ISYE 7406 Homework 6

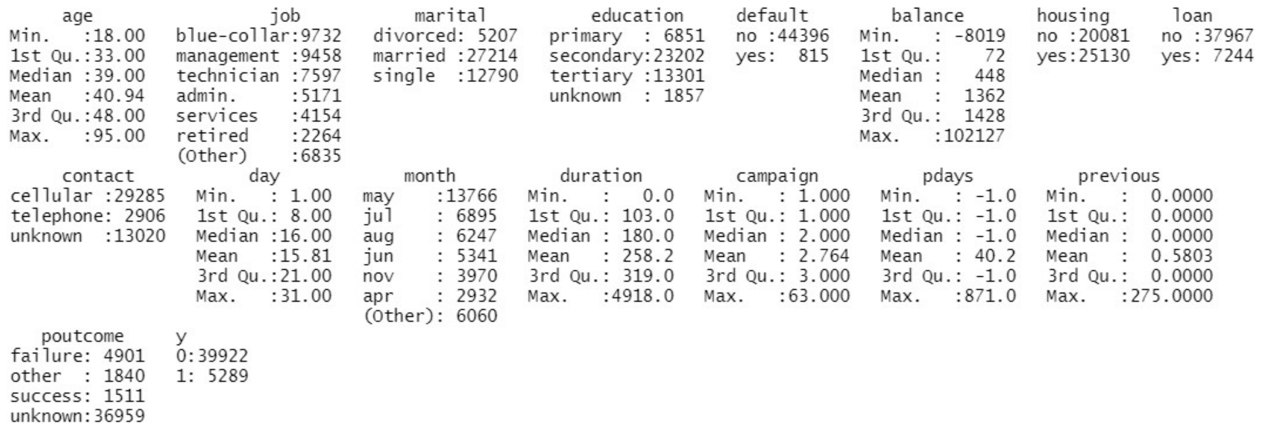
October 31, 2021

Introduction

This report will review the Bank Marketing Data Set retrieved from the University of Irvine machine learning repository which includes the customer's age, job, marital status, education, default status, housing, loan status, and 13 other predictor variables that can all be explained in more detail in the attribute information section of where the data lives. The purpose of analyzing the dataset is to generate Random Forests and Boosting with the classification goal being to predict if a client will subscribe (yes/no) to a term deposit (which is variable y). I will also identify the accuracy of the results compared to simple baseline methods like Logistic Regression and Naïve Bayes. The examination of the models will be accomplished through choosing certain tuning parameters based only on the training set through cross-validation. This will be evaluated by comparing Test errors of each model with and without Monte Carlo cross-validation to determine which method performed best. We also provide an analysis on whether it's a fair comparison between the models. The purpose of this report is to also identify other findings and visually explore the dataset for important/unusual patterns.

