Jeremy Ogg (Leader)

Gastone Riccardo Tolli

Seehyun Park

**Machine Learning Coursework 3**

**1. Introduction**

Orange telecom company’s innovation and RD subdivision issued a challenge in 2009 for teams to develop predictive models for their Customer Relationship Management (CRM) goals [1]. The challenge was to find the best predictive model that can utilize the inputs provided by the company to predict the three outputs of interest: churn, appetency and up-selling. Churning being the propensity of a customer switching providers; appetency is the propensity to buy new products or services, and up-selling is to purchase upgrades or add-ons for a more profitable sale [1]. The company has provided the model training data, to be used for model training, and a test variable dataset, to produce a prediction output. This report focuses on the efforts of our team to produce the three best individual models for the respective output classification variable (i.e. churn, appetency and up-selling). The task distribution is as follows: Jeremy Ogg for churn prediction, Gastone Riccardo Tolli for appetency prediction and Seehyun Park for up-selling prediction.

**2. Churn Prediction – Jeremy Ogg**

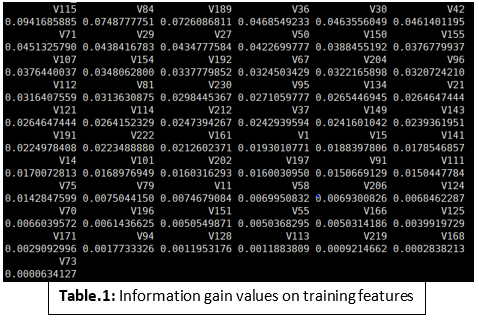
**2.1 Methods**

**2.1.1 Data Preprocessing**

The training data provided in the file *train\_X.csv* had 5,034,449 total missing values, as well as columns that were primarily missing values. This informed the initial dataset cleaning of: removing columns with 90% or more NA values, removing any columns with 70% or more missing values, removing any factor variable columns with more than 53 levels and removing any rows with more than 80% missing values. Once this step was complete, the training data was shuffled and split into a model training dataset and model validation dataset, 75% and 25% of the total training data, respectively. Due to the high levels of class imbalance in the training dataset, 22921 values for stay and 1830 values for churn, the hybrid resampling method SMOTE (Synthetic Minority Over-sampling Technique) was used to rectify this discrepancy. The resulting class distribution was 7320 and 5490 for stay and churn, respectively. The resulting data-frame had a total of roughly 8.4% NA values, so the k-nearest neighbour imputation method [2] was used to assign values based on similar observation cases. The final preprocessing methods were implemented during the model training process, these were: near-zero variance variable removal, zero variance variable removal, scaling and centring.

**2.1.2 Feature Selection**

           After preprocessing the data and removing unwanted variables, the training data had a total of 61 features. The datatypes were a mix of numerical values and factor variables with individual level values. The information gain method was used to identify the predictive ability that these factors had, and subsequently remove any with a low predictive ability. Table.1 illustrates the predictive ability of the 61 features, in descending order. This table was used to identify any feature with an information gain value of less than 0.015 and remove it for the following model selection.



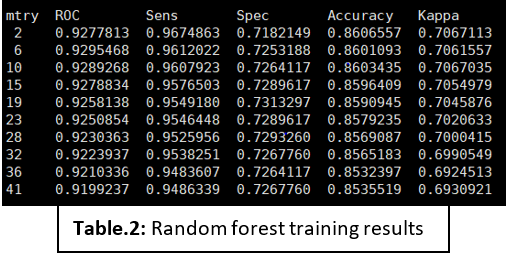
**2.1.3 Model Selection**

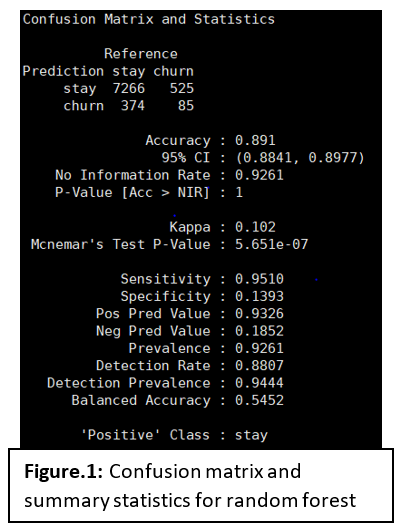
The preprocessing and feature selection methods yielded a training dataset that has 12810 observations, 41 predictor variables and 1 outcome variable. The model selection process utilized the R caret package to train the models and test the effectiveness of said models on the validation set. The models implemented were caret’s: random forest, stochastic gradient boosting, adaboost classification trees, k-nearest neighbour and average neural network. The caret function trainControl was used to set the parameters for the training function, which utilized a 10 fold cross-validation method for resampling.

