CRM and Machine Learning

Giovanni Compian

Contents

```
library(bit64)
library(data.table)
library(glmnet)
library(ggplot2)
library(corrplot)
library(knitr)
library(dplyr)
```

1 Data

The data that we use is a development sample that includes records on 250,000 randomly selected customers from the data base of a company that sells kitchenware, housewares, and specialty food products. The data include a high-dimensional set of more than 150 customer features. The random sample represents only a small percentage of the whole data base of the company.

Customers are identified by a unique `customer_id`. The targeting status of a customers as of October 10, 2017 is indicated by `mailing_indicator`. The total dollar spend (online, phone and mail order, in-store) in the subsequent 15 week period is captured by `outcome_spend`. All other variables are customer summary data based on the whole transaction history of each customer. These summary data were retrieved one week before the catalog mailing.

The customer features are largely self-explanatory. For example, `orders_online_1yr` indicates the number of orders placed during the last year. Variables such as `spend_m_1yr` indicate spending in a specific department, labeled `m`. Due to privacy reasons, what this department exactly is cannot be revealed. Similarly, no details on specific product types (e.g. `clicks_product_type_104`) can be disclosed. Also, to preserve confidentiality, some variables had to be scaled. Hence, you may see an order count such as 2.4, even though originally orders can only be 0, 1, 2, ... Please note that the scaling has no impact on the predictive power of the statistical models that you estimate. Furthermore, the restricted interpretability of some of the variables has no bearing on our analysis. Ultimately, we use the features to predict spending levels or incremental spending levels, but we do not interpret these features as causal. In particular, we cannot *manipulate* variables such as `orders_online_1yr`, which explains why the corresponding estimates have no causal interpretation.

Load the `crm_DT` data.table.

```
data_folder    = "Data"
path = "Customer-Development-2017.RData"
load(paste0(data_folder, "/", path))
```

I recommend renaming the `mailing_indicator` to make it clear that this variable represents a targeting *treatment*, W_i.

```
setnames(crm_DT, "mailing_indicator", "W")
```

```
# Check Available columns
names(crm_DT)
```

We split the sample into a 50% training and 50% validation sample. To ensure that we all have the same training and validation sample, use the following seed:

```
set.seed(1999)
# Indicate which rows are training and which are testing
crm_DT[, training_sample := rbinom(nrow(crm_DT), 1, 0.5)]
```

2 Data inspection [15 points]

Summarize and describe some key aspects of the `outcome_spend` variable. In particular, what is the purchase incidence (i.e., what fraction of customers make a purchase), what is the distribution of dollar spending, and what is the conditional distribution of dollar spending given that a customer made a purchase?

```
# Summarize Key Aspects of the `outcome_spend`
# 1. Purchase incidence
# - % of customers make a purchase
# - Proportion of cust that has > 0 outcome spend
# `outcome_spend` > 0 creates a logical vector where each entry is TRUE, else FALSE
purchase_incidence <- mean(crm_DT$outcome_spend > 0)
print(purchase_incidence)
```

```
# 2. Distribution of dollar spending
# - how much customers are spending overall, including both customers who made purchases and those who
summary(crm_DT$outcome_spend)  # Provides summary statistics
hist(crm_DT$outcome_spend,
     main = "Distribution of Dollar Spending",  # Title of the plot
     xlab = "Dollar Spending",                  # Label for x-axis
     ylab = "Frequency",                        # Label for y-axis
     col = "lightblue",                         # Fill color
     breaks = 30)                               # Number of # Boxplot of outcome_spend
ggplot(crm_DT, aes(y = outcome_spend)) +
  geom_boxplot(
    fill = "lightgreen",
    color = "black",
    outlier.color = "red"
  ) +
  labs(
    title = "Boxplot of Non-Zero Dollar Spending",
    y = "Dollar Spending"
  ) +
  theme_minimal()
```

```
# 3. Conditional distribution of dollar spending
# - Focus on people who made the purchase outcome_spend > 0
spend_given_purchase <- crm_DT[outcome_spend > 0]
summary(spend_given_purchase$outcome_spend)  # Summary statistics for non-zero spending
hist(spend_given_purchase$outcome_spend,
     main = "Conditional Distribution of Dollar Spending",  # Title of the plot
     xlab = "Dollar Spending",                  # Label for x-axis
     ylab = "Frequency",                        # Label for y-axis
     col = "lightblue",                         # Fill color
     breaks = 30)       # Histogram for conditional spending
ggplot(spend_given_purchase, aes(y = outcome_spend)) +
  geom_boxplot(
    fill = "lightgreen",
    color = "black",
    outlier.color = "red"
  ) +
  labs(
    title = "Boxplot of Non-Zero Dollar Spending",
    y = "Dollar Spending"
  ) +
```

```
  theme_minimal()

outlier_count <- sum(crm_DT$outcome_spend > 153.85)
outlier_count
```

