	2	0	0	0	0	0	0
6	5	35	8	45	91330	4	1	2	0	0	0	0	0	1
7	6	37	13	29	92121	4	0.4	2	155	0	0	0	1	0
8	7	53	27	72	91711	2	1.5	2	0	0	0	0	1	0
9	8	50	24	22	93943	1	0.3	3	0	0	0	0	0	1
10	9	35	10	81	90089	3	0.6	2	104	0	0	0	1	0

  - A total of 14 variables (columns)
  - ID, ZIP Code: irrelevant column (remove)
  - Personal loan: target variable

R Exercise: Install packages

- Install packages and prepare them to be used

```r
# Install necessary packages
# glmnet: Ridge, Lasso, Elastic Net Logistic Regression
# GA: genetic algorithm
install.packages("glmnet")
install.packages("GA")
library(glmnet)
library(GA)
```

R Exercise: Performance Evaluation Function

- 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))
}
```

✓ Function name: perf_eval

- Argument: confusion matrix
- Outputs: six classification performance metrics

R Exercise: Performance Evaluation Function

- Performance Evaluation Function

```
Perf_Table <- matrix(0, nrow = 8, ncol = 6)
rownames(Perf_Table) <- c("All", "Forward", "Backward", "Stepwise", "GA", "Ridge",
                          "Lasso", "Elastic Net")
colnames(Perf_Table) <- c("TPR", "Precision", "TNR", "Accuracy", "BCR",
                          "F1-Measure")
```

✓ Initialize the performance comparison matrix

✓ A total of 8 logistic regression models are compared

- All: All variables are used
- Forward/Backward/Stepwise: Variables selected by Forward Selection, Backward Elimination, and Stepwise selection are used
- GA: Variables selected by genetic algorithm are used
- Ridge/Lasso/Elastic Net

R Exercise: Data Load and Preprocessing

- Data Load and Preprocessing

```
# Load the data & Preprocessing
Ploan <- read.csv("Personal Loan.csv")

Ploan_input <- Ploan[,-c(1,5,10)]
Ploan_input_scaled <- scale(Ploan_input, center = TRUE, scale = TRUE)
Ploan_target <- Ploan$Personal.Loan
Ploan_data_scaled <- data.frame(Ploan_input_scaled, Ploan_target)

trn_idx <- 1:1500
tst_idx <- 1501:2500

Ploan_trn <- Ploan_data_scaled[trn_idx,]
Ploan_tst <- Ploan_data_scaled[tst_idx,]
```

✓ Remove 1st, 5th, and 10th columns from input variables

✓ Perform input variable normalization

✓ Use the first 1,500 rows for training and the other 1,000 rows for test

R Exercise: Use All Variables

- Logistic Regression 1: All variables

```
# Variable selection method 0: Logistic Regression with all variables
full_model <- glm(Ploan_target ~ ., family=binomial, Ploan_trn)
summary(full_model)
full_model_coeff <- as.matrix(full_model$coefficients, 12, 1)
```

✓ glm( ) function: provide logistic regression model

- Arg 1: Formula, "Target ~ Input" form, period(.) for input means all variables except the target variable are used as input variables
- Arg 2: dataset for training

✓ summary( ): provide summarized information of the trained model

✓ Store the regression coefficients for further comparison

R Exercise: Use All Variables

- Logistic Regression 1: All variables
  - ✓ Insignificant variables (alpha = 0.05)
    - Age
    - Experience
    - Mortgage
    - Online

```
> summary(full_model)

Call:
glm(formula = Ploan_target ~ ., family = binomial, data = Ploan_trn)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.1781  -0.2189  -0.0906  -0.0365   3.5345  

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)    
(Intercept)        -4.445483   0.272085 -16.339  < 2e-16 ***
Age                -0.776031   1.276510  -0.608 0.543233    
Experience          0.910984   1.272791   0.716 0.474154    
Income              2.374701   0.206877  11.479  < 2e-16 ***
Family              0.739703   0.153237   4.827 1.38e-06 ***
CCAvg               0.264244   0.120946   2.185 0.028902 *  
Education           1.328860   0.170227   7.806 5.88e-15 ***
Mortgage            0.009294   0.100392   0.093 0.926239    
Securities.Account -0.501459   0.183861  -2.727 0.006384 ** 
CD.Account          0.982082   0.151231   6.494 8.36e-11 ***
Online             -0.182069   0.139815  -1.302 0.192843    
CreditCard         -0.617374   0.181094  -3.409 0.000652 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 984.01  on 1499  degrees of freedom
Residual deviance: 401.11  on 1488  degrees of freedom
AIC: 425.11

Number of Fisher Scoring iterations: 7
```

R Exercise: Use All Variables

- Logistic Regression 1: All variables

```
# Make prediction
full_model_prob <- predict(full_model, type = "response", newdata = Ploan_tst)
full_model_prey <- rep(0, nrow(Ploan_tst))
full_model_prey[which(full_model_prob >= 0.5)] <- 1
full_model_cm <- table(Ploan_tst$Ploan_target, full_model_prey)
full_model_cm

# Peformance evaluation
Perf_Table[1,] <- perf_eval(full_model_cm)
Perf_Table
```

✓ type = “response” option for predict( ) function gives the probability of belonging to class 1

✓ Use 0.5 as the cut-off

```
> full_model_cm
   full_model_prey
      0    1
  0 881   15
  1  36   68
```

고려대학교 KOREA UNIVERSITY

DSBA Data Science & Business Analytics

R Exercise: Use All Variables

- Logistic Regression 1: All variables

```
> Perf_Table
                  TPR Precision       TNR Accuracy       BCR F1-Measure
All         0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward     0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Backward    0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Stepwise    0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
GA          0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Ridge       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Lasso       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Elastic Net 0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Forward Selection

- Logistic Regression 2: Forward Selection

```
# Variable selection method 1: Forward selection
tmp_x <- paste(colnames(Ploan_trn)[-12], collapse=" + ")
tmp_xy <- paste("Ploan_target ~ ", tmp_x, collapse = "")
as.formula(tmp_xy)

forward_model <- step(glm(Ploan_target ~ 1, data = Ploan_trn),
                      scope = list(upper = as.formula(tmp_xy),
                      lower = Ploan_target ~ 1), direction="forward", trace = 1)

summary(forward_model)
forward_model_coeff <- as.matrix(forward_model$coefficients, 12, 1)
forward_model_coeff
```

✓ step( ) function: perform forward selection/backward elimination/stepwise selection

- Arg 1: Initial model, forward selection model begins with the model with no input variable
- Arg 2: Range of selected variables
  - Upper: the largest set of selected variables (all variables in this experiment)
  - Lower: the smallest set of selected variables (0 in this experiment)
- Arg 3: direction = "forward" (perform forward selection)