**2.2 Results**

**2.2.1 Random Forest**

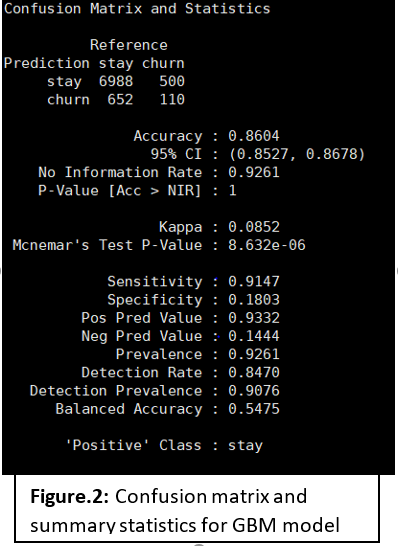
The random forest model’s preprocessing centred and scaled 30 features, while ignoring 11. The number of trees that were used for this model was 500 and the best performing iteration was when the model used 6, which represents the number of features used at each split of each tree node (**Table.2**). The training results were: ROC (.93), sensitivity (.96), specificity (.73), accuracy (.86) and kappa (.71). The testing results can be seen by **Figure.1**, where the confusion matrix and summary statistics are highlighted. The test accuracy of 0.891 (95% CI: 0.8841, 0.8977) is a surprisingly high achievement; however, the low specificity (.1393) and balanced accuracy (.5452) indicate that this model is being biased by the positive class ‘stay’ and unable to achieve a high amount of negative class prediction value.

****

****

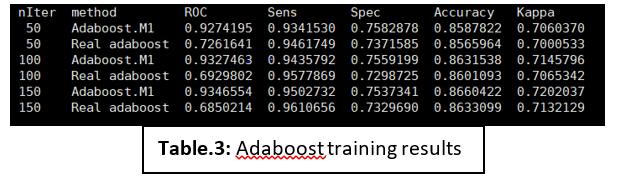
**2.2.2 Stochastic Gradient Boosting**

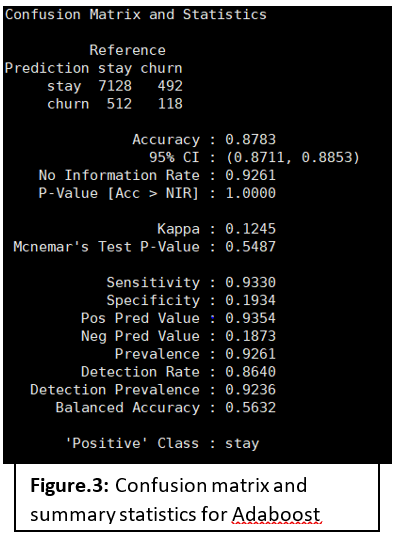
The stochastic gradient boosting model (GBM) shared the same preprocessing results as the random forest; centred and scaled 30 features, while ignoring 11. In order to find the optimal parameters, a tuning grid was created for this model. This grid used the tuning parameters: interaction depth (10,15,20), number of trees (between 1500 and 4500), shrinkage (0.1) and n.minobsinnode (20). The optimal tuning parameters used were: number of trees (4500), shrinkage (0.1), n.minobsinnode (20) and interaction depth (20). The training results of this model were: ROC (.93), sensitivity (.94), specificity (.76) and kappa (.72). With the test accuracy of this model (.86) and the specificity (.18), this again follows the same inference of the random forest model, where the model’s ability to predict the negative class is underwhelming.