Analysis - Only 6.2% of customers made purchase - Out of those purchased, the distribution is average $128.55 per person, and the majority of people spend less than the mean. However, there are substantial amount of outlier. 3861 outlier customers out of 15520 customers, who purchased from $153 to $1462 per person.

3 Data pre-processing

Data sets with a large number of inputs (features) often contain highly correlated variables. The presence of such variables is not necessarily a problem if we employ an estimation method that uses regularization. However, if two variables are almost perfectly correlated then one of them captures virtually all the information contained in both variables. Also, OLS (or logistic regression) without regularization will be unfeasible with perfectly or near perfectly correlated inputs. Hence, it is helpful to eliminate some highly correlated variables from the data set.

Here is a helpful method to visualize the degree of correlation among all inputs in the data. Install the package `corrplot`. Then calculate a matrix of correlation coefficients among all inputs:

```
cor_matrix = cor(crm_DT[, !c("customer_id", "W", "outcome_spend"),
                         with = FALSE])
```

Now use `corrplot` to create a pdf file that visualizes the correlation among all variables in two separate graphs. There is a huge amount of information in each graph, hence I recommend zooming in! Please see the `corrplot` documentation for a description of all the options.

```
pdf("Correlation-Matrix.pdf", height = 16, width = 16)
corrplot(cor_matrix, method = "color",
         type = "lower", diag = FALSE,
         tl.cex = 0.4, tl.col = "gray10")

corrplot(cor_matrix, method = "number", number.cex = 0.25, addgrid.col = NA,
         type = "lower", diag = FALSE,
         tl.cex = 0.4, tl.col = "gray10")
dev.off()
```

Create a data table that contains the correlations for all variable pairs:

```
cor_matrix[upper.tri(cor_matrix, diag = TRUE)] = NA

# Converts the lower triangular part of the correlation matrix into a long-format data table,
# where each row represents a correlation between two variables.
cor_DT = data.table(row = rep(rownames(cor_matrix), ncol(cor_matrix)),
                    col = rep(colnames(cor_matrix), each = ncol(cor_matrix)),
                    cor = as.vector(cor_matrix))
cor_DT = cor_DT[is.na(cor) == FALSE]
```

In the first statement above, we set the correlations in the upper triangle and on the diagonal of the correlation matrix to `NA`. The correlations on the diagonal are 1.0 and the correlations in the upper triangle are identical to the correlations in the lower triangle. Hence, we do not need to summarize these correlations.

Then we create a new data table, `cor_DT`, that includes all pairs of features and the respective correlation coefficient. Make sure you understand how this table is computed in the four lines of code above!

Now find all correlations larger than 0.95 in absolute value. Inspect these correlations, and then eliminate one of the virtually redundant variables in each highly correlated pair from the data set (to ensure that we end up with the same data, eliminate the redundant variables in the `row` column).

```
large_cor_DT = cor_DT[abs(cor) > 0.95]
kable(large_cor_DT, digits = 4)
```

```
# Remove the large_cor_DT$row from crm_DT
crm_DT = crm_DT[, !large_cor_DT$row, with = FALSE]
```

Note that this last step eliminates from `crm_DT` all variables listed in `large_cor_DT$row`.

4 Predictive model estimation [25 points]

Use the training sample to estimate the conditional expectation of dollar spending, based on all available customer information (features). In particular, following the common approach in the industry that we have discussed in class, estimate the model only for customers who were targeted, such that $W_i = 1$. Hence, we estimate a model that predicts expected dollar spending, conditional on all customer features and conditional on being targeted. This is the same approach we have followed, e.g., in the JCPenney example in class.