R Exercise: Forward Selection

- Logistic Regression 2: Forward Selection
  - ✓ Mortgage and Online are not selected
    - Age
    - Experience
    - ~~Mortgage~~
    - ~~Online~~

```
Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)          0.101881   0.006029  16.899  < 2e-16 ***
Income               0.149121   0.008154  18.288  < 2e-16 ***
CD.Account           0.083880   0.006887  12.180  < 2e-16 ***
Education            0.068442   0.006366  10.750  < 2e-16 ***
Family               0.039866   0.006121   6.513 1.00e-10 ***
CreditCard          -0.024395   0.006331  -3.853 0.000122 ***
Securities.Account  -0.025492   0.006506  -3.918 9.32e-05 ***
CCAvg                0.019494   0.007808   2.497 0.012642 *  
Experience           0.107328   0.058000   1.850 0.064444 .  
Age                 -0.098853   0.058078  -1.702 0.088950 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

R Exercise: Forward Selection

- Logistic Regression 2: Forward Selection

```
# Make prediction
forward_model_prob <- predict(forward_model, type = "response",
                              newdata = Ploan_tst)
forward_model_prey <- rep(0, nrow(Ploan_tst))
forward_model_prey[which(forward_model_prob >= 0.5)] <- 1
forward_model_cm <- table(Ploan_tst$Ploan_target, forward_model_prey)
forward_model_cm

# Peformance evaluation Perf_Table[2,] <- perf_eval(forward_model_cm)
Perf_Table
```

✓ type = "response" option for predict( ) function gives the probability of belonging to class 1

✓ Use 0.5 as the cut-off

```
> forward_model_cm
   forward_model_prey
      0   1
  0 893   3
  1  57  47
```

R Exercise: Forward Selection

- Logistic Regression 2: Forward Selection

```
> Perf_Table
                   TPR Precision       TNR Accuracy       BCR F1-Measure
All          0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward      0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward     0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Stepwise     0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
GA           0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Ridge        0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Lasso        0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Elastic Net  0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Backward Elimination

- Logistic Regression 3: Backward Elimination

```
# Variable selection method 2: Backward elimination
backward_model <- step(full_model, scope = list(upper = as.formula(tmp_xy),
              lower = Ploan_target ~ 1), direction = "backward",
                        trace = 1)

summary(backward_model)
backward_model_coeff <- as.matrix(backward_model$coefficients, 12, 1)
backward_model_coeff
```

✓ step( ) function: perform forward selection/backward elimination/stepwise selection

- Arg 1: Initial model, forward selection model begins with the model with no input variable
- Arg 2: Range of selected variables
  - Upper: the largest set of selected variables (all variables in this experiment)
  - Lower: the smallest set of selected variables (0 in this experiment)
- Arg 3: direction = "backward" (perform backward elimination)

고려대학교 KOREA UNIVERSITY

DSBA Data Science & Business Analytics

R Exercise: Backward Elimination

- Logistic Regression 3: Backward Elimination
  - ✓ Age, Experiment, Mortgage, and Online are not selected
    - ~~Age~~
    - ~~Experience~~
    - ~~Mortgage~~
    - ~~Online~~

```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)    
(Intercept)        -4.4217     0.2685 -16.467  < 2e-16 ***
Income              2.3871     0.2021  11.814  < 2e-16 ***
Family              0.7319     0.1507   4.858 1.18e-06 ***
CCAvg               0.2482     0.1197   2.073 0.038149 *  
Education           1.3033     0.1661   7.849 4.20e-15 ***
Securities.Account -0.4675     0.1809  -2.585 0.009751 ** 
CD.Account          0.9373     0.1426   6.572 4.95e-11 ***
CreditCard         -0.5889     0.1769  -3.329 0.000871 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

R Exercise: Backward Elimination

- Logistic Regression 3: Backward Elimination

```
# Make prediction
backward_model_prob <- predict(backward_model, type = "response",
                               newdata = Ploan_tst)
backward_model_prey <- rep(0, nrow(Ploan_tst))
backward_model_prey[which(backward_model_prob >= 0.5)] <- 1
backward_model_cm <- table(Ploan_tst$Ploan_target, backward_model_prey)
backward_model_cm

# Peformance evaluation
Perf_Table[3,] <- perf_eval(backward_model_cm)
Perf_Table
```

✓ type = “response” option for predict( ) function gives the probability of belonging to class 1

✓ Use 0.5 as the cut-off

```
> backward_model_cm
   backward_model_prey
      0   1
  0 881  15
  1  37  67
```

R Exercise: Backward Elimination

- Logistic Regression 3: Backward Elimination

```
> Perf_Table
                  TPR Precision       TNR Accuracy       BCR F1-Measure
All         0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward     0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward    0.6442308 0.8170732 0.9832589    0.948 0.7958930  0.7204301
Stepwise    0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
GA          0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Ridge       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Lasso       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Elastic Net 0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Stepwise Selection

- Logistic Regression 4: Stepwise Selection

```
# Variable selection method 3: Stepwise selection
tmp_x <- paste(colnames(Ploan_trn)[-12], collapse=" + ")
tmp_xy <- paste("Ploan_target ~ ", tmp_x, collapse = "")
as.formula(tmp_xy)

stepwise_model <- step(glm(Ploan_target ~ 1, data = Ploan_trn),
                       scope = list(upper = as.formula(tmp_xy),
                       lower = Ploan_target ~ 1), direction="both", trace = 1)
summary(stepwise_model)
stepwise_model_coeff <- as.matrix(stepwise_model$coefficients, 12, 1)
stepwise_model_coeff
```

✓ step( ) function: perform forward selection/backward elimination/stepwise selection

- Arg 1: Initial model, forward selection model begins with the model with no input variable
- Arg 2: Range of selected variables
  - Upper: the largest set of selected variables (all variables in this experiment)
  - Lower: the smallest set of selected variables (0 in this experiment)
- Arg 3: direction = "backward" (perform backward elimination)

R Exercise: Stepwise Selection

- Logistic Regression 4: Stepwise Selection
  - ✓ Mortgage and Online are not selected (same result with the forward selection)
    - Age
    - Experience
    - ~~Mortgage~~
    - ~~Online~~

```
Coefficients:
                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)         0.101881   0.006029  16.899  < 2e-16 ***
Income              0.149121   0.008154  18.288  < 2e-16 ***
CD.Account          0.083880   0.006887  12.180  < 2e-16 ***
Education           0.068442   0.006366  10.750  < 2e-16 ***
Family              0.039866   0.006121   6.513 1.00e-10 ***
CreditCard         -0.024395   0.006331  -3.853 0.000122 ***
Securities.Account -0.025492   0.006506  -3.918 9.32e-05 ***
CCAvg               0.019494   0.007808   2.497 0.012642 *  
Experience          0.107328   0.058000   1.850 0.064444 .  
Age                -0.098853   0.058078  -1.702 0.088950 .  
```