****

**2.2.3 Adaboost Classification Trees**

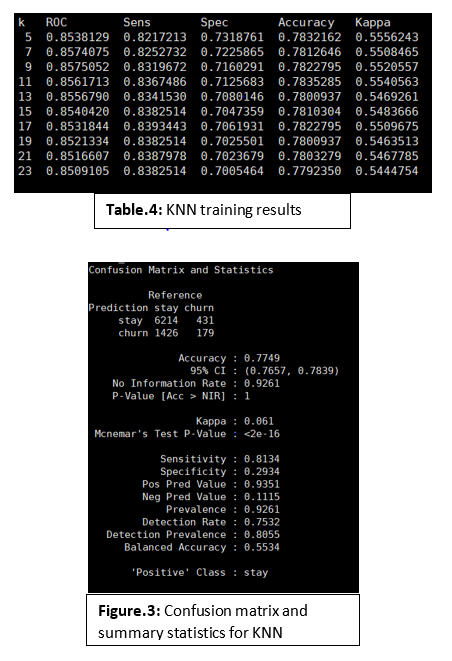
The adaboost classification trees model shared the same preprocessing parameters as the previous models mentioned. Much like the GBM model, there was a parameter grid search to optimize the model; however, the attempted search parameters took many hours to implement and thus were forgone for the default parameters. The final tuning parameters were: the number of iterations (150) and the method (Adaboost.M1). The training results can be found in **Table.3**, where the optimal results are: ROC (.93), sensitivity (.95), specificity (0.75), accuracy (.87) and kappa (.72).  The test results can be seen in **Figure.3**, and while the accuracy (0.8783) is lower than the random forest model, it does have a higher specificity (0.1934). While this specificity is still unacceptably low, it does indicate that the adaboost model is more appropriate for churn prediction than the previously mentioned models.

****

****

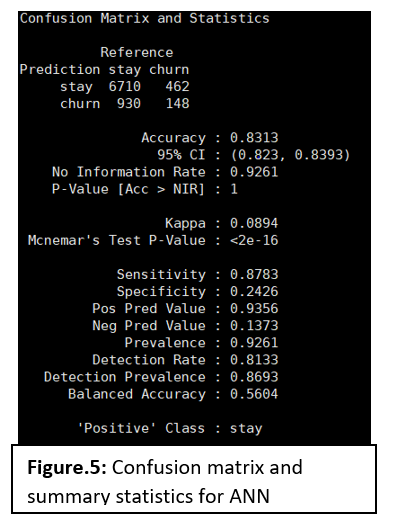
**2.2.4 K-nearest Neighbour**

The k-nearest neighbour (KNN) did not utilize that same feature preprocessing features as the prior models; instead, this model did not ignore any feature and scaled and centred all 41 features. However, it must be noted that there was a necessary transformation of factor variables to numeric for caret’s KNN model and was thus applied to the training data. The training tuning parameter results can be found in **Table.4**, where the optimal parameter is k equalling 9. The results of this model were interesting, despite it having the lowest test accuracy (.7749). The specificity is the largest with a value of .2934; however, this seems to be at a sacrifice of the other evaluating values.

****

**2.2.5  Average Neural Network**

           An average neural network (ANN) is when the same neural network is trained various random number of ‘seeds’, then all of the results are used for prediction [3]. This model utilized the same feature preprocessing as the random forest model. The training of the model yielded that the optimized parameter was to have five hidden-layer nodes in the neural network; however, it must be noted that anything greater than five nodes provided errors in the model, thus the potential of this model was not fully explored. The test results can be seen in **Figure.5**, while this model’s accuracy was acceptable (.8313), it too was unable to garner a larger amount of specificity (.2426) and negative prediction value (.1373).

****

**3. Appetency prediction - Riccardo Tolli**

This section will report the process with which I obtained the prediction for the variable “appetency”.

**3.1 Preprocessing**

The dataset needed data cleaning and preprocessing. At first the rows and columns with many NAs or blank cells (>50%) were removed so that I could perform K-Nearest Neighbor to fill the few NAs left. However, KNN ended up being a computation that was too demanding to perform locally and would have needed the use of the computer cluster. Therefore, I fell back on removing the columns with more than twenty percent Nas and and only removed the rows with more than 50% Nas that belonged to negative y observations in an effort to keep all positive y observations (which were scarce). This process was carried out only on the train set, while the test set was only reduced in terms of columns but not rows.

The second biggest problem with the data is the fact that the y variable is heavily imbalanced. The ratio of positive observations (“1”) compared to the negative ones (“-1”) was 0.018, specifically 32417 negative and 584 positive observations. This cannot stay this way or the machine learning algorithms will either not have predicting power or possibly will not even converge, returning an error. This will be handled later with SMOTE (Synthetic Minority Over-sampling Technique).

