ISOM 674 Final Project Report

Team 7: Jingyi Li, Levent Kayin, Wen Zhang

Overview

This is a classification problem to predict if there is a click for each entry row. We drew a random sample from the original dataset to create our training data and validation data. We used the training data to build models and validation data to see the out of sample performance for model selection. We prepared the data by transforming some variables, limiting the levels of the categorical variables, and matching levels of columns in the training data with those of the validation data and test data. We tried several types of models including lasso regression, random forest, stepwise-forwards-selection logistic regression, and neural network. Our best models come from lasso regression and random forest. We ensembled them to make our final prediction for the test data.

Data preparation

We started with drawing a 100,000-row random sample form the original training data. The columns are all categorical. We transformed the original "hour" column to be only the hour of day and another column showing the weekday of that day. We then checked the levels of each column by looking at the histogram/barplot of the factor distribution for each column (the x-axis has no numeric meaning). We dropped the columns with levels larger than 1/10 of the number of rows of the whole data frame, such as some ids. We also dropped a column that has very similar distribution of level values with another column and has large number of levels. Then we assigned the most frequent value in each column to the levels other than the top 30 frequent levels of that column. If the number of levels is below 30, then they are untouched for now. This is for dealing with too many levels in some columns.

We then split the data frame into training data and validation data as in 60%:40% ratio. The size of the data is large so we think a split validation should be good enough instead of using k-fold. The training data is for building model, and the validation data is for model selection. We checked the levels of each column in the training data and the validation data to see if there are factors in the x variables of the validation data that are not in the training data. If they don't, then the most frequent value in that column is assigned to the no-matching values. Therefore, the levels of training data and validation data will be the same.

Model building and feature selection

We tried to use forward-stepwise logistic regression, lasso regression, simple decision tree, and random forest. We picked lasso regression and random forest to continue to work in depth because they gave smaller log loss and are easy to work with. The decision tree, logistic

regression, and neural nets gave much higher log loss and it was time consuming to adjust them. We did not create dummy variables for the random forest model because we have levels lower than 33, which the randomForest package in R can handle and "it will check to see which factor levels should be on one side of the split and which on the other". (https://stats.stackexchange.com/questions/177852/best-practices-for-coding-categorical-features-for-decision-trees) For the lasso regression, we transformed the data using model.matrix().

For random forest, we set mtry to be a little higher than sqrt(number of x variables) and tried different parameters in ntree and maxnodes to get the lowest log loss on the validation data. Since random forest is a bagging approach, it was forcibly selecting different features and ensemble the functions it got from bootstrap samples, which should reduce overfitting.

For lasso regression, since lasso is a regularization approach to penalize overfitting and already does feature selection, it should reduce overfitting as well. We tried with different grids of lambda to find the lambda that gave the lowest log loss.

Performance on validation set

Random forest: log loss = 0.4124

Lasso regression: log loss = 0.4203

We chose random forest to predict test because it has lower log loss. A good way to predict can also be ensemble the two best models by taking the average of their predictions.

Prediction on test data

We cleaned the test data in the same way as we cleaned the training and validation data so that we can apply the models. Instead of looking for the top 30 most frequent values of levels in each column in test data and then assigning the rest to be the most frequent value in that column, we just assigned all levels not in the levels of the training data in each column of the test data to be the most frequent level of that column. Thus, the columns of the test data and the training data all had the same levels.

We then used our best random forest model to predict on the test data.

Code file description

"datacleaning.R": importing a random sample from original training data using sqldf package.

"Team7FinalProject.R": the code of data preparation, model building, and predicting.

Reference

reference: https://stackoverflow.com/questions/22261082/load-a-small-random-sample-from-a-large-csv-file-into-r-data-frame