R Exercise: Stepwise Selection

- Logistic Regression 4: Stepwise Selection

```
# Make prediction
stepwise_model_prob <- predict(stepwise_model, type = "response",
                               newdata = Ploan_tst)
stepwise_model_prey <- rep(0, nrow(Ploan_tst))
stepwise_model_prey[which(stepwise_model_prob >= 0.5)] <- 1
stepwise_model_cm <- table(Ploan_tst$Ploan_target, stepwise_model_prey)
stepwise_model_cm

# Peformance evaluation
Perf_Table[4,] <- perf_eval(stepwise_model_cm)
Perf_Table
```

✓ type = "response" option for predict( ) function gives the probability of belonging to class 1

✓ Use 0.5 as the cut-off

```
> stepwise_model_cm
 stepwise_model_prey
     0   1
  0 893   3
  1  57  47
```

R Exercise: Stepwise Selection

- Logistic Regression 4: Stepwise Selection

```
> Perf_Table
                  TPR Precision       TNR Accuracy       BCR F1-Measure
All         0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward     0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward    0.6442308 0.8170732 0.9832589    0.948 0.7958930  0.7204301
Stepwise    0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
GA          0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Ridge       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Lasso       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Elastic Net 0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm

```
# Variable selection method 4: Genetic Algorithm
# Fitness function: F1 for the training dataset
fit_F1 <- function(string){
    sel_var_idx <- which(string == 1)
    # Use variables whose gene value is 1
    sel_x <- x[, sel_var_idx]
    xy <- data.frame(sel_x, y)
    # Training the model
    GA_lr <- glm(y ~ ., family = binomial, data = xy)
    GA_lr_prob <- predict(GA_lr, type = "response", newdata = xy)
    GA_lr_prey <- rep(0, length(y))
    GA_lr_prey[which(GA_lr_prob >= 0.5)] <- 1
    GA_lr_cm <- matrix(0, nrow = 2, ncol = 2)
    GA_lr_cm[1,1] <- length(which(y == 0 & GA_lr_prey == 0))
    GA_lr_cm[1,2] <- length(which(y == 0 & GA_lr_prey == 1))
    GA_lr_cm[2,1] <- length(which(y == 1 & GA_lr_prey == 0))
    GA_lr_cm[2,2] <- length(which(y == 1 & GA_lr_prey == 1))
    GA_perf <- perf_eval(GA_lr_cm)
    return(GA_perf[6])
}
```

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm
  - ✓ fit_F1( ) function
    - Input: chromosome (binary vector whose length is the same as the number of variables)
      - 1: use the corresponding variable in the current model
      - 0: do not use the corresponding variable in the current model
      - Example

1	1	0	0	0	1	0	1	0	0	1

    - Output: fitness function in terms of F1 measure (cut-off = 0.5)

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm

```
x <- as.matrix(Ploan_trn[,-12])
y <- Ploan_trn[,12]

# Variable Selection via Genetic Algorithm
start_time <- proc.time()
GA_F1 <- ga(type = "binary", fitness = fit_F1, nBits = ncol(x),
            names = colnames(x), popSize = 50, pcrossover = 0.5, pmutation = 0.01,
            maxiter = 100, elitism = 2, seed = 123)
end_time <- proc.time()
end_time - start_time
```

✓ ga( ) function: variable selected via genetic algorithm

- Arg 1: type of chromosome, if it is "binary", each gene has either 0 or 1 value
- Arg 2: fitness function
- Arg 3 & 4: number of variables and variable names
- Arg 5 & 6 & 7: number of chromosomes, crossover rate, mutation rate
- Arg 8 & 9: Maximum number of iterations, number of chromosomes to preserve

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm

```
best_var_idx <- which(GA_F1@solution == 1)

# Model training based on the best variable subset
GA_trn_data <- Ploan_trn[,c(best_var_idx, 12)]
GA_tst_data <- Ploan_tst[,c(best_var_idx, 12)]
GA_model <- glm(Ploan_target ~ ., family=binomial, GA_trn_data)

summary(GA_model)
GA_model_coeff <- as.matrix(GA_model$coefficients, 12, 1)
GA_model_coeff
```

✓ best_var_idx: index of best variable subset selected by GA

```
> best_var_idx
[1]  1  3  4  5  6  7  8  9 11
```

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm
  - ✓ Unselected variables
    - Experience
    - Online

```
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)    
(Intercept)         -4.4333     0.2705 -16.391  < 2e-16 ***
Age                  0.1292     0.1353   0.954  0.33985    
Income               2.3956     0.2063  11.611  < 2e-16 ***
Family               0.7508     0.1527   4.918 8.76e-07 ***
CCAvg                0.2572     0.1202   2.140  0.03237 *  
Education            1.3081     0.1670   7.833 4.75e-15 ***
Mortgage             0.0135     0.1003   0.135  0.89298    
Securities.Account  -0.4726     0.1822  -2.595  0.00947 ** 
CD.Account           0.9276     0.1429   6.489 8.63e-11 ***
CreditCard          -0.5733     0.1770  -3.240  0.00120 ** 
```

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm

```r
# Make prediction
GA_model_prob <- predict(GA_model, type = "response", newdata = GA_tst_data)
GA_model_prey <- rep(0, nrow(Ploan_tst))
GA_model_prey[which(GA_model_prob >= 0.5)] <- 1
GA_model_cm <- table(GA_tst_data$Ploan_target, GA_model_prey)
GA_model_cm

# Peformance evaluation
Perf_Table[5,] <- perf_eval(GA_model_cm)
Perf_Table
```

✓ type = "response" option for predict( ) function gives the probability of belonging to class 1

✓ Use 0.5 as the cut-off

```
> GA_model_cm
   GA_model_prey
      0    1
  0 883   13
  1  38   66
```

R Exercise: Genetic Algorithm

- Logistic Regression 5: Genetic Algorithm

```
> Perf_Table
                  TPR Precision       TNR Accuracy       BCR F1-Measure
All         0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward     0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward    0.6442308 0.8170732 0.9832589    0.948 0.7958930  0.7204301
Stepwise    0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
GA          0.6346154 0.8354430 0.9854911    0.949 0.7908273  0.7213115
Ridge       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Lasso       0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Elastic Net 0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Ridge Regression

- Logistic Regression 6: Ridge Regression

```
# Shrinkage method 1: Ridge logistic regression
Ploan_trn_X <- as.matrix(Ploan_trn[,-12])
Ploan_trn_y <- as.factor(Ploan_trn[,12])
Ploan_tst_X <- as.matrix(Ploan_tst[,-12])
Ploan_tst_y <- as.factor(Ploan_tst[,12])

Ridge_model <- glmnet(Ploan_trn_X, Ploan_trn_y, family = "binomial", alpha = 0)
plot(Ridge_model, xvar = "lambda")
```

