James Taylor

Assignment 3

University of Maryland University College

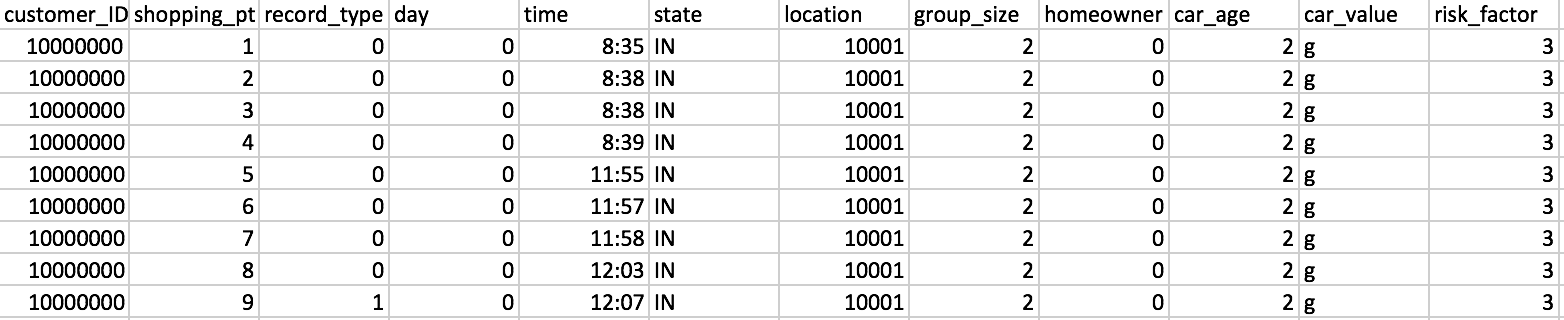
DATA640

Dr. Knode

james.taylor.ps3@gmail.com

**Introduction**

The goal of this paper is to use ensemble models to make accurate predictions on unseen data. Different approaches will be used and see which is the best performer. The software used will be SAS Enterprise Miner.

The data used was sourced from the Allstate Purchase Predication Challenge on Kaggle. The data is on online customer information, the products the customer looked at and when/what they ended up purchasing if they did purchase. The target variable is if the customer purchased or not, an imbalance variable with 14.5% of observations having a purchase. In the data, the same customer normally has multiple observations. This is because they look at different coverages until they settle on one or don’t purchase any. Below is an example of some variables and observations. 

**Data Preparation**

There are 665,250 observations on 25 variables. There were 277,840 missing values which were ignored. The data was randomly separated into training and validation sets. Not all the variables were used in the decision tree models. All of them were unique identifiers for the specific customer, location of the customer and the time of the query. None of the variables were transformed. Some of the variables had to be specified as nominal, ordinal and interval so the software would treat them properly in the models. The models will be trained on 70% of the whole dataset and validated on the remaining 30%.

**Predictive Models Developed**

There were four different ensemble models developed to predict which observations will purchase a product. These techniques were bagging, boosting, gradient boosting and random forest. Each of these utilize multiple decision trees to make predictions. What makes ensemble models powerful is the fact that they use multiple simple models to make their prediction.

Bagging utilizes multiple decision trees that are individually trained on different subsets of the training data. Some observations will be used in the training of more than one models. Predictions are made by presenting the independent values to all the models, collecting the predicted values from all the models and forming a consensus from them, like a majority rule. Our model had 10 trees trained on a 25% sample of the original set.

Boosting is another ensemble technique that uses multiple decision trees. It starts with a single decision tree trained on all training observations with equal weights. The incorrectly classified observations from this model have their weights increased (more important). Another decision tree is then built incorporating those weights in an attempt to classify them correctly, along with the other observations. This sequence occurs until some criteria is met. Our model did this 10 times.

Gradient boosting as a type of boosting model. It has the same sequential process as normal boosting. The difference is the residuals (misclassified) observations are the target variable for the next tree. The model is attempting to find patterns in the residuals to make better predictions.