The next task was to remove 9 features in the dataset that had almost a constant value as these will not have much predicting power. Before doing so I checked if the non-dominant values in those features were heavily correlated with positive y observations. However, none of those were so I proceeded to remove all 9 features from the dataset without being afraid of losing valuable information. This was done on both train and test.

As KNN could not be an option I filled the train and test sets with the mode,median or mean of their respective column. The mode was used for the categorical variables, the mean was meant to be used on numerical columns without outliers however since all numerical columns had outliers the median was used to fill the Nas.

One last preprocessing task was carried out. Thirteen categorical variables had a very large number of categories therefore I aggregated (for train and test) all the categories that had less than 400 observations to a new variable, unique to its column. With the exception of three of those columns which had an extremely large number of categories and very few of those categories were over 400 observations. Since the variable names was not known it was not possible to recode the variables in such a way to keep them; therefore, those features were removed from the train and test set.

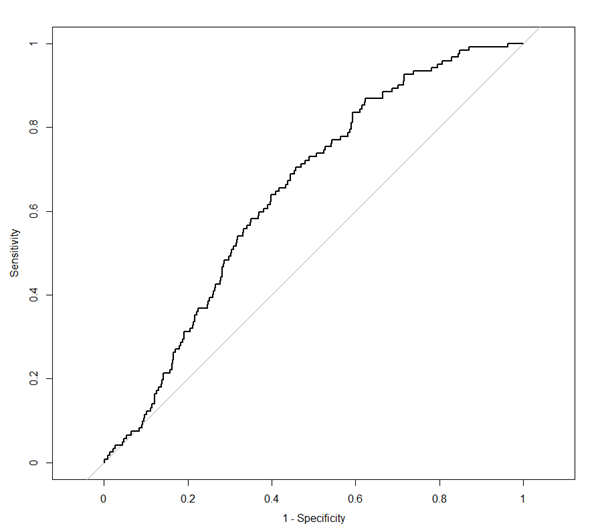
At this point the training set was split into a further train and test set. The SMOTE algorithm was applied to the new train dataset only in order to deal with the problem of imbalanced class. The drawback of this method is that it creates overfitting which will most likely affect the prediction of the predictive models developed later.

**3.2 Modeling and Feature Extraction**

The next predictive models will be trained on a partition of the original train set and tested on a test partition of the original train set. The models will be assessed with the AUC (Area Under Curve) of the ROC curve (Receiver Operating Characteristic Curve). The threshold was selected with the trapezoid method.

The first model fitted was a simple logistic regression with cross validation. However, many categorical features in the dataset had several classes which did not consistently show up in both train and test, therefore, cross validation causes the test to have observations that are not present in the training set.

Logistic regression was carried out without cross validation. The ROC curve can be found in Fig.1.

Fig.1

Below in Fig.2 it is possible to find the confusion matrix and statistics for this model.

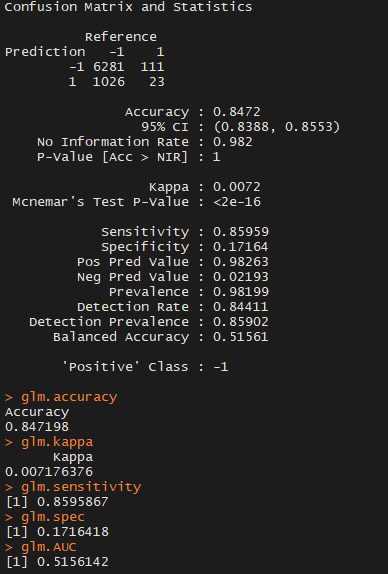


Fig.2

The model underwent post processing, Fig.3 shows how the model’s statistics changed after post processing.

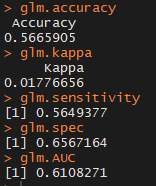


Fig.3

After using post processing ,namely, finding the probability threshold that maximizes the above curve. The AUC value went from 0.529 to 0.61.

From the model above the most important features were extracted and a second logistic regression was fitted. Fig.4 is its ROC curve.