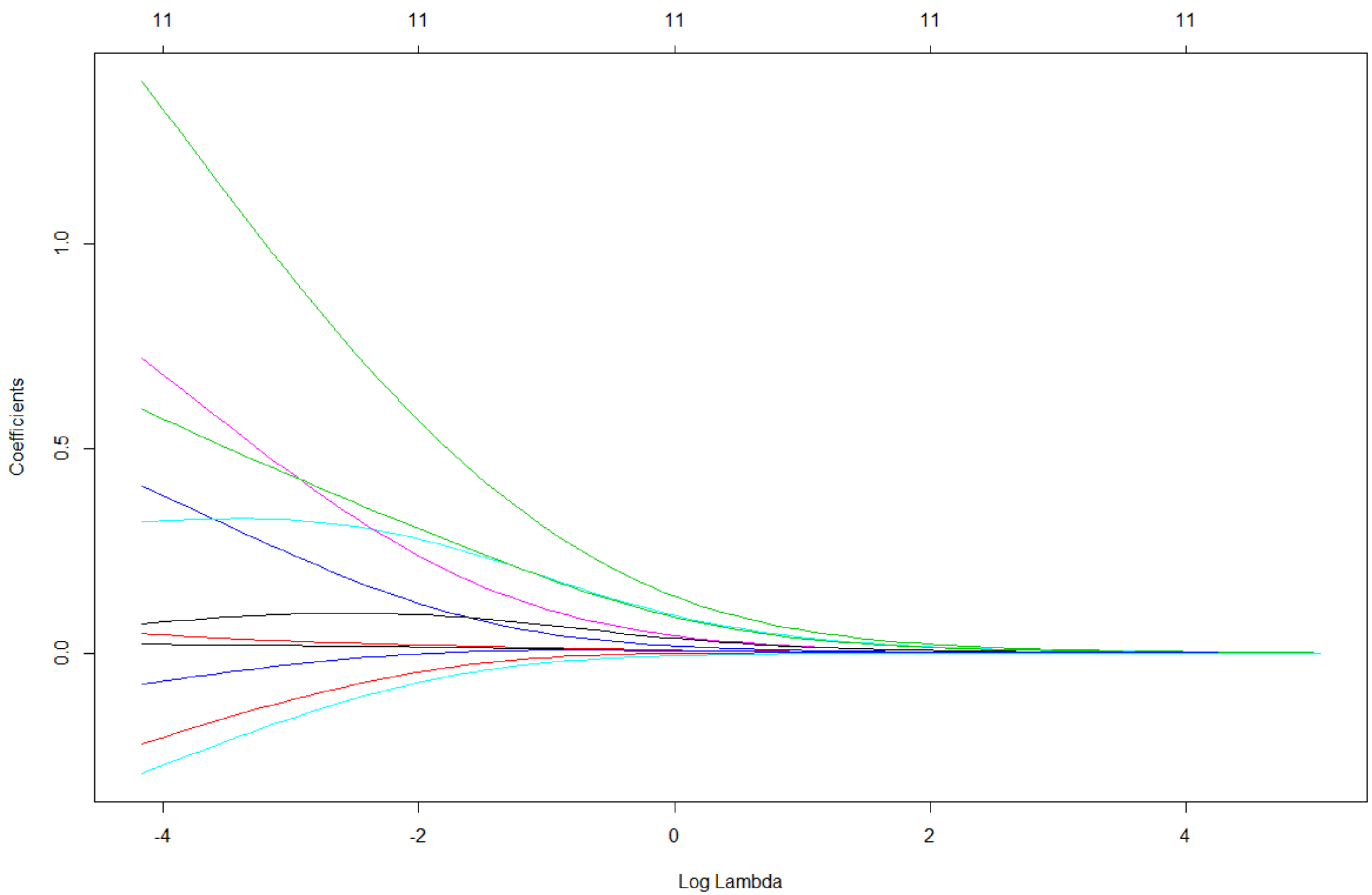
✓ glmnet( ) function: can learn shrinkage methods

- Arg 1: Input variables (matrix form)
- Arg 2: Target variable (factor form)
- Arg 3: family = "binomial" (binary classification = logistic regression)
- Arg 4: Weight for L1 and L2 norm (alpha = 0 → Ridge regression)

$$(1-\alpha) \times \lambda_1 \sum_{j=1}^{d} |\hat{\beta}_j| + \alpha \times \lambda_2 \sum_{j=1}^{d} \hat{\beta}_j^2$$

R Exercise: Ridge Regression

- Logistic Regression 6: Ridge Regression



R Exercise: Ridge Regression

- Logistic Regression 6: Ridge Regression

```
# Find the best lambda based in 5-fold cross validation
CV_Ridge <- cv.glmnet(Ploan_trn_X, Ploan_trn_y, family = "binomial", alpha = 0)
plot(CV_Ridge)
best_lambda <- CV_Ridge$lambda.min
```

✓ cv.glmnet( ): function for 5-fold cross-validation

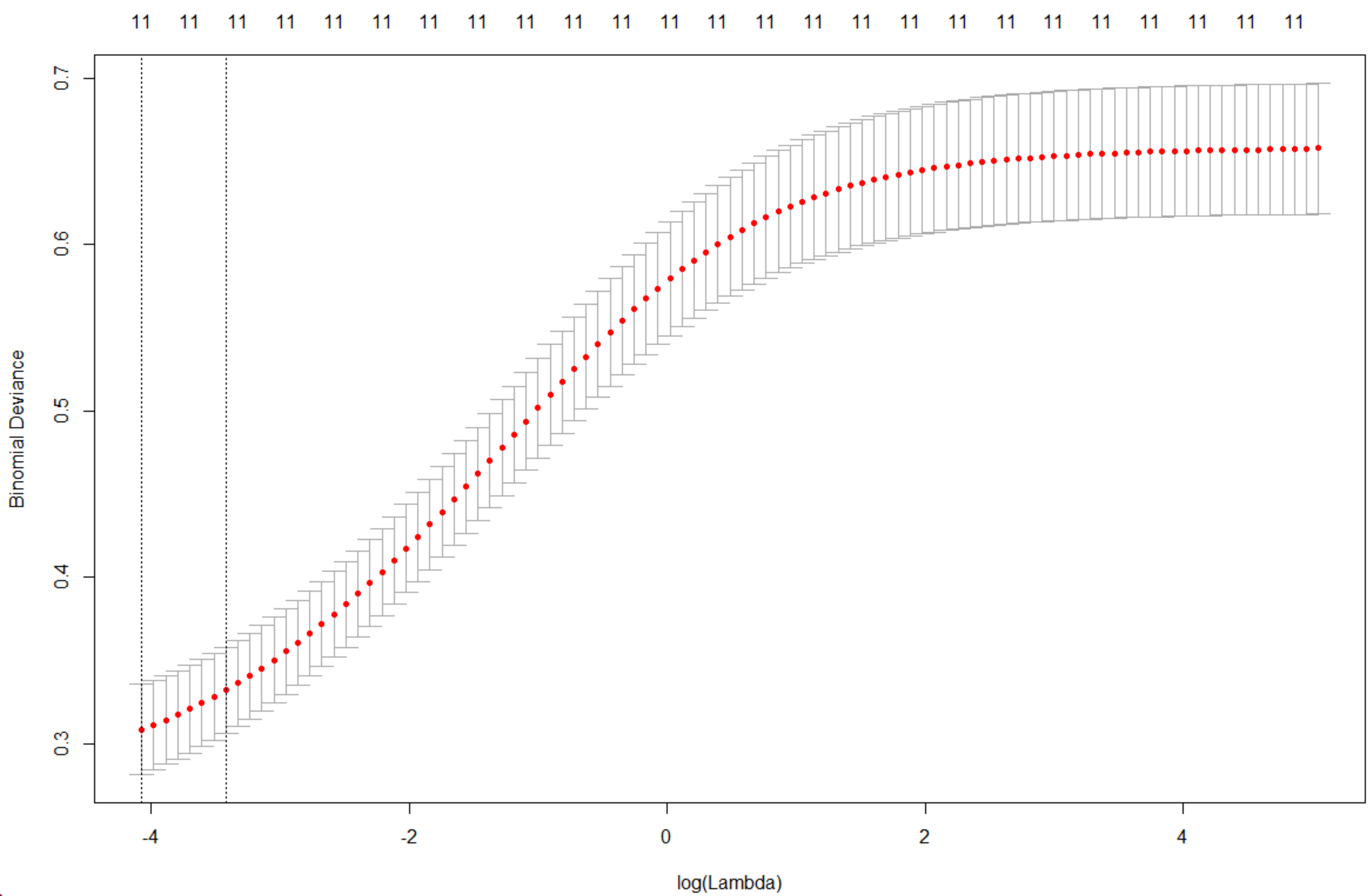
- Arg 1: Input variables (matrix form)
- Arg 2: Target variable (factor form)
- Arg 3: family = "binomial" (binary classification = logistic regression)
- Arg 4: Weight for L1 and L2 norm (alpha = 0 → Ridge regression)

