FINAL PROJECT

Predicting Bank Telemarketing Campaign Effectiveness

Jared Hansen, Matt Isaac, Colby Wight

STAT 6910 -- Statistical Learning and Data Mining

December 13, 2018

Introduction

Telemarketing, in banking as well as in many other industries, is one marketing method companies use to increase profit. Whether this is an effective method or not is still up for debate. The aim of our study is to try to make this method more effective. We looked at the results of a telemarketing campaign conducted by a Portuguese banking company. Their goal was to get clients to sign up for a long-term deposit product. The purpose of this project is to see if we can help target the more likely clients that will sign up for these long-term deposits. If we are successful, this means increased productivity for the company. The company will potentially be able save time and money when conducting similar marketing campaigns in the future if they can increase their ratio of success to time spent contacting customers. This study will also give insight into telemarketing in general. We plan to use various machine learning methods on this data. We will then be able to understand what methods work better on these type of data, thus aiding others who wish to work with similar data in the future.

Studies in the past have been conducted concerning the effectiveness of telemarketing. The majority of these studies find a higher correlation in marketing strategies as opposed to client demographic. One of the biggest indicators of having a successful contact was the duration of the call. If the caller can keep the other person on the phone longer, they have a better chance of making a deal. These types of things may be interesting to know as well. Maybe it is more worthwhile to spend more time training the telemarketers than finding the right clients. We will further explore concepts such as these.

Data

Our data set can be found at: https://www.kaggle.com/sonujha090/bank-marketing. The dataset contains a binary "yes/no" response, with 16 other features that can be used for prediction. There are roughly 45,000 observations in the data set. Some of the features given include: age, occupation, marital status, education level, and account balance of the given client. Luckily the data is already very clean. There may be a few missing values present, but otherwise the data is in a format that is ready for predictive modeling. The key issue with this data is that there is a large imbalance in response values. About 88% of the observations have a "no" response associated with them. The data will be split randomly into training (80% of data) and test (20% of data) groups. We will use both upsampling and downsampling on the training data to achieve better results than simply using the unbalanced data.

Methods

We choose to use the following four methods: logistic regression, support vector machines (SVM) with both linear and gaussian kernels, random forests, and k-nearest neighbors (k-NN). These methods balance approaches that we've learned about in class as well as ones from outside of this class. All of these methods have been implemented in the R language.

Logistic regression seemed like a good choice for its ease of implementation. Not only is it relatively straightforward to understand and implement, it is pretty fast as well. It also has

some nice interpretability qualities. We will use the results from this "simple" method as a benchmark in comparison to the other methods. Logistic regression while simple, can be a good method to use in various circumstances. We used the default glm() function from the stats package making sure to set the type equal to "binomial" in the function call.

SVMs were chosen for their reputation as formidable predictors. While they may be more complicated and take longer to run and tune, SVMs often perform much better than simpler "off the shelf methods" as we will see. We decided to try SVMs with linear and gaussian kernels to see if the hassle of the extra parameter is worth it with the gaussian kernel. We used the e1071 package for these. For linear kernel SVMs there is only one parameter to tune: C. This parameter controls the width of the margin between our separating hyperplanes. Since our original data is fairly large in the context of parameter tuning for an SVM (45,000 observations), we used a subsample of the data to tune parameters. In order to ensure we had a decent number of "buyer" observations (the minority class), we took this subsample from the downsampled training data.

For Gaussian kernel SVMs there are two parameters to tune. In the e1071 package, they are "cost" (the C value in the SVM mathematical construction) and the "gamma" parameter (a scaled version of the sigma value in the SVM mathematical construction). Just as we did for linear kernel SVM tuning, since our original data is fairly large in the context of parameter tuning for an SVM (45,000 observations), we used a subsample of the data to tune parameters. We used the same 500-observations subsample from the downsampled training data to do this.