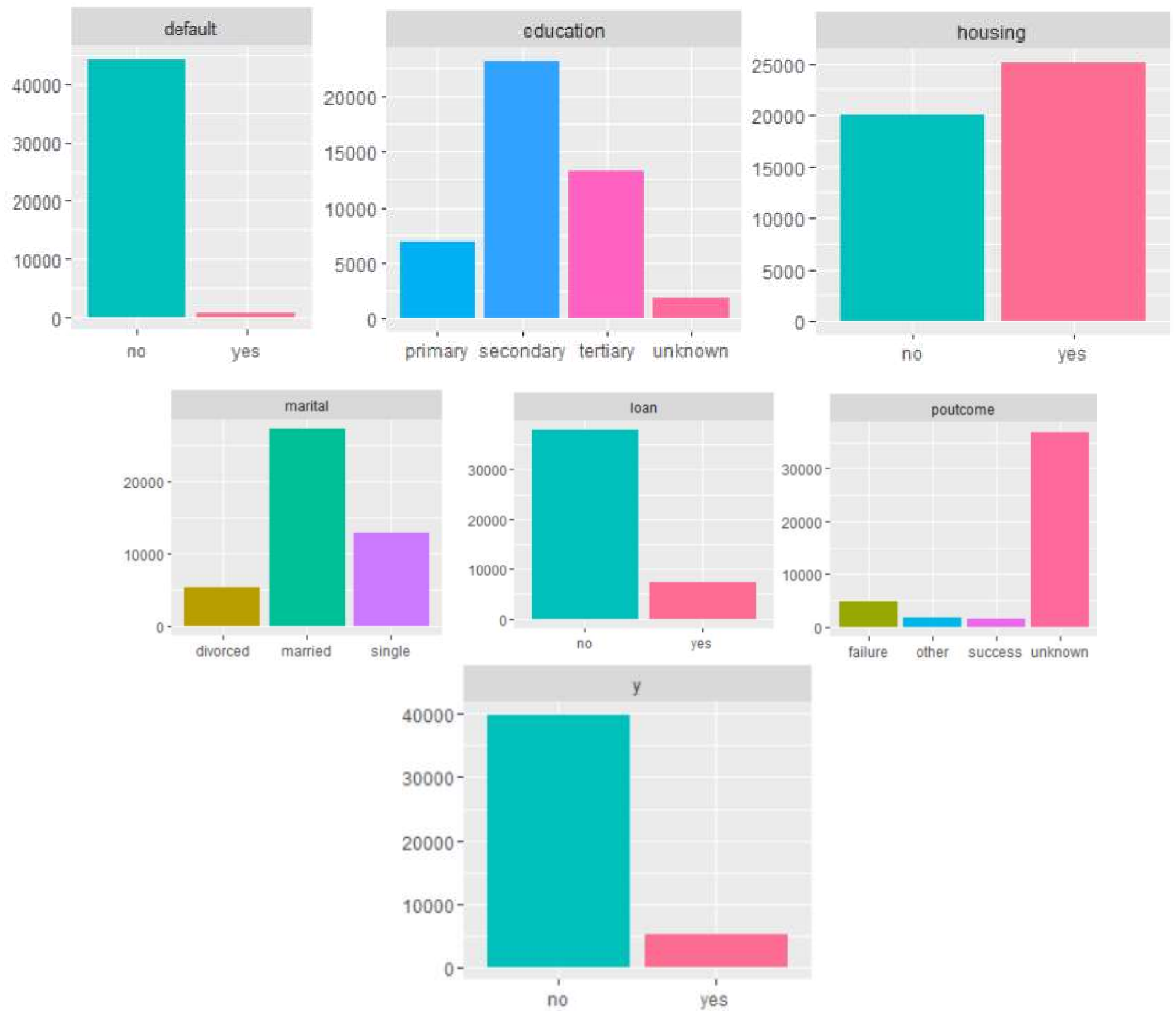
Exploratory Data Analysis

The exploratory data analysis will check for important/unusual patterns in the data through plots and summary statistics. According to the summary statistics shown below, it was interesting to see how only 13% invested in a Term Deposit suggesting that most of the customers (~87%) chose to not subscribe to a Term Deposit. An example of a Term deposit is a Certificate of Deposit (also known as a CD) where money is deposited and cannot be pulled out until the "term" (time to maturity) is reached. It's also worth noting that the age distribution is right tailed with most customers in their 30's to 50's. Note: The 0 and 1's in the "y" variable represent No and Yes respectively.