고려대학교 KOREA UNIVERSITY

DSBA Data Science & Business Analytics

R Exercise: Ridge Regression

- Logistic Regression 6: Ridge Regression



R Exercise: Ridge Regression

- Logistic Regression 6: Ridge Regression

```
# Check the coefficients
Ridge_model_coeff <- predict(Ridge_model, s = best_lambda,
                             newx = Ploan_tst_X, type = "coefficient")
Ridge_model_coeff

# Make predictions
Ridge_model_prey <- predict(Ridge_model, s = best_lambda,
                            newx = Ploan_tst_X, type = "class")

Ridge_model_prey <- as.factor(Ridge_model_prey)
Ridge_model_cm <- table(Ploan_tst_y, Ridge_model_prey)
Ridge_model_cm
```

✓ predict( )

- Arg 1: trained model
- Arg 2: Lambda
- Arg 3: Input variables of test dataset
- Arg 4: Type of the output (estimated coefficients or predicted class)

R Exercise: Ridge Regression

- Logistic Regression 6: Ridge Regression

```
> Perf_Table
                   TPR Precision       TNR Accuracy       BCR F1-Measure
All          0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward      0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward     0.6442308 0.8170732 0.9832589    0.948 0.7958930  0.7204301
Stepwise     0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
GA           0.6346154 0.8354430 0.9854911    0.949 0.7908273  0.7213115
Ridge        0.5384615 0.8888889 0.9921875    0.945 0.7309274  0.6706587
Lasso        0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
Elastic Net 0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Lasso Regression

- Logistic Regression 7: Lasso Regression

```
# Shrinkage method 2: Lasso regression
Lasso_model <- glmnet(Ploan_trn_X, Ploan_trn_y, family = "binomial", alpha = 1)
plot(Lasso_model, xvar = "lambda")