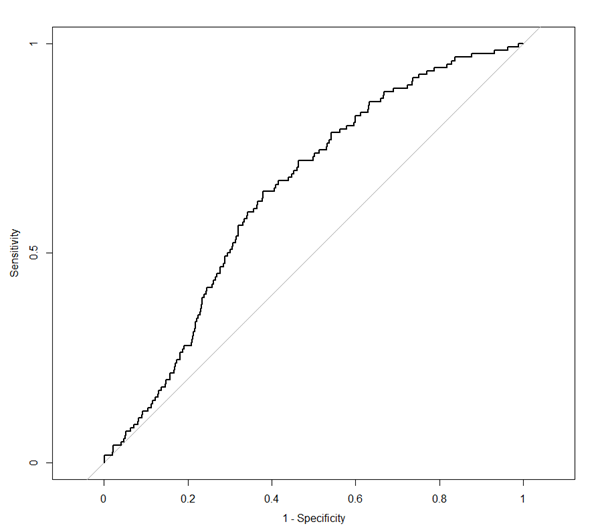


Fig.4

The AUC value is 0.62 after post processing.

This model was, eventually, chosen for submission because it outputs the highest AUC of the models tested for appetency. Even though the Kappa-statistics value is very low as it should be 0.6.

Fig.5 and Fig.6 show the metrics with which the model was chosen in Rstudio. They are in order: Confusion Matrix,Accuracy, Kappa-statistics, Sensitivity, Specificity and AUC value.

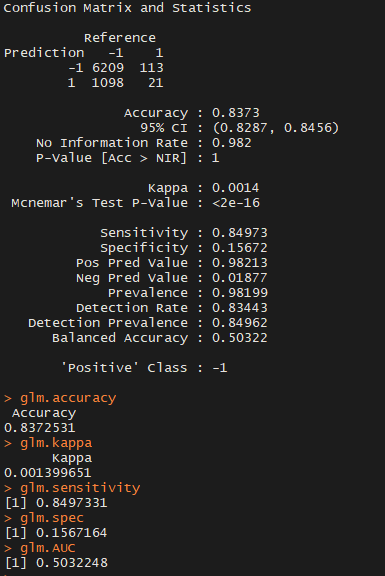


Fig.5 Before applying threshold

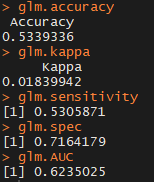


Fig.6 After applying threshold

The next model is a Random Forest. The mtry was optimized with the function tuneRF; however,  the confusion matrix on the train set was extremely poor therefore this model was scrapped.

Fig .7 shows the importance of the variable according to the Random Forest, although this information was not used as the model performed too poorly.

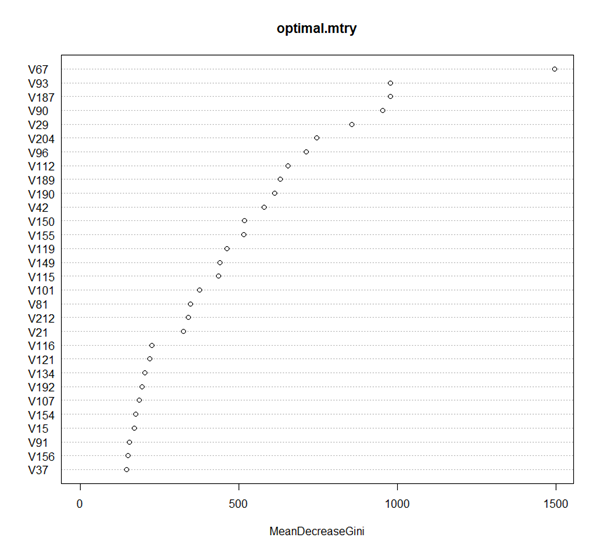


Fig.7

A Support vector machine model was attempted. However, both the confusion matrix and the initial AUC were too poor and was scrapped.

The figure below (Fig.8) shows the statistics for the SVM.

