BIG RANDOM FORESTS

An Analysis of the Springleaf Marketing Data

Final Report

Andrew Mehrmann (mehrman2)

Zachary Turner (zbturne2)

Noah Salas (nsalas5)

12/15/2015

STAT 542 - University of Illinois

INTRODUCTION

This project has two main components: first, a statistical analysis of the Springleaf Marketing data set, and second, a research investigation involving decision tree methodology. Our research goal is to create and compare a simpler ensemble decision tree method to the traditional random forest algorithm. We will test these different techniques using the Springleaf Marketing data, which simultaneously allows us to analyze and gain insights into the business classification problem.

DESCRIPTION OF DATA

As stated, the data we are using for this project comes from the Kaggle Springleaf Marketing Response competition. This data set is a typical machine learning scenario where we have large dimensionality with little to no inference on the features. When the data is read into R, the predictors are represented by 4 data types: factor, integer, logical, and numeric. Again, with no a priori information about the variables, we must assume that the integer variables are continuous, not categorical.

Summary of Data Types			
Factor	**Integer**	**Numeric**	**Logical**
51	1867	12	3

Figure 1: Original Predictor Variables

The data is presented to us in two splits: a training set (145,231 observations) and a test set (145,232). For this problem, our response is binary, and represents whether a customer responds to a direct mail offer or does not. Our goal is to to predict which customers will respond to a direct mail offer. It should be noted that the two classes are imbalanced: 23.3% of observations are targets, the rest are non-targets. In terms of missingness, 478 features (24.7% of all features) contain missing values. Of those features, very few are more than 50% missing (11).

DATA PREPARATION

We analyzed the *factor* variables and found that many of them had missing values coded as either "", "[]", or -1. We replaced those values with NA and noticed that some of the factors were noninformative, meaning that they had only one level other than NA, so we dropped those variables. We also noticed that some of the factors were date variables, so we parsed them into month and year columns, while dropping the original date variables. Similarly, we reduced all state variables into one of nine geographical regions[1] (New England, Mountain, etc.). Unfortunately, we were still left with some variables with more than 32 levels, which is a problem for many tree-based algorithms. Since there is no obvious way to reduce these, we dropped them.[2]

The *numeric* variables (including *integer*) presented an interesting challenge as many of them include extreme values. Some of these values do not appear to be randomly assigned, suggesting that they may be human-coded. For example, several variables include values like 999999996, 999999997, 999999998, etc., which we assumed to be missing indicators. Therefore, we coded all values in the set

{"","NA"," ","NULL",-1,-99999,999999999,999999998,999999997,999999996}

as NA. It also turns out that all of the variables read in as *logical* are strictly NA, so we dropped them. Once we had a data set that only included reasonable values for the predictors and NA, we were able to impute those NAs with the median for *numeric* variables and the mode for *factor* variables. We did a final check for any constant variables and removed them before writing the full (cleaned) data set to disk for use in further analysis.

Summary of Data Types - Full Data	
Factor	**Numeric**
48	1835

Figure 2: Summary of Full (Cleaned) Data

[1] http://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf
[2] We considered coding them as numeric, but left that to future work

DIMENSION REDUCTION

Redundancy Removal

After cleaning our data, our training set contained 1,883 predictors. This is too many features to fit into any reasonable model, or to manually inspect. Also, it is unlikely that all 1,883 predictors have an independent relationship with our target variable. For these reasons, we decided to perform a parameter redundancy dimension reduction. We computed the correlations between each predictor to determine which variables are redundant and can therefore be removed. We created a correlation matrix for our numeric variables, and examined variables that had a pairwise correlation of 0.9 or higher. Whichever of the two variables was most correlated with all other predictors was chosen as the redundant variable and therefore removed. By using the correlation threshold of 0.9, we removed 861 numeric variables from our data set. The distribution of data types in our reduced data set is outlined below:

Summary of Data Types - Reduced Data	
Factor	**Numeric**
48	973

Figure 3: Summary of Reduced Data

Principal Component Analysis

We also performed a principal component analysis in parallel to the redundancy removal in order to reduce our dimensionality in an alternative way. We replaced the numeric variables in the cleaned data with their principal components such that 95% of the variance was retained to produce a new data set. We scaled and centered the data to produce the principal components in order to overcome differences in the magnitude of the predictors. To be clear, this was done to the *full* (cleaned) data, not the *reduced* data, so we reduced the number of numeric features from 1835 to 537.

Summary of Data Types - PCA Data	
Factor	**Numeric**
48	537

Figure 4: Summary of PCA Data

Summary of Data Sets		
Name	**Features**	**Description**
cleaned_data	1883	NAs imputed and troublesome variables dropped
reduced_data	1021	Redundant variables dropped as determined by correlations
pca_data	585	Numeric variables replaced with PCs (95%)

Figure 5: Final Data Sets

MODELING METHODS

Sampling Methods

As we explored our various modeling methods, we were curious to see how the methods performed using different data sampling techniques. Since it isn't obvious the types of data we are working with, we thought it could be useful to explore different data samplings to help make our model more generalizable. More specifically, we examined the following types of data structures:

- Sampled Data: The dataset is sampled into *one* subset of a particular size, and then the model is fit on that one specific subset.
- Chunked Data: The dataset was chunked into N different subsets, and a model

was fit on each of the N subsets. These were then averaged to obtain the final predictions

- Bagged Data: The dataset was bagged (sampled with replacement) such that we had N different samples, each with the same number of observations as the original dataset.