# Find the best lambda based in 5-fold cross validation
CV_Lasso <- cv.glmnet(Ploan_trn_X, Ploan_trn_y, family = "binomial", alpha = 1)
plot(CV_Lasso)
best_lambda <- CV_Lasso$lambda.min
```

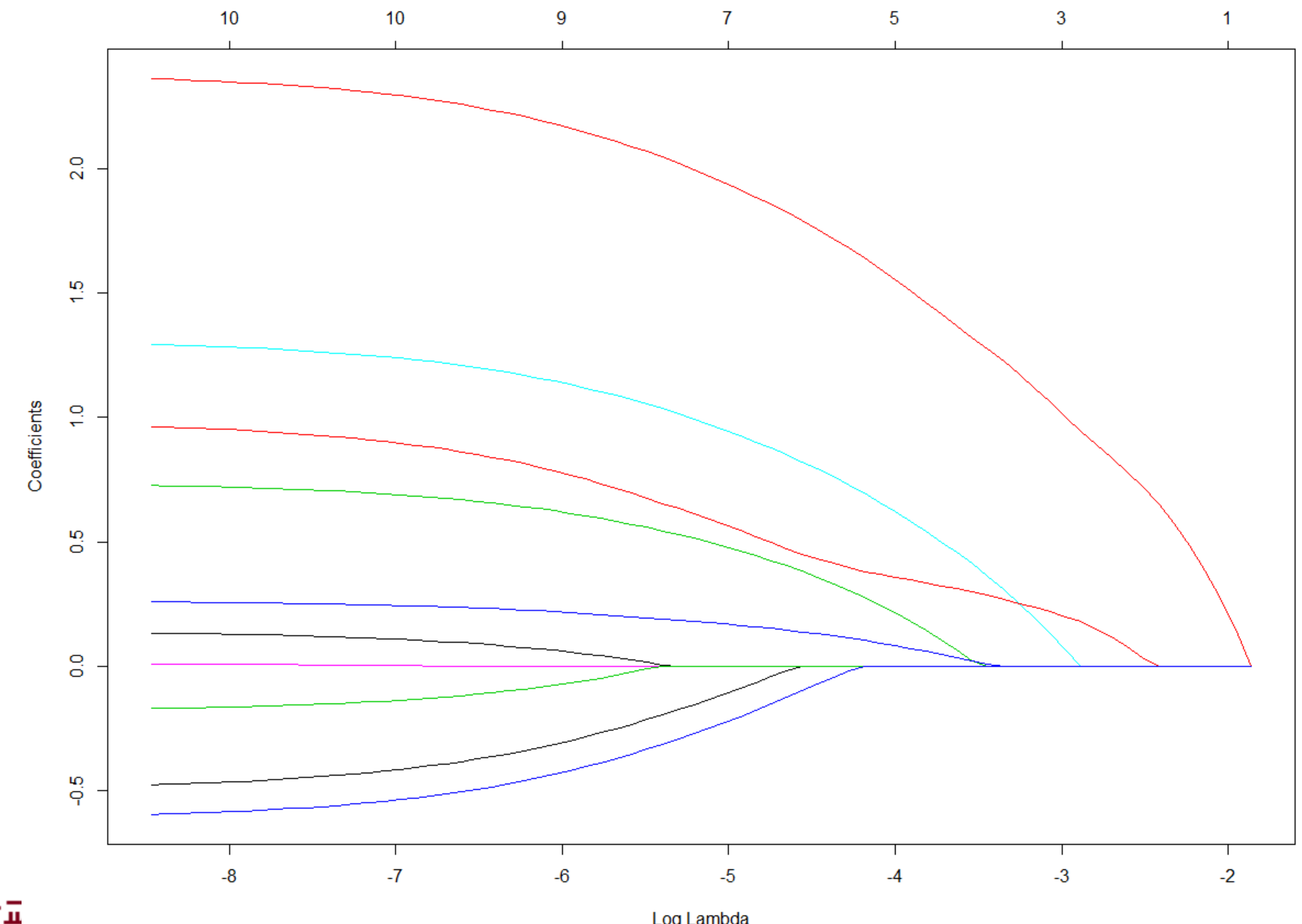
✓ glmnet( ) function: can learn shrinkage methods

- Arg 1: Input variables (matrix form)
- Arg 2: Target variable (factor form)
- Arg 3: family = "binomial" (binary classification = logistic regression)
- Arg 4: Weight for L1 and L2 norm (alpha = 1 → Lasso regression)

$$(1-\alpha) \times \lambda_1 \sum_{j=1}^{d} |\hat{\beta}_j| + \alpha \times \lambda_2 \sum_{j=1}^{d} \hat{\beta}_j^2$$

R Exercise: Lasso Regression

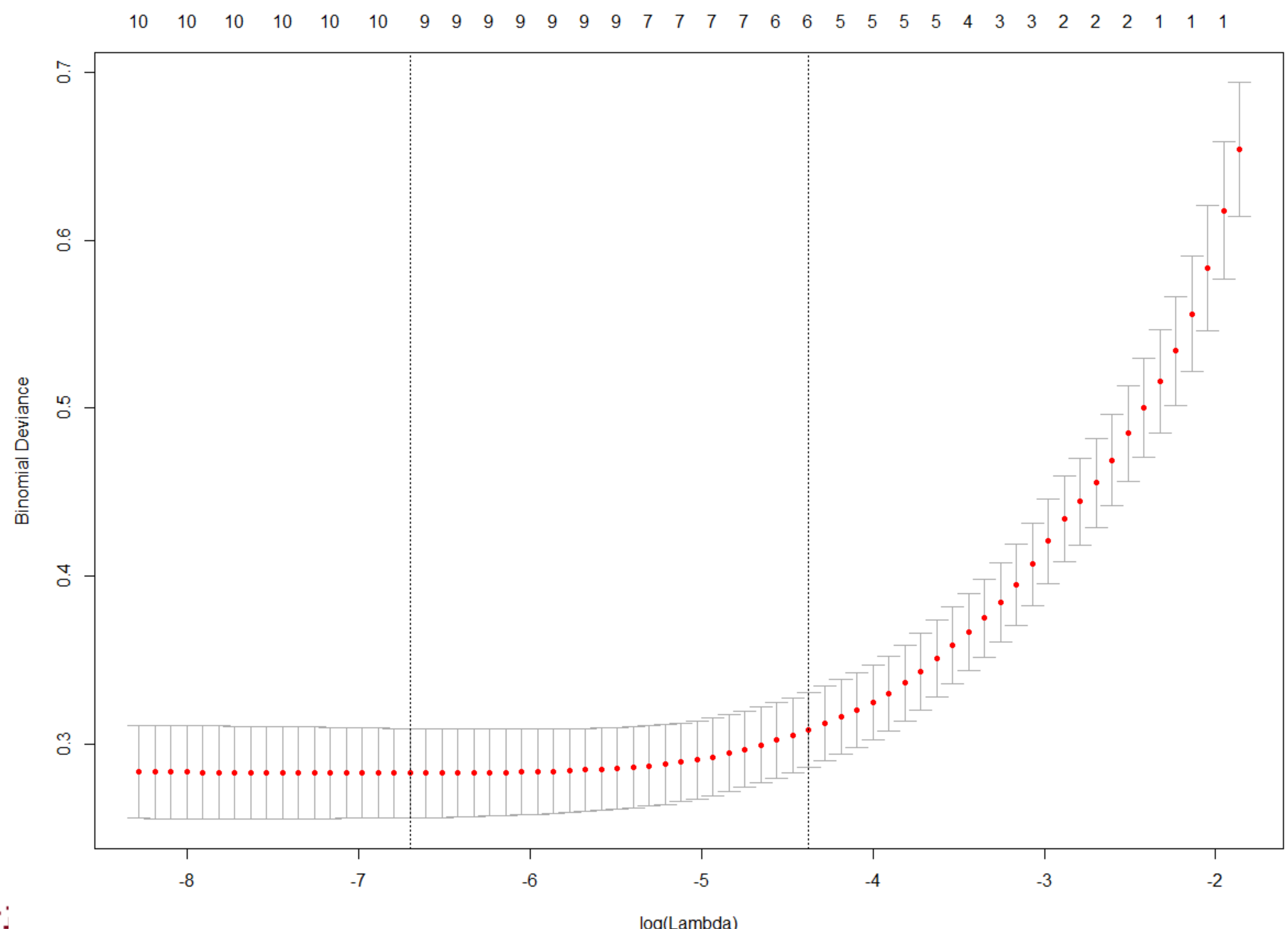
- Logistic Regression 7: Lasso Regression



고려대학교 KOREA UNIVERSITY
DSBA Data Science & Business Analytics

R Exercise: Lasso Regression

- Logistic Regression 7: Lasso Regression



R Exercise: Lasso Regression

- Logistic Regression 7: Lasso Regression

```
# Check the coefficients
Lasso_model_coeff <- predict(Lasso_model, s = best_lambda,
                             newx = Ploan_tst_X, type = "coefficient")
Lasso_model_coeff

# Make predictions
Lasso_model_prey <- predict(Lasso_model, s = best_lambda,
                            newx = Ploan_tst_X, type = "class")
Lasso_model_prey <- as.factor(Lasso_model_prey)
Lasso_model_cm <- table(Ploan_tst_y, Lasso_model_prey)
Lasso_model_cm
```

✓ predict( )

- Arg 1: Trained model
- Arg 2: Lambda
- Arg 3: Input variable of test dataset
- Arg 4: Output type (regression coefficients or predicted class)

R Exercise: Lasso Regression

- Logistic Regression 7: Lasso Regression

```
> Perf_Table
                  TPR Precision       TNR Accuracy       BCR F1-Measure
All         0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward     0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward    0.6442308 0.8170732 0.9832589    0.948 0.7958930  0.7204301
Stepwise    0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
GA          0.6346154 0.8354430 0.9854911    0.949 0.7908273  0.7213115
Ridge       0.5384615 0.8888889 0.9921875    0.945 0.7309274  0.6706587
Lasso       0.6250000 0.8227848 0.9843750    0.947 0.7843688  0.7103825
Elastic Net 0.0000000 0.0000000 0.0000000    0.000 0.0000000  0.0000000
```

R Exercise: Elastic Net

- Logistic Regression 8: Elastic Net