We thought we would also see how k-NN would hold up against these other methods. Due to the large sample size, and in order to make the training and tuning of the k-NN more feasible, fifteen subsamples (n = 3000) were randomly selected from the original training data, the upsampled training data, and the downsampled training data. The downsampled data was used to tune the value of k to be equal to 23. Ideally, we would have tuned the value of k for each model individually, but due to computational demands for the large sample sizes of the original and upsampled data, k = 23 was used for each of the three models. The implementation in the caret package was used for the k-NN classification.

Random forests are known to be one of the best 'out of the box' classifiers, meaning that little to no tuning is required. We set the number of trees to 100, and the number of candidate variables at each split was left at the default of the square root of the number of features. As our data has 16 features, this parameter was set equal to 4. We used the randomForest package for this method.

Results

We will report the findings from each method first and then compare and contrast. We will report both overall test error rate and test error rate for the buyers. For this problem we care much more about identifying customers that are likely to buy whatever product is being marketed. This allows telemarketers can spend their time calling those customers that are most

likely to buy, rather than wasting large amounts of time calling customers at random, most of whom will not end up buying.

1. **Logistic Regression**

Surprisingly, logistic regression did better than most methods tested in regard to overall error rate. Prediction for the buyers was still not good with a test error of .643. Using upsampled and downsampled hurt us in terms of overall test error, but helped us a lot when trying to predict buyers. We got a buyer test error rate of 0.171 on the upsampled data. This is not terrible compared to the rest of the methods used.

Logistic Regression	Overall Test Error	Buyer Test Error
Original data	0.093	0.643
Upsampled data	0.159	0.171
Downsampled data	0.158	0.172

2. **Support Vector Machines**

Linear Kernel

Through some iterative manual grid searching, we found the parameter of cost = 1.25 to give the best 10-fold cross-validated accuracy. This was used for constructing the three models below.

Model 1: Trained using original training data

This model took about 4 minutes to run, and gave a prediction error rate of 0.103 when predicting onto the test data. This is only slightly better than classifying all observations as "not a buyer", which would give a test error of just over 0.11. Also, this model did a very poor job of identifying buyers in the test data set, with a test buyer error rate of 0.813.

Model 2: Trained using upsampled training data

This model took about 12 minutes to run, and gave a test error rate of 0.167. This is actually worse than just classifying all observations as "not a buyer." However, as discussed above, what we truly care about is the percentage of buyers correctly classified. Although this model had a poor test error, it did much better at identifying the buyers in the test data set, correctly classifying 84.3% of them! (Test buyer error of 0.157.)

Model 3: Trained using downsampled training data.

This model took about 30 seconds to run, and gave a prediction error rate of 0.164 when predicting onto the test data. Again, a poor overall test error rate. Similarly to the upsampled model though, it did fairly well at classifying buyers, correctly identifying 84.0% of them in the test data set. (Test buyer error of 0.160.)

Linear Kernel	Overall Test Error	Buyer Test Error
Original data	0.103	0.813
Upsampled data	0.167	0.157
Downsampled data	0.164	0.160

Gaussian kernel

Through some iterative manual grid searching, we found the parameter of cost = 1000 and gamma = 0.001 to give the best 10-fold cross-validated accuracy. These parameters were used for constructing the three models below.

Model 1: Trained using original training data

This model took about 5 minutes to run, and gave a prediction error rate of 0.099 when predicting onto the test data. This is the best test error of all models. However, this model did a very poor job of identifying buyers in the test data set: it correctly identified only 28.1% of buyers.

Model 2: Trained using upsampled training data

This model took about 20 minutes to run, and gave a prediction error rate of 0.163 when predicting onto the test data. However, as discussed above, what we truly care about is the percentage of buyers correctly classified. Although this model had a poor test error, it did much better at identifying the buyers in the test data set, correctly classifying 87.4% of them!

Model 3: Trained using downsampled training data

This model took about 30 seconds to run, and gave a prediction error rate of 0.164 when predicting onto the test data. Again, a poor overall test error rate. Similarly to the upsampled model though, it did fairly well at classifying buyers, correctly identifying 86.6% of them in the test data set.