Estimate and compare the following models:

1. OLS
2. LASSO
3. Elastic net

```
# Create training and testing database
training_data <- crm_DT[training_sample==1]
testing_data <- crm_DT[training_sample==0]
dim(training_data)
dim(testing_data)
```

```
# Conditional Expectation
# Expected value (mean) of a variable Y given some conditions ie. Targered (W)
# E[Y | features, W=1]
# STEPS:
# 1. Filter the training sample for W=1
# 2. Estimate a Model to Predict Spending
# 3. Follow the "Common Approach" Discussed in Class
# 4. Output Expected Dollar Spending

# 1. Filter the training sample for W=1
dim(training_data)
training_data_targeted <- training_data[training_data$W == 1, ]
testing_data_targeted <- testing_data[testing_data$W == 1, ]
fit_OLS = lm(outcome_spend ~ ., data = training_data_targeted)
summary_OLS = summary(fit_OLS)
summary_OLS

# Interpretation : R is quite high and R^2 is low 0.1129
# suggesting that model is not capture much variation in y.
# Indicating true relationship between predictors and y is weak.
```

```
# 2. Estimate a Model to Predict Spending
# Output in vector
pred_y_OLS = predict(fit_OLS, newdata = testing_data_targeted)
mse_OLS = mean((testing_data$outcome_spend - pred_y_OLS)^2)

results = data.table(
  input = rownames(summary_OLS$coefficients),    # Extract row names (input variable names)
  est_OLS = summary_OLS$coefficients[, 1],      # Extract coefficient estimates
  p_OLS = summary_OLS$coefficients[, 4]         # Extract p-values
)


kable(results, digits=3)
```

```r
# 3. Follow the "Common Approach" Discussed in Class


# TODO : Evaluate the variables and discard those that are not influencing predictions

# Step 1: Identify columns with p > 0.05
insignificant_vars <- results[p_OLS > 0.05]$input  # Variables to drop

# Step 2: Drop these columns from the datasets
training_data_reduced <- training_data_targeted[, !(colnames(training_data_targeted) %in% insignificant
testing_data_reduced <- testing_data_targeted[, !(colnames(testing_data_targeted) %in% insignificant_va

# Step 3: Refit the model using the reduced dataset
fit_OLS_reduced <- lm(outcome_spend ~ ., data = training_data_reduced)

# Step 4: Predict on reduced dataset
pred_y_OLS_reduced <- predict(fit_OLS_reduced, newdata = testing_data_reduced)

# Step 5: Calculate Mean Squared Error for the reduced model
mse_OLS_reduced <- mean((testing_data_reduced$outcome_spend - pred_y_OLS_reduced)^2)

# Compare performance
cat("MSE (Full Model):", mse_OLS, "\n")
cat("MSE (Reduced Model):", mse_OLS_reduced, "\n")
```

```r
# TODO : Add the other models result for comparison

# LASSO

library(glmnet)

y_train <- training_data_targeted$outcome_spend
X_train <- model.matrix(outcome_spend ~ 0 + ., data = training_data_targeted)

# Set seed for reproducibility
set.seed(37105)

# Create custom folds for cross-validation
N_obs_training <- nrow(training_data_targeted)
folds <- sample(1:10, N_obs_training, replace = TRUE)

# Fit LASSO with custom folds
cv_lasso <- cv.glmnet(X_train, y_train, alpha = 1, foldid = folds)

# Identify the best lambda
best_lambda <- cv_lasso$lambda.min
cat("Best Lambda:", best_lambda, "\n")

# Refit the model with the best lambda
lasso_model <- glmnet(X_train, y_train, alpha = 1, lambda = best_lambda)

# Plot cross-validation results
plot(cv_lasso)
```

```r
# Display coefficients
length(coef(cv_lasso, s = "lambda.min")[, 1])
nrow(results)
coef_vals <- rep(coef(cv_lasso, s = "lambda.min")[, 1], length.out = nrow(results))
results[, est_LASSO := coef_vals]
# Verify matching dimensions
if (length(coef_vals) == nrow(results)) {
    results[, est_LASSO := coef_vals]
} else {
    stop("Dimensions do not match. Please check the data alignment.")
}
#results[, est_LASSO := coef(cv_lasso, s = "lambda.min")[,1]]
#coef(cv_lasso, s = "lambda.min")
```

```r
# Predict and calculate MSE
y_test <- testing_data_targeted$outcome_spend
X_test <- model.matrix(outcome_spend ~ 0 + ., data = testing_data_targeted)

# Predict using the LASSO model
predictions_lasso <- predict(cv_lasso, newx = X_test, s = "lambda.min")

# Calculate Mean Squared Error (MSE) on the test dataset
mse_lasso <- mean((y_test - predictions_lasso)^2)
cat("LASSO Mean Squared Error on Test Set:", mse_lasso, "\n")
```

```r
# Elastic net

library(data.table)

# Step 1: Prepare the data
y <- training_data_targeted$outcome_spend  # Replace 'outcome_spend' with your actual target variable n
X <- model.matrix(outcome_spend ~ 0 + ., data = training_data_targeted)  # Design matrix (exclude inter

# Step 2: Set a fixed seed and create custom folds
set.seed(37105)
N_obs_training <- nrow(training_data_targeted)
folds <- sample(1:10, N_obs_training, replace = TRUE)

# Step 3: Set up a coarser grid for alpha values
alphas <- seq(0, 1, by = 0.05)  # Range of alpha values with a step of 0.05
mse_results <- data.table(alpha = alphas, mse = rep(NA, length(alphas)))

# Step 4: Cross-validate for each alpha value using custom folds
for (i in 1:length(alphas)) {
  alpha_value <- alphas[i]
  cv_model <- cv.glmnet(X, y, alpha = alpha_value, foldid = folds)
  mse_results[i, mse := min(cv_model$cvm)]  # Store the minimum MSE for each alpha
}

# Step 5: Identify the best alpha and corresponding lambda
best_alpha <- mse_results[which.min(mse), alpha]
cat("Best Alpha:", best_alpha, "\n")

cv_best <- cv.glmnet(X, y, alpha = best_alpha, foldid = folds)
```

```r
best_lambda <- cv_best$lambda.min
cat("Best Lambda:", best_lambda, "\n")

# Step 6: Refit the Elastic Net model with the best alpha and lambda
elastic_net_model <- glmnet(X, y, alpha = best_alpha, lambda = best_lambda)
```

```r
# Step 7: Display coefficients
length(coef(elastic_net_model, alpha = best_alpha, lambda = best_lambda)[, 1])
nrow(results)
coef_vals <- rep(coef(elastic_net_model, alpha = best_alpha, lambda = best_lambda)[, 1], length.out = n
results[, est_elastic := coef_vals]

# Step 8: Evaluate on the testing data
y_test <- testing_data_targeted$outcome_spend
X_test <- model.matrix(outcome_spend ~ 0 + ., data = testing_data_targeted)

# Predict and calculate MSE
predictions_elastic <- predict(elastic_net_model, newx = X_test)
mse_elastic <- mean((y_test - predictions_elastic)^2)
cat("Mean Squared Error on Test Set:", mse_elastic, "\n")
```

```r
# 4. Compare the models with results coeeficients (see class example)
cat(mse_OLS, mse_lasso, mse_elastic)
kable(results, digits=3)
```