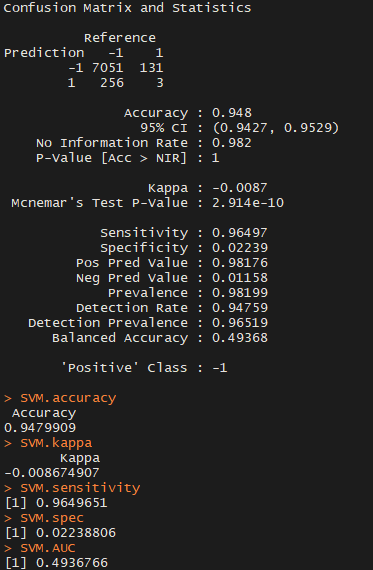


Fig.8

The last model attempted was a GBM (Gradient Boosting Machine). This model was tuned with the caret package and the expand.grid function. A variety of values for the parameters were tuned. However, the extended computational time required for tuning prevented an optimal tuning. The result of this GBM model were the least accurate, a summary of its statistics and confusion matrix is below, in Fig.9. The parameters for this final GBM model were n.trees=3000, shrinkage=.01,n.minobsinnode=100 and interaction.depth=7.

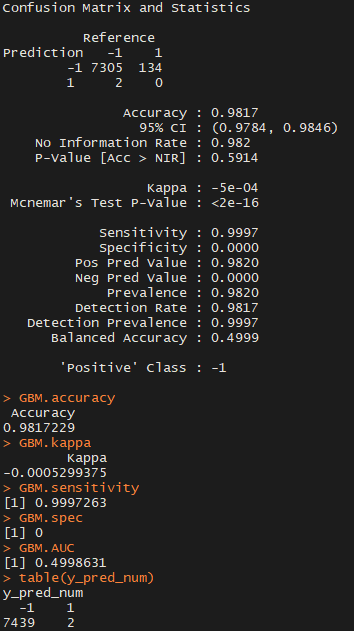


Fig.9

The following figure (Fig.10) shows how the statistics behaved after post processing.

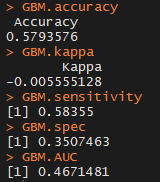


Fig.10

It is worth mentioning that neural network was attempted, however, for the nature of the data a simple neural network was computationally unmanageable. Possibly, a Sparse Convolutional Neural Network would have worked but I did not attempt this.

**4. Upselling Prediction – Seehyun Park**

**4.1 Data preprocessing:**

KDD Cup 2009 Customer Relationship Prediction data is certainly needed data preprocessing. It was too ambiguous to analyze the data without data preprocessing because there were over 50% of the missing values or empty values.  First step of preprocessing is to calculate the percentage of the missing values for each column, and then the data was truncated to columns with a missing percentage of less than 20 percent. Then the next step is to find the empty value index and remove the columns with more than 100 empty values.  After these two steps, the original 230 columns have been reduced to 59 that consist of numeric and factor characteristic variables. However, since the missing value still exists, I replaced the average value of each column for the numeric variables. There were 98 missing value in the factor variables among 33,000 rows which are identified as small portion of the overall, so it can be said that removing missing rows would not have a significant impact on the data.  Removing the factor variable columns with more than 53 different categories are also concerned due to the fact that when running the tree-based algorithms, it produces an error if one column has more than 53 distinct categories. The trainX data consisting of 32,903 rows and 47 columns was obtained after all theses above preprocessing. It is required to divide into train and validation set to check the model performance. The 80% of the trainX data is used for building the model; the remaining 20% ​​of the trainX is used for the validation set, that is, a set of model evaluations.

**4.2 Feature selection:**

The first thing to be checked before making a feature selection is to look at the data balance.  Data balancing is necessary because if the training data contains one class that accounts for more than 90% of the entire data set, then the model can be highly biased so that the predictive performance would be lower.  The data was originally heavily imbalance. It has 24,379 of -1 value and 1,943 of +1. Since the gap of two classes was large, the SMOTE (Synthetic Minority Over-sampling Technique) method was applied in order to match the balance of the training dataset. After applying SMOTE, balance was set to 7,772 for -1 and 5,829 for +1.  The Information Gain techniques also applied to reduce a bias towards multivalued attributes and finally it removed the 4 lowest attributes. The final training dataset consists of the remaining 43 predictors, and is used for training the model. The variables were first extracted through Information Gain and then the model was created and the model performance was tested in validation set.