Gaussian Kernel	Overall Test Error	Buyer Test Error
Original data	0.099	0.719
Upsampled data	0.163	0.126
Downsampled data	0.164	0.134

3. **K Nearest Neighbors**

Model 1: Trained on original data

Using the original data as the training set, the k-nearest neighbors performed moderately well when considering overall test error of 0.111. However, when only considering the buyers, the test data yielded a test error of 0.870 . This was the first indication that k-nearest neighbors may not be well suited for this problem.

Model 2: Trained on upsampled data

Things got somewhat better when resampling techniques were used. For the upsampled data, the overall test error was 0.248, which is not remarkable when compared to some of the other methods we used. For the buyers alone, a test error of 0.265 was achieved. This was a huge improvement from the test error of 0.870 obtained with the original data.

Model 3: Trained on downsampled data

Training the k-nearest neighbors algorithm on the downsampled data yielded similar results to the upsampled data. The overall test error was 0.241, and the test error of the buyers was 0.262.

k-Nearest Neighbors	Overall Test Error	Buyer Test Error
Original Data	0.111	0.870
Upsampled Data	0.248	0.265
Downsampled Data	0.241	0.262

4. **Random Forests**

Model 1: Trained on original data

Using the original data (without any resampling techniques) for the training data on the random forest classifier yielded mediocre results. The overall test error was 0.090. This seemed good at first glance, until looking at the test error for buyers, which was 0.506. Where this is slightly worse than guessing whether someone is a buyer or not, this model did not meet our expectations.

Model 2: Trained on upsampled data

With the upsampled data, we saw the random forests (as other methods) improve when predicting the buyers. The overall test error increased slightly to 0.096, but the test error for buyers decreased to 0.363 (slightly better than guessing).

Model 3: Trained on downsampled data

The downsampled data worked the best for the random forest classifier. While the overall test error increased even more to 0.165, the random forest classifier achieved a test error of 0.085. Because we were more interested in correctly predicting the buyers than we were in the overall correct classification, a loss in overall predictive accuracy was worth the increase in predictive accuracy of the buyers.

Random Forests	Overall Test Error	Buyer Test Error
Original Data	0.090	0.506
Upsampled Data	0.096	0.363
Downsampled Data	0.165	0.085

Conclusion

The table below shows a summary of the best accuracy obtained for each of the classifiers. The poorest performing classifier for these data was k-Nearest Neighbors. The classifier that achieved the highest accuracy (smallest test error) was by far the random forest trained on the downsampled data. Although the overall test error of 0.165 might not be anything remarkable, the random forest was able to achieve a test error of 0.085 (i.e. correctly predict 91.5%) for the individuals who bought the marketed product. The information gained by this predictive model has the ability to guide telemarketers to those who are most likely to buy the product being offered. We believe that business owners would be quite satisfied with this level of predictive accuracy.

Classifier	**Training Data**	**Overall Test Error**	**Buyer Test Error**
Logistic Regression	Upsampled Data	0.159	0.171
SVM (linear kernel)	Upsampled Data	0.167	0.157
SVM (gaussian kernel)	Upsampled Data	0.163	0.126
k-Nearest Neighbors	Downsampled Data	0.241	0.262
Random Forests	Downsampled Data	0.165	0.085

By far the biggest challenge for us was dealing with the imbalance in the response variable. Using the original data did not give very good results. Through upsampling and downsampling we were able to get much better results. There was no clear winner as to which of the sampling methods worked best. In the case of random forest, training on the upsampled data lead to mediocre predictive accuracy while training on the downsampled data gave us our best

classification of all the methods for predicting buyers, which was quite surprising to us. Given more time, we would like to test other methods and see if we can find anything that can predict as well as the random forest. For example, a neural network might give even better results. It would also be interesting to apply this approach to other marketing campaigns and see if we get similar results.

Contributions

Matt: headed up compilation of the progress report, proofread project proposal, exploratory data analysis, did analysis of the data using Random Forests and K-Nearest Neighbors methods.
Colby: headed up compilation of the final report, proofread project proposal, exploratory data analysis, did analysis of the data using logistic regression.
Jared: found data set, wrote the project proposal, proofread progress report and final report, did analysis of the data using linear and Gaussian kernel SVM.