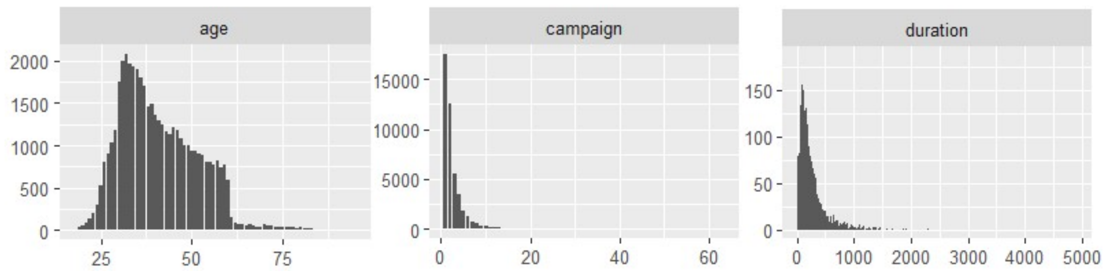


```
      age                 job            marital          education      default        balance         housing      loan
 Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851   no :44396   Min.   : -8019   no :20081   no :37967
 1st Qu.:33.00   management :9458   married :27214   secondary:23202   yes:  815   1st Qu.:    72   yes:25130   yes: 7244
 Median :39.00   technician :7597   single  :12790   tertiary :13301               Median :   448
 Mean   :40.94   admin.     :5171                    unknown  : 1857               Mean   :  1362
 3rd Qu.:48.00   services   :4154                                                  3rd Qu.:  1428
 Max.   :95.00   retired    :2264                                                  Max.   :102127
                 (Other)    :6835
      contact            day            month          duration         campaign          pdays          previous
 cellular :29285   Min.   : 1.00   may    :13766   Min.   :   0.0   Min.   : 1.000   Min.   : -1.0   Min.   :  0.0000
 telephone: 2906   1st Qu.: 8.00   jul    : 6895   1st Qu.: 103.0   1st Qu.: 1.000   1st Qu.: -1.0   1st Qu.:  0.0000
 unknown  :13020   Median :16.00   aug    : 6247   Median : 180.0   Median : 2.000   Median : -1.0   Median :  0.0000
                   Mean   :15.81   jun    : 5341   Mean   : 258.2   Mean   : 2.764   Mean   : 40.2   Mean   :  0.5803
                   3rd Qu.:21.00   nov    : 3970   3rd Qu.: 319.0   3rd Qu.: 3.000   3rd Qu.: -1.0   3rd Qu.:  0.0000
                   Max.   :31.00   apr    : 2932   Max.   :4918.0   Max.   :63.000   Max.   :871.0   Max.   :275.0000
                                   (Other): 6060
    poutcome      y
 failure: 4901   0:39922
 other  : 1840   1: 5289
 success: 1511
 unknown:36959
```

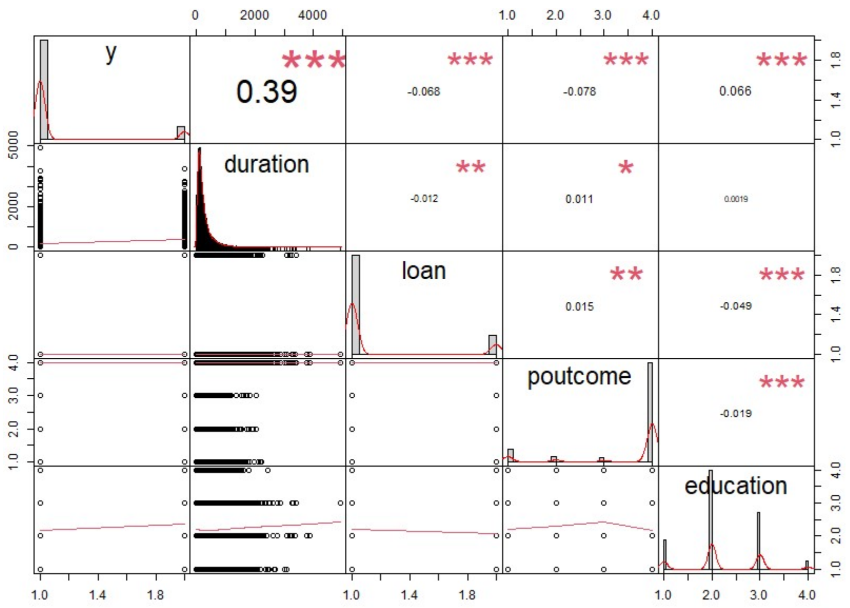
Based on the distributions of the “factor” data (shown below), we can see that the customers are mostly secondary educated (high school), no record of credit default, and has some type of housing loan which makes sense since most customers are already married. However, most customers don’t have a personal loan which has a similar distribution with the y response variable (subscribed to a term deposit). Now what’s important to identify in the data is the poutcome which is the outcome of the previous marketing campaign. You can see that although Failure ranks highest amongst Success and Other, these are much smaller than the value Unknown (~81%). Again, the bank’s ultimate goal is to find out who will subscribe (yes/no) to a term deposit and we can now see that they are most likely trying to solve this equation because their past marketing campaigns have been extremely unsuccessful due to lack of response from their clients (assuming lack of response equals unknown). This most likely has also led to the dismal personal loan and term-deposit performance.

