CRM and Machine Learning

Günter J. Hitsch

February 17, 2017

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

Data

The data is a development sample that consists of 280,000 randomly selected customers from the data base of a company that sells kitchenware, housewares, and specialty food products. The random sample represents only a small percentage of the whole data base.

Customers are identified by a unique `customer_id`. The targeting status of the customers as of October 14, 2015 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 mailing.

Load the `crm_DT` data.table.

```
load("./Data/Customer-Development.RData")
```

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

```
set.seed(1999)
crm_DT[, training_sample := rbinom(nrow(crm_DT), 1, 0.5)]
```

Data inspection

Summarize some key aspects of the `outcome_spend` variable. In particular, what is the purchase incidence, and what is the distribution of dollar spending, both unconditional and conditional on a purchase?

Data pre-processing

Data sets with a large number of inputs 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 one of them captures virtually all the information contained in both variables. Also, OLS (or logistic regression) without regularization will be infeasible with perfectly or near perfectly correlated inputs. Hence, it is helpful to eliminate some highly correlated variables from the data set.

Here is 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", "mailing_indicator", "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 to zoom 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

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.

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).

Note: When eliminating variables you need to be careful. For example, consider a situation where three variables, `x`, `y`, and `z` are highly correlated:

row	col	cor
x	y	0.99
y	z	0.99
z	x	0.99

If you don't pay attention you might eliminate all three variables, but of course one of them should remain in the data.

More generally, ideally one should eliminate only one variable at a time, then re-compute the correlation matrix, then eliminate another variable with correlation above the threshold, re-compute the correlation matrix, etc. However, given the fairly small number of highly correlated variables in the `crm_DT` data this is not strictly necessary here.

Predictive model estimation

Use the training sample to estimate the conditional expectation of dollar spending, based on all available customer information. Estimate and compare the following models:

1. OLS
2. LASSO
3. Elastic net
4. Random forest

Note: To save on computing time, it is OK to search over a coarse set of tuning parameters α in the elastic net.

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?

Model validation

Use the validation sample to compare the observed and predicted sales outcome.

To get a sense of the prediction variance I recommend to graph the relationship between the predicted and observed outcome (spending).

Then compare the mean-squared errors (MSE) based on the model predictions.

Create lift tables and charts (use 20 scores), plot the lifts (including standard errors), and compare the lift tables. I recommend **not** to normalize the lifts, thus we can assess the magnitude of predicted mean spending.

Overall, how well do the models fit?

Predictive modeling: Decomposition method

One can always decompose the conditional expectation of an outcome given the inputs as follows:

$$\mathbb{E}(Y_i|\boldsymbol{x}_i) = \Pr\{Y_i > 0|\boldsymbol{x}_i\} \cdot \mathbb{E}(Y_i|Y_i > 0, \boldsymbol{x}_i)$$

In our application, Y_i is customer-level spending. A common approach in the CRM industry to predict $\mathbb{E}(Y_i|\boldsymbol{x}_i)$ is as follows:

1. Estimate the conditional probability of $Y_i > 0$ using a logistic regression model.
2. Estimate the conditional expectation $\mathbb{E}(Y_i|Y_i > 0, \boldsymbol{x}_i)$ using regression analysis. Frequently, the sales outcome is transformed using the logarithm, and $\log(Y_i)$ is then regressed on the inputs $\boldsymbol{x}_i$.

Note: In general, $\mathbb{E}(Y)$ is not equal to $\exp(\mathbb{E}(\log(Y)))$. This issue is often ignored in practice. However, if Y has a log-normal distribution such that $\log(Y)$ is normally distributed with mean μ and variance σ^2, then the expectation of Y is given by the formula:

$$\mathbb{E}(Y) = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$

You can use this result to improve the prediction of Y_i in step 2 above. Note that σ^2 can be estimated using the residual variance in the regression:

```
sigma_2 = (summary(fit)$sigma)^2
```

Compare the results from the decomposition approach to the previous results based on the mean-squared error and lift table/chart.