Compare the estimated coefficients for OLS, the LASSO, and the elastic net. How "sparse" is the prediction problem, i.e. how many inputs are selected by the LASSO and the elastic net?

5 Model validation [30 points]

Take the validation sample and select only those customers who were targeted, i.e. $W_i = 1$. Using this sample, compare the observed and predicted sales outcomes.

First, compare the mean-squared error (MSE) based on the predictions of the three estimation methods.

```
cat(mse_OLS, mse_lasso, mse_elastic)
```

Second, create lift tables and charts (use 20 scores/groups), plot the lifts, and compare the lift tables. I recommend **not** normalizing the lifts by the average spending in the sample, so that we can directly assess the magnitude of predicted mean spending.

```
# Add predictions to testing data
lift_data <- copy(testing_data_targeted)  # Only targeted customers
lift_data[, predicted_OLS := pred_y_OLS]
lift_data[, predicted_lasso := predictions_lasso]
lift_data[, predicted_elastic := predictions_elastic]

# Function to create lift table
create_lift_table <- function(data, predicted_col, actual_col, n_groups = 20) {
  data_copy <- copy(data)  # Work with a copy to ensure original remains untouched
  data_copy[, predicted_group := ntile(get(predicted_col), n_groups)]  # Divide into groups
  lift_table <- data_copy[, .(
    mean_actual_spend = mean(get(actual_col)),    # Mean actual spending
    mean_predicted_spend = mean(get(predicted_col))  # Mean predicted spending
  ), by = predicted_group]
  return(lift_table)
}

# Create lift tables for each model
lift_table_OLS <- create_lift_table(lift_data, "predicted_OLS", "outcome_spend")
lift_table_LASSO <- create_lift_table(lift_data, "predicted_lasso", "outcome_spend")
lift_table_Elastic <- create_lift_table(lift_data, "predicted_elastic", "outcome_spend")

# Plot function for lift charts
plot_lift_chart <- function(lift_table, model_name) {
  ggplot(lift_table, aes(x = predicted_group)) +
    geom_line(aes(y = mean_actual_spend), color = "blue", linetype = "dashed", size = 1) +
    geom_line(aes(y = mean_predicted_spend), color = "red", size = 1) +
    labs(
      title = paste("Lift Chart for", model_name),
      x = "Predicted Spending Group",
      y = "Mean Spending"
    ) +
    theme_minimal()
}

# Plot lift charts
plot_OLS <- plot_lift_chart(lift_table_OLS, "OLS")
plot_LASSO <- plot_lift_chart(lift_table_LASSO, "LASSO")
plot_Elastic <- plot_lift_chart(lift_table_Elastic, "Elastic Net")

# Display plots
print(plot_OLS)
print(plot_LASSO)
```

```
print(plot_Elastic)
```