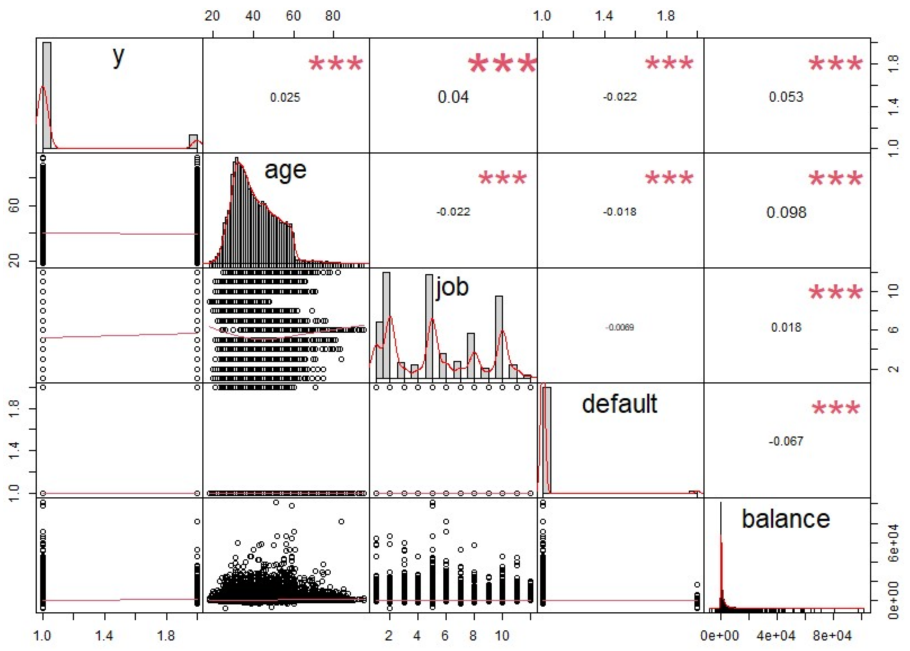


Based on the histograms of the "numeric" data (shown below), we can see that the customers as we stated before are mainly in their 30's-50's. The campaign values represent the number of contacts performed during this campaign and for a specific client. This tells me that campaigns are contacting less people over time (campaign after campaign). Duration (amount of seconds the call was made) also shows that calls are generally very short.



I decided to run a correlation test off a few variables that I thought were worth analyzing since running all the predictor variables at one time would be too taxing on the system (45K+ rows). I was pretty surprised to see that most of the variables I chose didn't have very much correlation with each other. I even created another correlation set (next page) to further verify whether the "y" response variable had any correlation to other predictor variables but it did not.



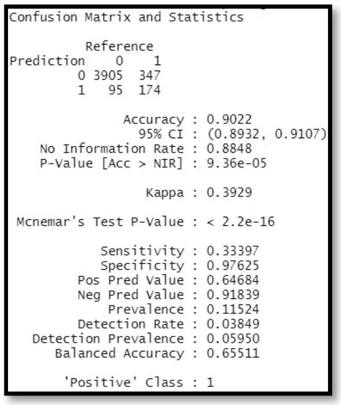


Methodology – Modeling for Training and Test errors

I performed the classification models (Random Forest, Boosting, Logistic Regression and Naïve Bayes) to retrieve training and test errors when predicting "y" (Yes or No to term deposit) as well as to compare the models. The testing data is used to evaluate the model's true performance since the training data may have excluded/included certain data that may have a significant positive/negative affect on the training error. I will also explain why I chose certain tuning parameters (especially for Logistic Regression and Naïve Bayes) based on the training set through Cross-Validation. For this exercise, I performed a 100-run cross validation for the model to obtain the test errors and observed which one had the best test error. Running a Monte Carlo cross-validation helps reduce the variance compared to just running one test run and depicts which model is truly the most accurate on a large scale.