**4.3 Methodologies:**

Machine learning algorithms that applied for the KDD Cup 2009 Customer Relationship Prediction are GBM (Generalized Boosted Regression Models) and ADA (AdaBoost).  I have created the eight different combinations for each model. The first comparison determines whether or not SMOTE is applied to the data balance. So it provides two combinations, and the extracting variable importance would be the second step.  The final combination uses post-processing methods youden or topleft to find the threshold and predict the model. If all combinations are combined by 2 x 2 x 2, so it can be represented as total of 8 combinations for each model. Both algorithms tune the stability of the model by repeating cross-validation 10 times. In the case of GBM, the expansion grid extends the ntree from 50 to 1000, and the shrinkage also extends from 0.001 to 0.1. In case of ADA, maxdepth is extended from 1 to 3 to create a tune grid.

**4.4 Results:**

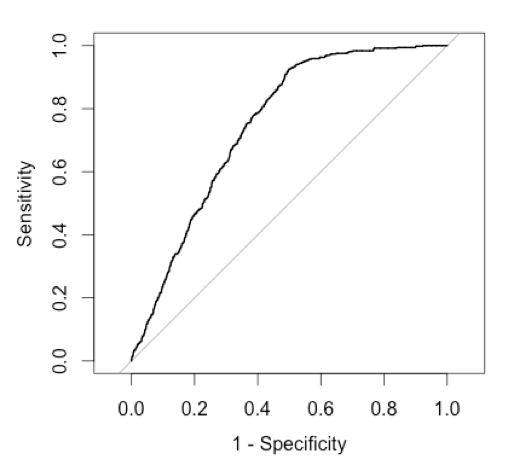
**1) GBM**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Specificity** | **Kappa** | **Trap.AUC** |
| **gbm\_smoteY\_varImpN\_Youd** | **0.5218052** | **0.4930271** | **0.8827160** | **0.09705213** | **0.6878716** |
| **gbm\_smoteY\_varImpN\_Top** | **0.6228537** | **0.6144381** | **0.7283951** | **0.11059345** | **0.6714166** |
| **gbm\_smoteY\_varImpY\_Youd** | **0.5594894** | **0.5378179** | **0.8312757** | **0.10282815** | **0.6845468** |
| **gbm\_smoteY\_varImpY\_Top** | **0.6213341** | **0.6113208** | **0.7469136** | **0.11458181** | **0.6791172** |
| **gbm\_smoteN\_varImpN\_Youd** | **0.5646558** | **0.5397867** | **0.8765432** | **0.11568289** | **0.7081650** |
| **gbm\_smoteN\_varImpN\_Top** | **0.6293876** | **0.6175554** | **0.7777778** | **0.12733482** | **0.6976666** |
| **gbm\_smoteN\_varImpY\_Youd** | **0.5529555** | **0.5264971** | **0.8847737** | **0.11177765** | **0.7056354** |
| **gbm\_smoteN\_varImpY\_Top** | **0.6435192** | **0.6354389** | **0.7448560** | **0.12734531** | **0.6901474** |

**2) AdaBoost**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **Sensitivity** | **Specificity** | **Kappa** | **Trap.AUC** |
| **ada\_smoteY\_varImpN\_Youd** | **0.5351770** | **0.5095980** | **0.8559671** | **0.09713105** | **0.6827826** |
| **ada\_smoteY\_varImpN\_Top** | **0.6372892** | **0.6369155** | **0.6419753** | **0.09516936** | **0.6394454** |
| **ada\_smoteY\_varImpY\_Youd** | **0.5362407** | **0.5102543** | **0.8621399** | **0.09897045** | **0.6595283** |
| **ada\_smoteY\_varImpY\_Top** | **0.5973256** | **0.5865463** | **0.7325103** | **0.09778658** | **0.6595283** |
| **ada\_smoteN\_varImpN\_Youd** | **0.5108646** | **0.4785890** | **0.9156379** | **0.09930087** | **0.6971134** |
| **ada\_smoteN\_varImpN\_Top** | **0.6646406** | **0.6623462** | **0.6934156** | **0.12672343** | **0.6778809** |
| **ada\_smoteN\_varImpY\_Youd** | **0.5208935** | **0.4899098** | **0.9094650** | **0.10235506** | **0.6996874** |
| **ada\_smoteN\_varImpY\_Top** | **0.6351618** | **0.6273995** | **0.7325103** | **0.11889830** | **0.6799549** |

In the above table, there are AUC obtained by using eight combinations of each algorithm, accuracy, sensitivity specificity, kapp and trapezoid method.  The formula of the Trapezoid method of AUC is equal to specificity\*(1-sensitivity))/2 + (sensitivity\*(1-specificity))/2 + (sensitivity\*specificity)). I chose a model based primarily on the value of the trapezoidal AUC, and the table shows that 70.8% of the AUC is the highest of the 16 models. So I chose a model that uses GBM for model learning. This model did not balance the class with SMOTE, did not extract variable variables, and used the Youden method in post-processing.  The ROC Curve is shown below for that specific model.

