Homework 4

10/15/2015

In this homework, you will analyze data from KDD Cup 2009: Customer relationship prediction. The data were given by the French Telecom company Orange. In the original competition, there were three tasks:

- churn: predicting the propensity of customers to switch provider;
- appetency: predicting the propensity of customers to buy new products or services;
- up-selling: predicting the propensity of customers to buy upgrades or add-ons proposed to them to make the sale more profitable.

The website for the challenge can be found here: *http://kdd.org/kdd-cup/view/kdd-cup-2009*.

Note that there were two challenges: small and large. We will focus on the small version.

The data can be downloaded from here: *https://github.com/ChicagoBoothML/DATA_____KDDCup2009_Customer_relationship*.

There are total of 50000 observations and 230 features. 190 of those variables are numeric and 40 are categoric. There are three possible targets to predict, as discussed above. Pick any one of those targets and try to build a predictive model. The data have been anonymized for this task. That means that the features do not come with meaningful names.

This is a challenging task for a number of reasons, that we list below.

- Large number of observations.
  - Computation may become slow. While building a mode, consider working with a subset of the training data.
- Large number of features.
  - There are 230 features. Many are of limited value and should not be included in the final model. For example, some features will contain only one value or have too many missing values; exclude such features.
- Missing values.
  - Most columns contain missing values. Some columns contain only missing values; remove such columns. For numeric features, there are a number of ways to deal with missing values. The simplest one is to simply impute the missing value with the mean of observed values. That is, if a particular value is missing in a column just simply substitute NA with mean of the column. For categorical features, the easiest way to deal with missing values is to treat them as if they are one of the levels. For example, if a feature takes on values a, b, and c, then the missing value NA can be treated simply as value d.
- Categorical features with a large number of levels.
  - Some categorical features have a large number of different levels. For example, if you try to fit a decision tree using the features, the algorithms will need to make a large number of splits. One way around this is to combine or aggregate levels for which there are a few observations. One suggestions would be as follows: create a new level "low", which combines all existing levels for which we have ≤ 250 observations; create a level "medium" that aggregates existing levels for which there are $250 - 299$ observations; create level "high" that aggregates all existing leveles for which there are $500 - 999$ observations; and keep all other levels as they are.

You tasks

Start by splitting data into two parts: train + validation set (~80% observations) and test set (~20%).

Data Cleaning

- Identify features with large number of missing values. Remove them from the data frame. Report which features did you remove.
- Replace missing values for numeric features with means of the observed values.
- Identify categorical features that have a large number of distinct levels. You do not need to follow our suggestion above when performing aggregation, nor do you need to aggregate at all. You may want to check how does leaving data as is affects classification performance. You may also simply ignore all features that have too many levels.

Feature selection

After data cleaning, you may want to identify important features to keep for your final model.

Here are some things to try:

- Fit a classifier for Y using each features on its own.
- Fit a classifier for Y using pairs of features on their own.

Evaluate performance of each one of these simple classifiers. Keep variables that have good performance.

Another thing to try is to fit a random forest model and keep important variables.

Choosing the final classifier

Now that you have selected a subset of variables, fit a final model. Random forests and boosting should give reasonable results here. Remember to use cross-validation or out-of-bag error estimate to select tuning parameters.

Finally, fit the model on all of the train + validation data and report performance on test.

Have fun!