Results & Findings

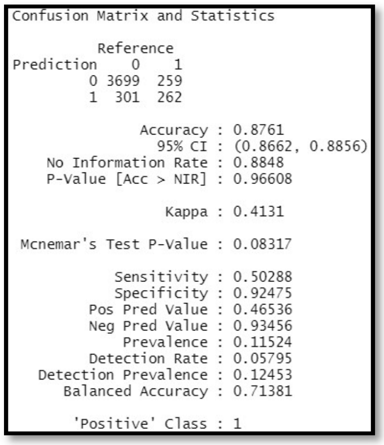
Logistic Regression – For the Logistic Regression model, I wanted to use the intuitive approach where (cutoff value) c*=0.5 but because the proportion of 0's to 1's is significantly uneven in the training data, I decided to compute c* based on the proportion of Y=0 in the training data set which was 0.87. This approach was explained in Dr. Yajun Mei's lecture notes (*5.1.2*, pg.10). I then ran the Logistic Regression model with all predictors and got an initial training error of 0.1125857. The cross-validation test error resulted at 0.110408 for the model and I thought this was very odd because these test errors seemed awfully large so when I attempted the same procedure with c* = 0.50, it actually came out with a better test error at 0.09776598 and cross-validation test error of 0.09907692. This made me think whether or not I had chosen the correct predictor variables but I decided to leave it as is since the correlation matrix did not provide any clear significant variables with a value of >= +/- 0.50 a significant factor which is considered "significantly correlated" based on a few peer reviewed articles.



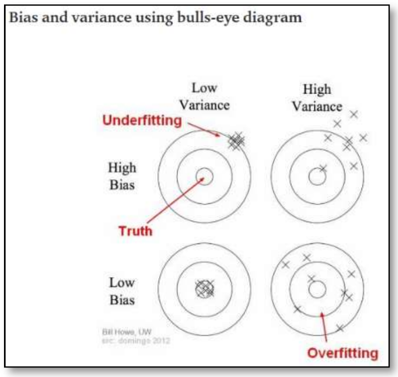
```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3905  347
         1   95  174

               Accuracy : 0.9022
                 95% CI : (0.8932, 0.9107)
    No Information Rate : 0.8848
    P-Value [Acc > NIR] : 9.36e-05

                  Kappa : 0.3929

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.33397
            Specificity : 0.97625
         Pos Pred Value : 0.64684
         Neg Pred Value : 0.91839
             Prevalence : 0.11524
         Detection Rate : 0.03849
   Detection Prevalence : 0.05950
      Balanced Accuracy : 0.65511

       'Positive' Class : 1
```

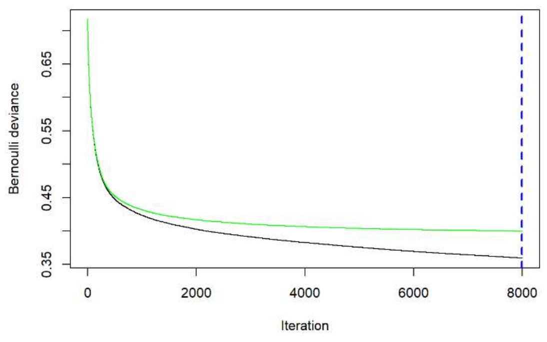
Although the accuracy of the model with the c*=0.50 resulted at about 90.22%, I believe the model is generally terrible for this particular data-set due to the poor Sensitivity score. The Specificity had a great score of 97.62%, and the Sensitivity did extremely poorly at 33.39%. A low Sensitivity means that there are many FALSE negative results meaning more cases of "no" being misclassified as "yes" (Type I error). Specificity being higher shows that the model tends give less FALSE positive results meaning less cases of the model stating "yes" when the true value is "no" (Type II error). As Dr. Yajun Mei explained before, for a data-set that classifies cancer vs. no-cancer, this would be a terrible model since the model would have doctors classify their patients as negative when they are actually cancer positive. Similarly, this model would tell Financial Banking Analysts to not call people who would actually be interested in subscribing to a term deposit which is the main reason for the model. Having a high Specificity is great so that the callers don't waste time with calling unlikely potential term deposit clients, but it doesn't solve the overall problem so I think this model doesn't provide value to the bank.