Overall, how well do the models fit?

6 Traditional targeting profit prediction [20 points]

Now we work with the whole validation sample, including customers who were targeted and customers who were not targeted.

We use the preferred model that according to our previous analysis fits the data best. Using this model, we predict expected dollar spending for *all* customers in the validation sample. Consistent with standard marketing analytics practice (again think about the JCPenney example from class), we take these predictions to be indicative of what customers would spend if they were targeted, i.e.

$$\mathbb{E}(Y_i|\boldsymbol{x}_i, W_i = 1).$$

and, conversely, we assume that spending is zero whenever a customer is not targeted, $\mathbb{E}(Y_i|\boldsymbol{x}_i, W_i = 0) = 0$. (Here, we use $\boldsymbol{x}_i$ to denote the set of independent variables for customer i). Given this, we predict the expected targeting profit for each customer in the validation sample. The margin and targeting cost data are:

```
margin = 0.325          # 32.5%
cost   = 0.99           # 99 cents

# Use the best model to predict expected spending for all validation sample customers
y_validation <- testing_data$outcome_spend
X_validation <- model.matrix(outcome_spend ~ 0 + ., data = testing_data)

predicted_spending <- predict(elastic_net_model, newx = X_validation)

results_data <- data.table(
  customer_id = testing_data_reduced$customer_id,
  predicted_spending = predicted_spending
)

# Predict profit for all customers
results_data[, profit := (margin * predicted_spending) - cost]

# Determine customers to target (those with positive profit)
results_data[, target := profit > 0]

# Calculate the percentage of customers to target
percentage_to_target <- mean(results_data$target) * 100

# Print the result
print(paste("Percentage of customers to target:", round(percentage_to_target, 2), "%"))
```

What is the percentage of customers who should be targeted based on this analysis?

7 Incrementality [10 points]

Under what conditions is the analysis in the last section valid? In particular, what could invalidate the assumption that spending is zero whenever a customer is not targeted, i.e. $\mathbb{E}(Y_i|\boldsymbol{x}_i, W_i = 0) = 0$, in the context of our data? Discuss.

- Conditions for Validity

1. The assumption holds if the targeting intervention (e.g., catalog, ads) is the only factor influencing spending behavior. In such cases, non-targeted customers would not make purchases without being targeted.
2. The assumption is valid if customers who are not targeted share the same characteristics as those who are targeted and would behave identically in the absence of targeting.
3. If customers have no inherent propensity to spend without being targeted, $\mathbb{E}(Y_i|\boldsymbol{x}_i, W_i = 0) = 0$ is valid. This is often assumed in direct response campaigns where purchases are directly attributed to the targeting intervention.
4. The assumption is valid if the campaign's effects are confined to targeted individuals and do not influence non-targeted customers through social interactions or other indirect channels.

- Conditions That Could Invalidate the Assumption

1. If customers have a natural propensity to spend regardless of targeting (e.g., due to brand loyalty or regular purchasing habits), the assumption fails. For example, frequent buyers may continue spending even without a marketing intervention.
2. If targeting is based on customer characteristics that are also predictors of spending (e.g., targeting high-value customers), the assumption overestimates the impact of targeting. This is often the case in non-randomized targeting scenarios.
3. External influences such as holidays, economic conditions, or concurrent promotions could drive spending among non-targeted customers, invalidating the assumption.
4. If there is variation in customer behavior, with some customers spending independently of targeting while others require targeting to make purchases, the assumption does not universally hold.
5. Targeted campaigns may have indirect effects on non-targeted customers, such as through social influence, word-of-mouth, or shared household dynamics. This would result in non-zero spending for Wi= 0.