Final Project

Stats 202

Salmonn Talebi

**Abstract:**

The objective of this project is to create a prediction algorithm that can determine whether the url is relevant for the query. We are given 12 predictors and a test set of ~80,000 observations. We investigate logistic regression, naïve bayes, trees, bagging and random forests, and ada boosting as prediction models. Random Forests gave the lowest cross validated test error and will be used as the classifier for the final test set.

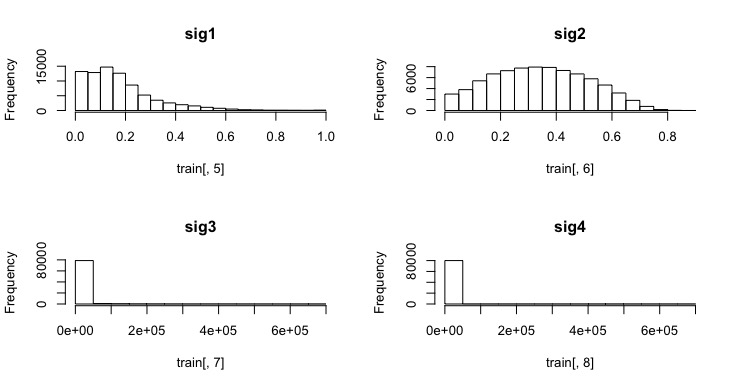
1. **Introduction**

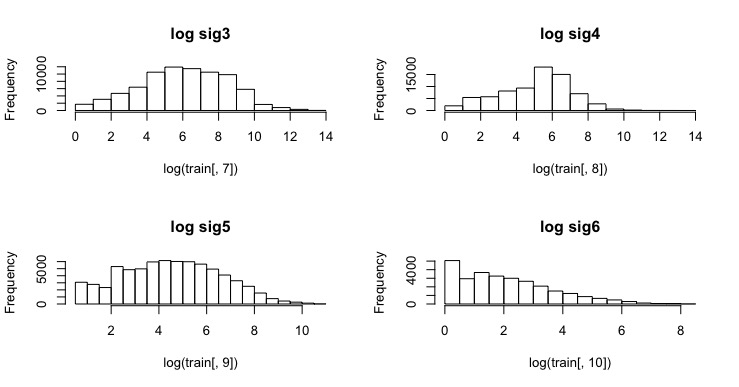
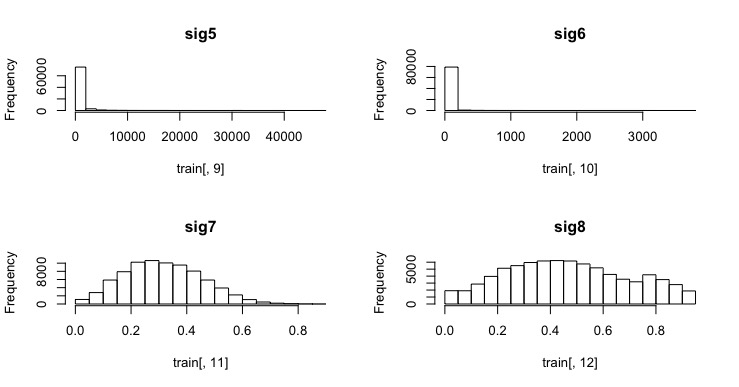
The goal of this project is to identify whether the url is relevant for a specific query given 10 predictors. A training set of 12 predictors and ~80,000 observations is provided. A final test data set with 30,001 is provided to evaluate the performance of the classifier.

1. **Data Observation**

No missing data is found in the training and test data sets. The second predictor, URL ID contains mostly unique sequential values and most likely does not provide any predictive power. Query\_length most likely represents the numbers of terms in each query. Is\_homepage represents whether the user launched the query from the user’s homepage or not. The remaining predictors, sig1-8 are unknown in what the value represents.

The 8 different signals were plotted on a histogram to determine the distribution. Three of the signals had a near normal distribution while the other 4 signals were skewed to the left. These 4 signals were log transformed to make the distribution near normal. The distribution plots can be seen below.





Looking at a correlation table between the predictors we see that sig3 and sig5 have a slight correlation. But not substantial enough for us to remove one of the predictors from our classifier. As expected, query\_id and url\_id is highly correlated because they both increased sequentially.



1. **Solutions Evaluations**

I evaluated five different classifier techniques on this dataset. In order to determine which classifier performed the best, I split the training data into a training and test set. For all methods I used a 5-fold cross validation method to estimate the test error. The 5-fold cross validation error is determined by splitting the dataset into 5 groups. The classifier will be trained on 4 of the groups and tested on the left out group. This process will be repeated 5 times until all 5 groups have been used as test sets. The test error is then average out over each 5 folds.