Naïve Bayes – For the Naïve Bayes model, there wasn't anything I needed to tune/modify in the parameters since all get passed through the model. I then ran the Naïve Bayes model with all predictors and got an initial training error of 0.1229568 which was worse than the Logistic Regression model. I ran the prediction off the test set as well and it came out to 0.1238664. Now unfortunately because the dataset contains (45K+ rows), I had to run the cross-validation off the test data (4K+ rows). However, it's worth mentioning that this model is extremely slow for cross-validation and this is a huge factor when selecting models (everyone hates a slow program). The cross-validation test error came out to 0.1432798 which was a lot worse compared to the Logistic Regression model. Similar to the Logistic Regression model, the Specificity performed well at 92.48% but the Sensitivity performed poorly at 50.29% but not as bad compared to the Logistic Regression model. We've learned that Naïve Bayes has a higher bias but lower variance compared to Logistic Regression which means that the model pays small attention to the training data (oversimplifies the model) which leads to high errors on training and test data (cited here) thus underfitting.



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3699  259
         1  301  262

               Accuracy : 0.8761
                 95% CI : (0.8662, 0.8856)
    No Information Rate : 0.8848
    P-Value [Acc > NIR] : 0.96608

                  Kappa : 0.4131

 Mcnemar's Test P-Value : 0.08317

            Sensitivity : 0.50288
            Specificity : 0.92475
         Pos Pred Value : 0.46536
         Neg Pred Value : 0.93456
             Prevalence : 0.11524
         Detection Rate : 0.05795
   Detection Prevalence : 0.12453
      Balanced Accuracy : 0.71381

       'Positive' Class : 1
```

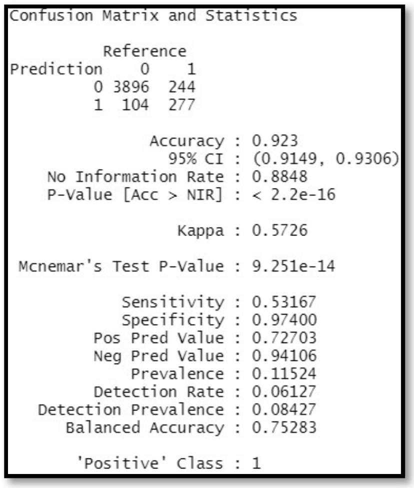




```
perf_gbm1
```

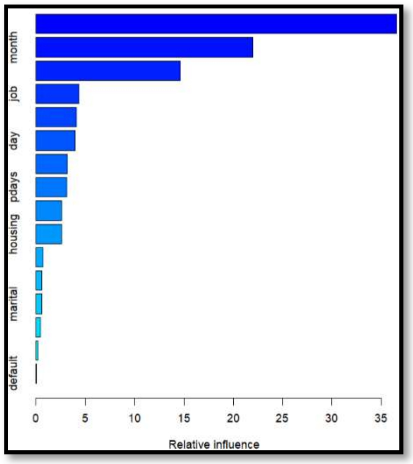
```
## [1] 7993
```

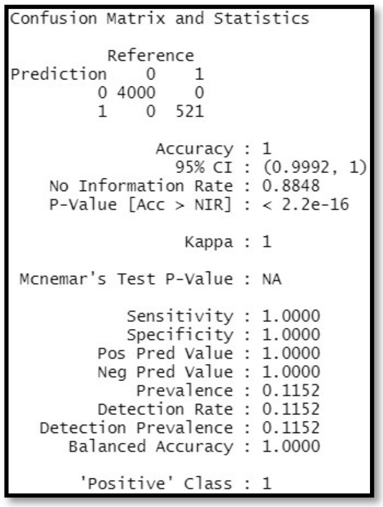
<u>Boosting</u>– The Boosting (ensemble | Re-weighting) model creates trees sequentially with each new tree built to correct mis-classification from previous grown trees with the weighted average of the trees <u>improving the training error</u>. In terms of processing speed, the model was extremely slow to run with *n.trees* in the upper thousands (5000+) (~15-30 minutes). However, the upside to the long queue time was achieving a training error of 0.07679547 and a test error of 0.07697412. In terms of the parameters, I chose a high *n.trees (8000)* because I wanted to be able to estimate the optimal number of iterations based on a large number so I can get the most accurate *n.trees.* I then used the optimal *n.trees (7993)* when retrieving the prediction values on the training and test errors.