```
# Shrinkage method 3: Elastic net regression
Elastic_model <- glmnet(Ploan_trn_X, Ploan_trn_y, family = "binomial", alpha = 0.5)
plot(Elastic_model, xvar = "lambda")

# Find the best lambda based in 5-fold cross validation
CV_Elastic <- cv.glmnet(Ploan_trn_X, Ploan_trn_y, family = "binomial", alpha = 0.5)
plot(CV_Elastic)
best_lambda <- CV_Elastic$lambda.min
```

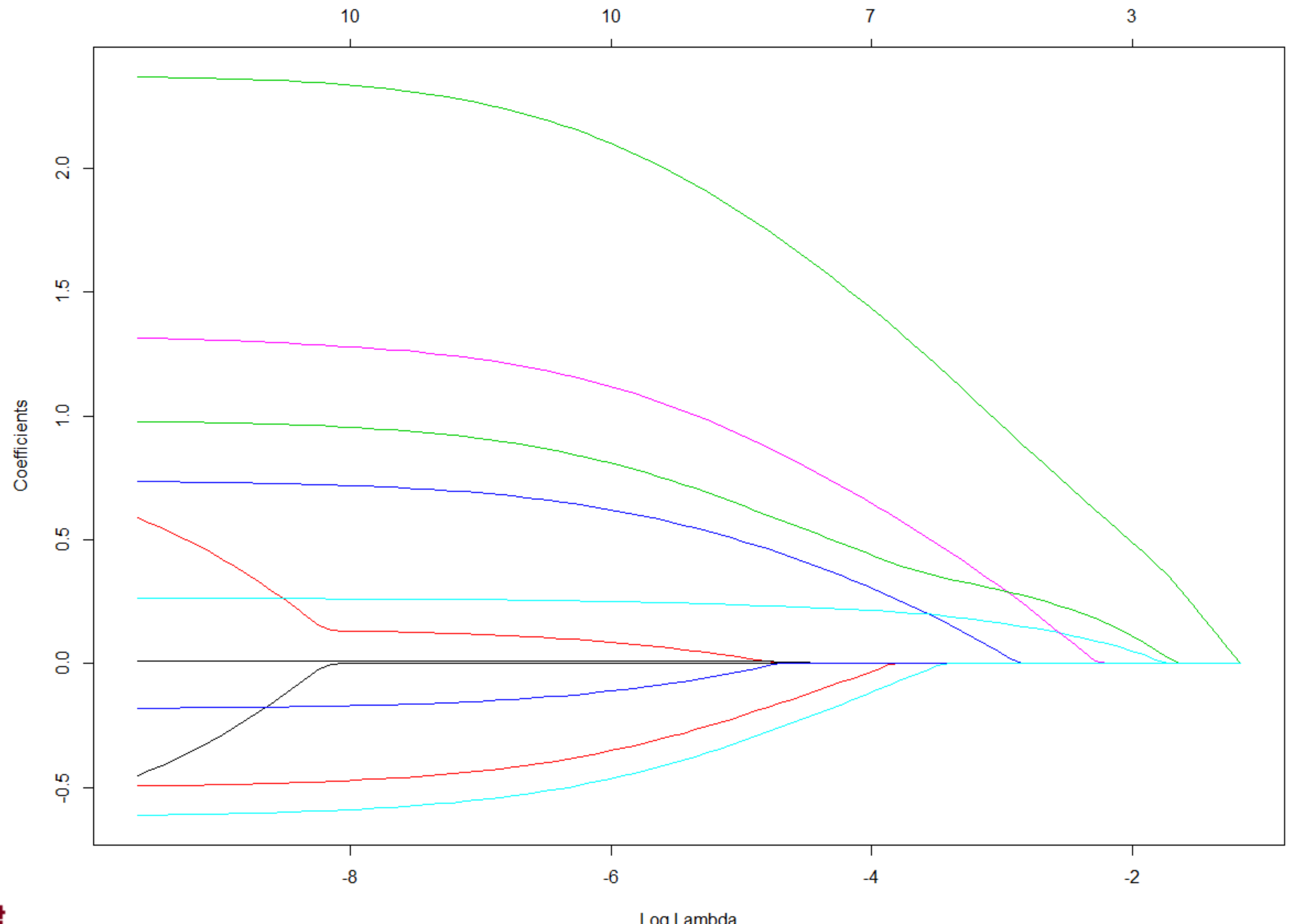
✓ glmnet( ) function: can learn shrinkage methods

- Arg 1: Input variables (matrix form)
- Arg 2: Target variable (factor form)
- Arg 3: family = "binomial" (binary classification = logistic regression)
- Arg 4: Weight for L1 and L2 norm (alpha = 0.5 → Elastic Net)

$$(1-\alpha) \times \lambda_1 \sum_{j=1}^{d} |\hat{\beta}_j| + \alpha \times \lambda_2 \sum_{j=1}^{d} \hat{\beta}_j^2$$

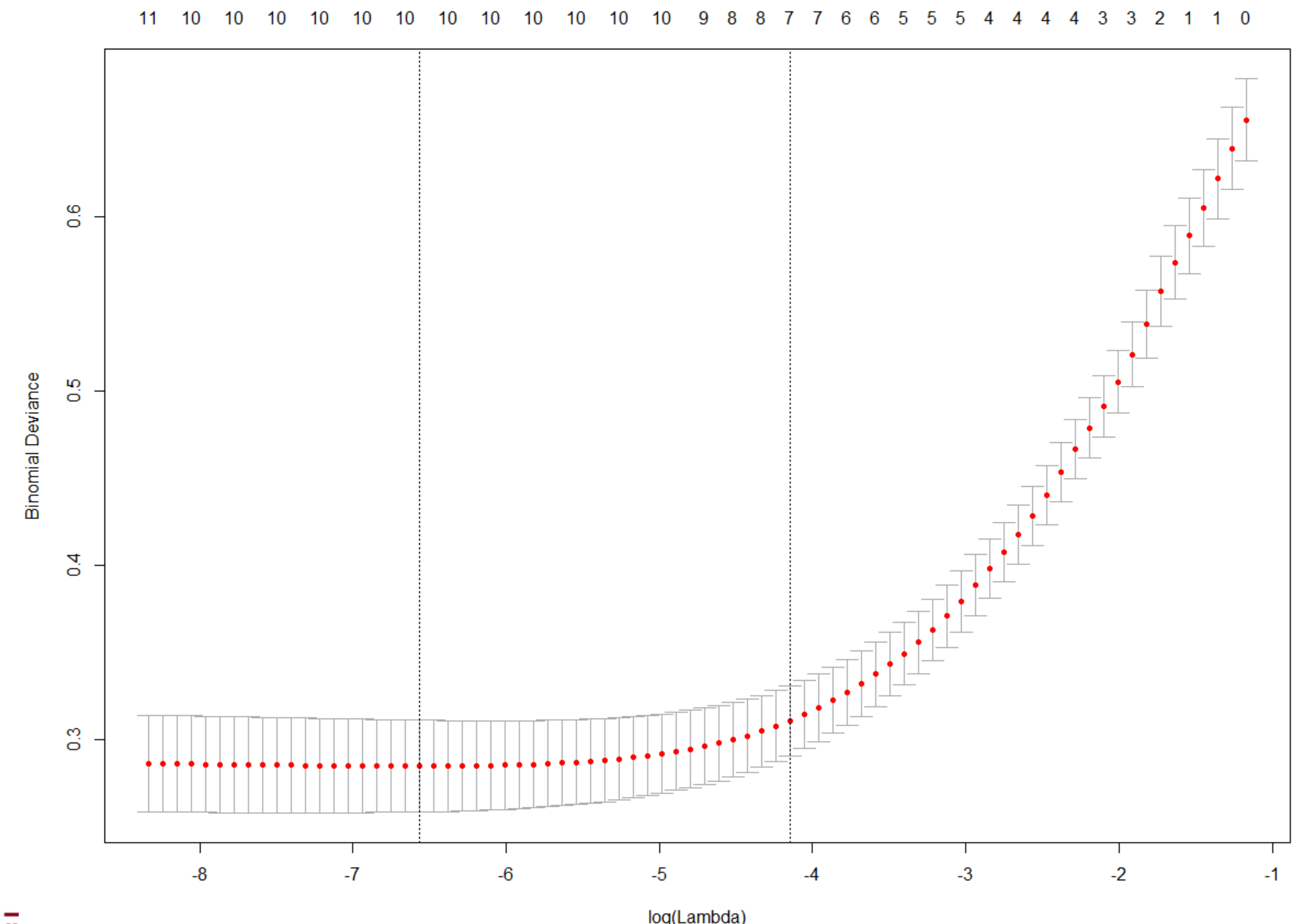
R Exercise: Elastic Net

- Logistic Regression 8: Elastic Net



R Exercise: Elastic Net

- Logistic Regression 8: Elastic Net



R Exercise: Elastic Net

- Logistic Regression 8: Elastic Net