****

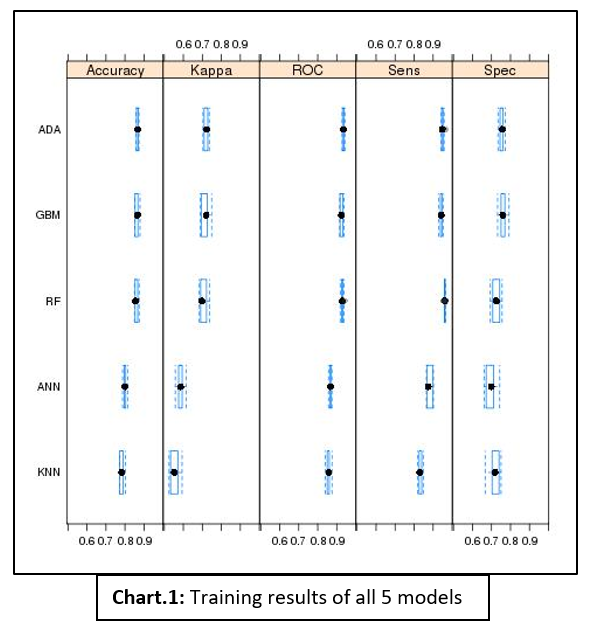
**5. Discussion and Conclusion**

**5.1 Trapezoidal method**

As previously described in section 4.4, our group utilized the trapazoidal area-under curve (AUC) to evaluate the models selected in our final outcome prediction. The formula is specificity\*(1-sensitivity))/2 +  (sensitivity\*(1-specificity))/2 + (sensitivity\*specificity)); this was inferred by summing the area of two triangles and a square together. This is different to the ‘standard’ AUC method, which takes the complete area under the ROC curve.

**5.2 Churn**

For predicting churn, I chose the adaboost model to produce the predicted output variable. This was due to the model’s higher ability to classify the negative class value ‘churn’, without sacrificing sensitivity nor accuracy. **Chart.1** shows how well each model’s training performed, defined by the five values of interest. Moving forward with this model I would allot enough time to implement the tuning grid, so that the most optimal parameters can be found and utilized. Additionally, postprocessing methods, such as top-left and Youden, would be implemented for further model optimization. This project was difficult due to the necessary preprocessing methods that needed to be implemented; there were many missing values and approaches to be utilized for this task. Also, the class imbalance proved to be difficult for the testing of models; each model struggled in correctly classifying the negative class value and this could be primarily due to the biased distribution of the outcome variable.



**5.3 Appentency**

Among the several models attempted for appetency prediction the one that was ultimately selected was a logistic regression. The GBM model that underwent tuning was surprisingly low performance. This may be due to pre-processing or SMOTE overfitting. The model selected for submission has a low Kappa-statistics nonetheless, which makes it unstable; moreover, even if 0.63 AUC was the best it is still rather low.

**5.4 Upselling**

As shown in the Upselling results table, I chose the GBM model as a Youden post-processing method without matching the data balance. However, even with high trapezoidal AUCs, we can see that the kappa values ​​and accuracy are low. The Kappa value is too low, meaning that the stability of the model is very low as well. I think the biggest problem in this analysis is preprocessing. I would have been better off using feature extraction methods and change data characteristics, rather than selecting importance features.

**References**

[1] Blog, S. I. G. K. D. D. KDD Cup 2009 : Customer relationship prediction. Retrieved from<http://www.kdd.org/kdd-cup/view/kdd-cup-2009/Intro>

[2] Kuhn, M. (2017, September 4). The caret Package. Retrieved from<http://topepo.github.io/caret/pre-processing.html#imputation>

[3] caret. Retrieved from https://www.rdocumentation.org/packages/caret/versions/6.0-79/topics/avNNet