```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3896  244
         1  104  277

               Accuracy : 0.923
                 95% CI : (0.9149, 0.9306)
    No Information Rate : 0.8848
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.5726

 Mcnemar's Test P-Value : 9.251e-14

            Sensitivity : 0.53167
            Specificity : 0.97400
         Pos Pred Value : 0.72703
         Neg Pred Value : 0.94106
             Prevalence : 0.11524
         Detection Rate : 0.06127
   Detection Prevalence : 0.08427
      Balanced Accuracy : 0.75283

       'Positive' Class : 1
```

Although the testing error performed better than the Logistic Regression and Naïve Bayes model, the Sensitivity problem still occurred at 53.17%. Below you can also see the relative influence of the model with most of the influence being the *month* variable. After running this plot, it may have been worth plotting the month variable in the correlation plot to see how it correlated with the *y* response variable.





```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4000    0
         1    0  521

               Accuracy : 1
                 95% CI : (0.9992, 1)
    No Information Rate : 0.8848
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.1152
         Detection Rate : 0.1152
   Detection Prevalence : 0.1152
      Balanced Accuracy : 1.0000

       'Positive' Class : 1
```

Random Forest– The Random Forest (ensemble | Re-sampling) model creates many trees in parallel, with each tree built on a bootstrap data set, independent of the other trees and uses the mean of these trees to reduce the variance. In terms of processing speed, this model was similar to the Boosting model where running a single model took a few minutes but running the cross-validation model took significantly longer due to the large dataset. However, I couldn't believe my eyes when I observed that the testing error resulted with a perfect 0.00 error (shown on the confusion matrix to the left) meaning it provided a perfect classification. This was done by creating the model on the training set and running the prediction on the test set. Now based on the homework instructions, I decided to again run the cross-validation on just the training data and create the test set based on the training data instead of using the provided test set. It was interesting to see that the cross-validation resulted with a (~0.09) test error especially after just completing a perfect classification based on the single-run model. I had a hypothesis that splitting the training data could have caused the model to lose valuable information from the split so I decided to also run the cross-validation off the full training data to create the model and run the prediction off the provided test set from the UCI repository (business as usual). Because of the large dataset, I was tempted to run the cross-validation with 10 loops but out of sheer curiosity, I decided to perform the usual 100-run (took 2 hours to process) but I'm sure a 10-run loop would be close since my belief is that the variance should be extremely small (especially when the single-run model ran perfectly). The 100-run cross-validation resulted in a test error of only 3.98142e-05 which is exceptionally small and a significantly smaller variance of 8.282659e-09 which tells me that running a cross-validation of 10 would have been sufficient. This nearly perfect classification confirms my theory that running the cross-validation using the full training data to create the model and running the prediction off the test data set is needed since splitting the training data must have excluded valuable rows in the model. It's worth mentioning that out of the 100 runs, 16 runs had a test error of 0.00022119 (1 value misclassified) and only 1 run had a test error of 0.00044238 (2 values misclassified) with the other 83 runs having perfect classification. For reporting processing purposes, I'm reluctant to knit the R code with 100-runs (having to wait another 2 hours) and instead performed a 10-run cross-validation in its place since the variance was significantly small. By no surprise, the test error resulted at 0.000176952 with a variance of 5.218668e-08 which still presents the same conclusion.

Conclusion

When analyzing all the cross-validated models, it is clear that Boosting performs the best when tuning parameters like *n.trees* while running solely on training and testing the model from the training data with exception of Naïve Bayes (specifically used test data due to poor processing speed). However, the Random Forest model is the clear winner when creating the model using the full training set and testing the prediction with the separate test data set. This a great example of how important test data is for model testing since without it, we would of assumed Boosting was the winner (using “winner” loosely due to poor Sensitivity %). We can also see how overall the baseline methods performed worse than the ensemble methods. I also noticed that the results were extremely close amongst most of the models so I decided to run different seeds to get a better understanding of the average performance. I’ve already started using this exact data set for my project and will dive in deeper (especially in Random Forest) on how combining the training and test data to create the model will impact the overall accuracy and explain in more detail the pros and cons each model provides. I hope I can re-attempt this type of problem set for my own work and hopefully will be able to better understand each classification model’s strengths in a personal work-based application but I won’t hold my breath for the same accuracy score as the one shown here.