```
# Check the coefficients
Elastic_model_coeff <- predict(Elastic_model, s = best_lambda,
                               newx = Ploan_tst_X, type = "coefficient")
Elastic_model_coeff

# Make predictions
Elastic_model_prey <- predict(Elastic_model, s = best_lambda,
                              newx = Ploan_tst_X, type = "class")
Elastic_model_prey <- as.factor(Elastic_model_prey)
Elastic_model_cm <- table(Ploan_tst_y, Elastic_model_prey)
Elastic_model_cm
```

✓ predict( )

- Arg 1: Trained model
- Arg 2: Lambda
- Arg 3: Input variable of test dataset
- Arg 4: Output type (regression coefficients or predicted class)

R Exercise: Elastic Net

- Logistic Regression 8: Elastic Net

```
> Perf_Table
                  TPR Precision       TNR Accuracy       BCR F1-Measure
All         0.6538462 0.8192771 0.9832589    0.949 0.8018105  0.7272727
Forward     0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
Backward    0.6442308 0.8170732 0.9832589    0.948 0.7958930  0.7204301
Stepwise    0.4519231 0.9400000 0.9966518    0.940 0.6711259  0.6103896
GA          0.6346154 0.8354430 0.9854911    0.949 0.7908273  0.7213115
Ridge       0.5384615 0.8888889 0.9921875    0.945 0.7309274  0.6706587
Lasso       0.6250000 0.8227848 0.9843750    0.947 0.7843688  0.7103825
Elastic Net 0.6250000 0.8227848 0.9843750    0.947 0.7843688  0.7103825
```

R Exercise: Summary

- Logistic Regression Coefficients

```
> full_model_coeff
                         [,1]
(Intercept)      -4.445483193
Age              -0.776030543
Experience        0.910983523
Income            2.374700527
Family            0.739703391
CCAvg             0.264244401
Education         1.328860208
Mortgage          0.009294095
Securities.Account -0.501459121
CD.Account        0.982081783
Online           -0.182069399
CreditCard       -0.617373701
```

```
> forward_model_coeff
                          [,1]
(Intercept)         0.10188147
Income              0.14912124
CD.Account          0.08388042
Education           0.06844227
Family              0.03986637
CreditCard         -0.02439480
Securities.Account -0.02549191
CCAvg               0.01949386
Experience          0.10732764
Age                -0.09885277
```

```
> backward_model_coeff
                         [,1]
(Intercept)        -4.4217370
Income              2.3871278
Family              0.7319393
CCAvg               0.2481614
Education           1.3033304
Securities.Account -0.4674934
CD.Account          0.9372616
CreditCard         -0.5889305
```

```
> stepwise_model_coeff
                          [,1]
(Intercept)         0.10188147
Income              0.14912124
CD.Account          0.08388042
Education           0.06844227
Family              0.03986637
CreditCard         -0.02439480
Securities.Account -0.02549191
CCAvg               0.01949386
Experience          0.10732764
Age                -0.09885277
```

```
> GA_model_coeff
                        [,1]
(Intercept)      -4.43332725
Age               0.12916530
Income            2.39554937
Family            0.75075280
CCAvg             0.25721313
Education         1.30812212
Mortgage          0.01350098
Securities.Account -0.47264049
CD.Account        0.92755663
CreditCard       -0.57332418
```

```
> Ridge_model_coeff
12 x 1 sparse Matrix of class
                             1
(Intercept)        -3.25969793
Age                 0.02036741
Experience          0.04528453
Income              1.35739246
Family              0.39426207
CCAvg               0.32094559
Education           0.69815256
Mortgage            0.07387779
Securities.Account -0.21344808
CD.Account          0.58220572
Online             -0.07193090
CreditCard         -0.28325806
```

```
> Lasso_model_coeff
12 x 1 sparse Matrix of class "dg
                               1
(Intercept)        -4.2408732055
Age                 .
Experience          0.0986384046
Income              2.2682994528
Family              0.6748085859
CCAvg               0.2384203451
Education           1.2171062156
Mortgage            0.0006765386
Securities.Account -0.3918108753
CD.Account          0.8714276903
Online             -0.1234223620
CreditCard         -0.5131301095
```

```
> Elastic_model_coeff
12 x 1 sparse Matrix of class
                             1
(Intercept)        -4.21928953
Age                 .
Experience          0.11110504
Income              2.22361220
Family              0.67302703
CCAvg               0.25676164
Education           1.20324138
Mortgage            0.01181456
Securities.Account -0.41388674
CD.Account          0.88543356
Online             -0.14219408
CreditCard         -0.52966366
```


ANY
questions ?