1. **Candidate Solutions and Data Selection**

For this project, I investigate logistic regression, naïve bayes, Decision Tree, bagging and random forests, and Adaboosting. Below is a table that lists a summary of each classifier with the cross validated error rate.

|  |  |  |
| --- | --- | --- |
|  | **parameters** | **CV Error Rate** |
| **Logistic Regression** | all variables | 34.4% |
| **Logistic Regression** | Subset of variables | 34.9% |
| **Naïve bayes** | laplass | 38.01% |
| **Decision Tree** |  | 36.10% |
| **Random Forest** | m=12 | 34.2% |
| **Random Forest** | m=6 | 33.9% |
| **Random Forest** | m=3 | 33.6% |
| **Ada Boosting** | 170 trees | 35.4 |

1. **Logistic Regression**

Logistic regression is a useful tool to predict categorical responses. Here I evaluate logistic regression using the glm() function in R. Before I begin training the classifier I log transform sig3, sig4, sig5, sig6 to normalize the distribution of each predictor. A 5 fold cv method was used to evaluate the performance of logistic regression on the training set. An average test error of 34.4% was given. Looking at a summary of the glm fit we can also get a sense of which variables are important:



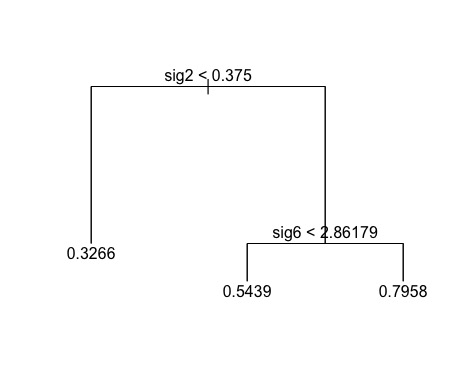
Here we see that sig3, query\_id, and url\_id do not seem to be significant. A logistic regression classifier is trained without those 3 predictors. The 5-fold CV error is 34.9%. The test error increased when we took out these 3 predictors.

1. **Naïve Bayes**

Naïve bayes classifier is a straight forward model that can be used for classification. Before I begin training the classifier I log transform sig3, sig4, sig5, sig6 to normalize the distribution of each predictor. I use the R function naiveBayes() to evaluate the classifier on this training set. The 5-fold cross validated training error is 38.01%. Turning on and off the Laplace smoothing did not change the error rates.

1. **Decision Tree’s**

Decision tree classifiers are advantageous in terms of interpretability. Here I evaluate a decision tree on the training data using the tree() function in R. Before I begin training the classifier I log transform sig3, sig4, sig5, sig6 to normalize the distribution of each predictor. The output is a tree with very few leaves. Since the tree is already small there is no point in determining the best pruned tree size. The 5-fold cross validation error rate is 36.1%.



1. **Bagging & Random Forest**

Bagging is a similar method to trees but uses an ensemble method to reduce the amount of variation in the model and is generally a better classifier. The disadvantage to using Bagging is you lose the interpretability that Tree’s provided. Here I use the randomForest() function provided by R. In the randomforest() function, we set m=p (bagging), m=p/2 (random forest), and m=sqrt(p) (random forest). Before I begin training the classifier I log transform sig3, sig4, sig5, sig6 to normalize the distribution of each predictor. Here we see that using a random forest with m=3 gives the lowest CV test error of 33.6%. m=6 and m=12 gave a CV test error rate of 33.9% and 34.2% respectively.

1. **Ada Boosting**

Another useful ensamble method is ada boosting. To test this classifier I use the gbm() function in R. Before I begin training the classifier I log transform sig3, sig4, sig5, sig6 to normalize the distribution of each predictor. 5-fold CV is used to determine the best number of trees to use in the boosting method. 170 trees are determined to give the lowest error rate. The CV test error rate is 35.4%.

1. **Fine Tuning**Using off the shelf parameters for each classifier I identified the Random Forest (m=3) classifier as having the lowest cross validated test error. Next I try to tweak the parameters to attempt to further reduce the test error.
2. **Number of predictors:**

I remove query\_id and url\_id from the analysis to see if it helps reduce the test error. The test error for 100, 500, and 1000 trees stayed roughly the same with and without the 2 variables. Removing the predictors did substantially reduce the computational time needed to train the classifiers. Since it looks like the two predictors do not provide any prediction benefit I will leave them out of the model.

1. **Number of trees:**

I range the number of trees used in the randomForest() function from 100 to 5000 trees. The test error bottomed out at approximately 700 trees.   