A random forest is the final ensemble model that was built. This technique randomly selects different input variables for each iteration of its decisions tree model building. This is a good way to have weak performing models individually, but combining all the predictions from them to become a strong one. This is the same logic to all the above ensemble models. The model that was used had 30 trees.

**Results**

The results of the models on the validation (unseen) data is displayed in the figures in the appendix. The most important measure in those figures is the misclassification rate. That is the percentage of predications that were incorrect on data the models had not seen.

As *Figure 1* displays, the bagging and gradient boosting models were tied on misclassification rate. They also have similar sensitivity and specificity rates. There is almost no difference in performance between the two.

The next best performing models was the random forest technique. They both had a misclassification rate of 14.4%. This may appear to perform almost as good as the bagging and gradient boosting models but this isn’t really the case. The random forest model achieved this accuracy by predicating almost all test observations as false, but because the target variable was imbalance it was able to achieve low misclassification. The dataset only had 14.5% of its target variable observations as true, so if a model predicted false every time it would still have a misclassification rate of .145. This is what the random forest did basically, explaining why the sensitivity of the model was less than 2%.

The boosting model was the worst performer. The model misclassified test data 42% of the time. The specificity was exceptionally low, about 50%. The model had the most true positive predictions by far with 27,307. This is greater than four times the amount the bagging model. The model also had many false positive predictions, which is why it performed so poorly.

Some basic decision trees where built so it could be inspected how the models were likely separating the data. The shopping point variable was very important for all of them. This variable tracked how many times this customer had looked at pervious offerings. It appears the more customers looked at different options, the more likely they were to purchase eventually. The models frequently separated on shopping points with 6, 7 and 8 views of offerings. This may support a theory that the more a customer is viewing different options and revisiting the site, the more likely they are to commit.

The depth and amount of decision trees utilized for gradient descent and random forest makes inspecting them difficult. Tweaking the parameters of the Random Forest model yielded varying results. The default number of trees built for the Random Forest was 100, which produced only two true cases on the training data and none on the validation. There were too many trees with variables that don’t predict purchasing and a majority could not be formed by all the trees to produce a true prediction. The results of displayed in the figures are from 30 decision trees. With less trees, there is a better chance to form a majority. Not all the variables are equally good predictors for a purchase, so the decision trees with bad variables won’t help.

**Conclusions and Takeaways**

For the purpose of predictions, one would should not struggle to decide between the bagging and gradient boosting models. Their performance was nearly exact. It is out of the question however for the boosting and random forest models to be used. With more parameter tweaking, those poor performing models may even outperform the others, but not much time was invested to explore this. Any model with nodes or leaves, like neural networks and decision trees, have so much potential to tune that one could spend lots of time doing just that. This of course can have negative consequences when the model gets deployed. The tuning may have been optimized the model for this sample of data (both training and test), causing it to struggle on new data.

From the inspectable decision trees, most of the accuracy is derived from the shopping point variable. It seems the more a person views different insurance quotes the more likely they end up buying one.

Since the data was sourced from Kaggle, other’s analyses were also displayed. They appear to have success breaking the data into different training sections based on shopping point. It seems that treating someone who got a quote twice differently from someone that got 10 quotes is a successful strategy.

There may be better ways to show the variables to a model allowing it to extract information. For example, there may be some variable interaction that makes a certain type of person very likely to purchase.

Appendix A

Figure 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Model** | **Misclassification** | **Sensitivity** | **Specificity** |
| 1 | Bagging | .133 | 0.214 | 0.979 |
| 2 | Boosting | .428 | 0.938 | 0.509 |
| 3 | Random Forest | .144 | 0.018 | 0.998 |
| 4 | Gradient Boosting | .133 | .22 | .978 |

Figure 2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number** | **Model** | **FN** | **TN** | **FP** | **TP** |
| 1 | Bagging | 22864 | 166858 | 3615 | 6239 |
| 2 | Boosting | 1796 | 86808 | 83665 | 27307 |
| 3 | Random Forest | 28567 | 170238 | 235 | 536 |
| 4 | Gradient Boosting | 22700 | 166704 | 3769 | 6403 |