XGBoost

Xgboost, short for Extreme Gradient Boost, is a powerful R package that runs parallel gradient boosted decision trees with a number of potential objective functions. The function has several parameters that can be tuned, which are important to note:

- max.depth: the maximum depth for each tree
- eta: the learning rate for the model. A lower learning rate results in a slower model, but makes the model less sensitive to overfitting.
- nround: the number of times the model iterates through the data. if nround>1, the model runs again on re-weighted training data

We ran the XGBoost procedure a number of times for each of the different sampling techniques described above. This was done as a sort of empirical parameter tuning. While we did not perform a formal parameter tuning (e.g. greedy, interval), we did note the changes in performance as various parameters were changed, and then reported out our optimal models.

Random Forest

Since we were limiting ourselves to working on off-the-shelf laptops, we were not able to build a Random Forest on the entire training dataset. Therefore we built forests on samples of 10% and 50% of the data. We also broke the data file up into 10 even chunks, and built a Random Forest of 101 trees on each chunk. After each forest was built, we calculated the prediction for each forest on the entire test set, and averaged the predicted probabilities for the 10 forests. This is similar to running a random forest on the entire dataset, but cost far less memory.

Our Method

We wanted to try a novel method of reading in chunks of data, doing predictions, and averaging many predictions over those smaller chunks of data. We thought it would be interesting to try reading in individual bootstrap samples (instead of storing them in temporary memory) and constructing individual decision trees on random subsets of the predictors. We combined the results manually in order to create

our own pseudo-random forests, instead of using a pre-defined R function. The goal of this was to reduce the memory load that is required in building traditional random forests, but the obvious drawback is that random subsets of predictors will be considered for the entire tree as opposed to at each split. Our hypothesis is that the additional randomness of considering a new subset of predictors at each split may not be worth the additional memory load.

RESULTS

We chose to run many of our tree-based algorithms on the reduced data to get a quick and dirty assessment of their efficacy. These results are shown below in Figure 6.

Summary of Results - Reduced Data		
Method	**Kaggle Score**	**Kaggle Submission File**
Random Forest - 10% Sample	0.74853	red_10.csv
Random Forest - 50% Sample	0.76246	red_50.csv
Our Method	0.71194	our_method_reduced_801.csv
Chunked RF	0.75386	rf_chunk_reduced_201.csv
Full XGBoost	0.76764	xgb_full.csv

Figure 6: Results for the Reduced Data

We also tested a few methods on the PCA-transformed data and found that they did not perform as well as even the worst performing methods on the redundancy-reduced data. These results are shown in Figure 7.

Summary of Results - PCA Data		
Method	**Kaggle Score**	**Kaggle Submission File**
Random Forest - 10% Sample	0.68238	pca_10.csv
Random Forest - 50% Sample	0.69985	pca_50.csv
Chunked XGBoost	0.68821	Submission3.csv

Figure 7: Results for the PCA Data

Finally, we tested some of the best-performing algorithms (in terms of prediction accuracy and computational efficiency) on the full dataset. Those results are shown in Figure 8.

Summary of Results - Full Data		
Method	**Kaggle Score**	**Kaggle Submission File**
Chunked XGBoost	0.76363	Avg_Submission6.csv
Bagged XGBoost	0.75795	bagged.csv
Full XGBoost	0.76618	xgb_full_clean.csv

Figure 8: Results for the Full Data

CONCLUSIONS

From our results, you can see why XGBoost was popular within the Kaggle

community for this competition. For starters, it's extremely computationally efficient. This is why we were able to run the method on the entire reduced data and full(cleaned) data sets. As seen in Figures 7 and 8, running an XGBoost on these data produced very similar Kaggle scores.

The performance of the sampled subset (10% and 50% of PCA and reduced data) Random Forests indicate to us that sample size is a significant driver of the Kaggle scores. This was expected because the Random Forest becomes more generalizable to the test data by increasing the the number of observations you build on. Even more surprising is that the Random Forest on a 50% sample of the data outperformed the chunked 10% Random Forests. This shows that simply averaging predictions made on smaller datasets will not perform as well as predictions made on larger datasets. We found a similar trend with the XGBoost models as well.

Our method was consistently outperformed by conventional methods for predicting on large data sets. This was to be expected, and our conclusion is that the extra randomness introduced by random forests is actually very important. As a side note, our algorithm ran very quickly and with little memory load compared with other algorithms (except XGBoost).

PCA did not work well on this data and for these methods. This was also to be expected as PCA is known to often cause worse predictions in classification tasks, but we did not expect the results to be as drastic as we saw.

REFERENCES

https://www.kaggle.com/darraghdog/springleaf-marketing-response/explore-springleaf/notebook

https://cran.r-project.org/web/packages/randomForest/randomForest.pdf

Our full (cleaned) data: https://uofi.box.com/s/vxezkm8ig3j1gfxstoyheeci6j9pf033

Our reduced data: https://uofi.box.com/s/fumkoha2l6xogsr706ivedpj2t90q1eg
https://uofi.box.com/s/h3bwsrmcd252t4l1u5uhhqsozljce51r

Our PCA data: https://uofi.box.com/s/2cta0nwp4tt5m14weo6bkkr8c0j8lckh
https://uofi.box.com/s/araae3gvhadmtzxe2t3p7tgi8vbvbbfa