Classification and Prediction on Bank Marketing Dataset

1 Primary Goal

1.1 Background/Dataset Description

In this project, our goal will be to use classification to predict whether a client will sign on to a long-term deposit or not. By using a predictive model, we can answer our primary question of interest of if the client will sign on to a long-term deposit. The question we want to answer is a binary classification problem, because the response variable is categorical with classes either "yes" or "no". The original research paper using this dataset can be found here. [1] Using bank data from a telemarketing campaign in 2008, we can analyze and explore the relationship between different variables to recognize patterns and draw conclusions. This will allow improvement to strategies for marketing campaigns in the future.

1.2 Statistical Questions of Interest

A statistical question of interest we have is "What features have the most significant influence on the response variable?". Our univariate analysis will help in identifying these features. We can further confirm this through the logistic regression model which helps in identifying which variables do not have a significant influence on customer response. Through the random forest model, we will also be able to identify which variables have the most influence in customer response from the variable importance plot.

1.3 Analysis Plan

To initialize the project, we will begin by cleaning and transforming the data. Then, we will make use of exploratory data analysis techniques and univariate analysis as a way to summarize key characteristics about the dataset and develop an overall sense of it. This includes quick visualizations to recognize patterns that appear within the data. This helps us in our classification as we will be able to recognize how variables have an influence on the clients response to the long-term deposit. However, this isn't conclusive evidence and should be confirmed through predictive modeling. We will implement the following classification methods:

Logistic Regression: We chose logistic regression, a generalized linear model, as our baseline model. Due to the underlying linearity in this method, it is simple and has good interpretability than other more complex and robust models. We will train a binary logsitic regression model to classify whether a sample will belong to class 1 or class 0. We will then use our trained model to predict whether new "unseen" samples will be classified as 0 or 1 based off estimated probabilities for each observation. If the observation is above a probability threshold, default $p = 0.5$, that observation is then predicted to be in class 1. We will use two performance metrics to analyze and compare each models performance of their prediction. We will create a confusion matrix consisting of the predicted classifications as the rows, and the actual, true classifications of our test data as the columns, and take the proportion of the sum of the diagonal out of the total number of observed predictions and actual, and this will be the classification accuracy of our model. However, classification accuracy can be biased if we have imbalanced data. In this dataset our response is binary. Our class 0 is the majority class, where as our class 1 is the minority class. So classification accuracy can skew the true accuracy of the class we are interested in, which is the minority vote in this case. Thus, we will also plot ROC Curves and compare AUC values.

Random Forest: For the random forest, we build a specific number of decision trees on our bootstrapped training data. What distinguishes random forest when building the decision trees, is that each time we consider a split, we carry out a random selection of m predictors from our set of predictors in our data.The split then employs one of those m

predictors. Then, we find the number of trees that we will be considering when we implement random forest, and our main objective is to minimize error. We will then compute the confusion matrix that will ultimately help us calculate the predictive accuracy of our model. In order to decide which model is more effective, we plan to calculate the classification accuracy of each model. To compare each models performance, we will plot the ROC curves and calculate the AUC to determine which model is best.

2 Statistical Analysis

2.1 Exploratory Data Analysis/Descriptive Data Analysis

The dataset consists of 41,188 (30,488 after cleaning) datapoints of client data which features 20 independent variables which will help in predicting whether the client will sign on to a long-term deposit. The variables are as follows:

Client Info: Age, job, marital, education, default status, housing, and loan
Campaign: Last contact type, last contact month of year, last contact day of the week, and last contact duration
Other: Number of contacts performed in current campaign, number of days that passed by after the client was last contacted, number of contacts performed before this campaign, outcome of previous campaign, and whether a client has subscribed a term deposit.

2.1.1 Univariate Analysis

Throughout this section we we will do univariate analysis by taking a feature and seeing how much it distinguishes between the two classes. However, this analysis is not conclusive and should be confirmed through predictive modeling. First, we take a look at the class distribution in Figure 1. We can see that the 'no' response is about 8 times greater than the 'yes' response creating class imbalance. XGBoost in this scenario is effective because it can deal with class imbalance.

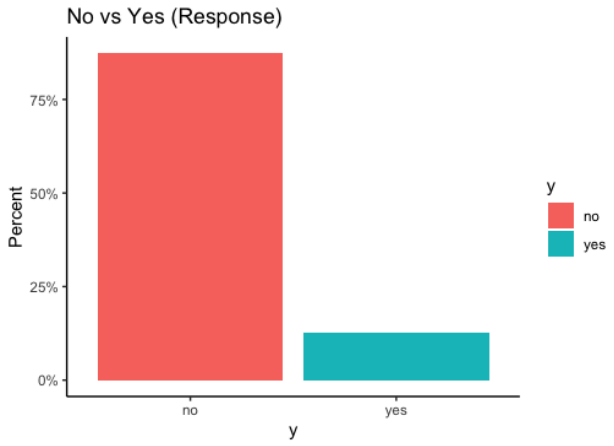


Figure 1: Percentage of Clients who Signed Long-Term Deposits

Age: Using Figure 2 and Figure 3, the histogram shows the age distribution to be mostly between 30-45 years along with the age boxplots showing similar results regardless of response. Here, we can recognize that age isn't a strong indicator of whether the customer will sign on to a long-term deposit or not.

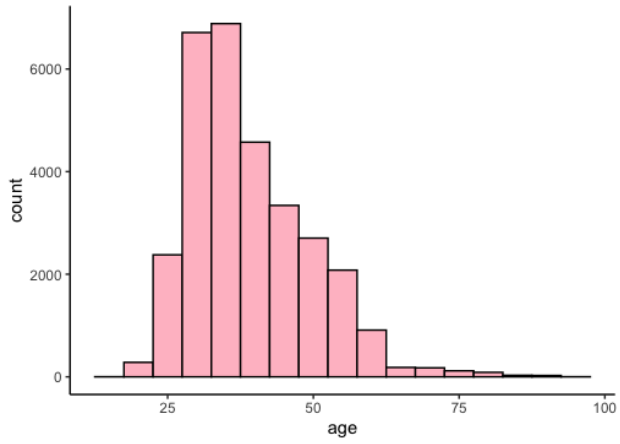


Figure 2: Histogram of Client Age

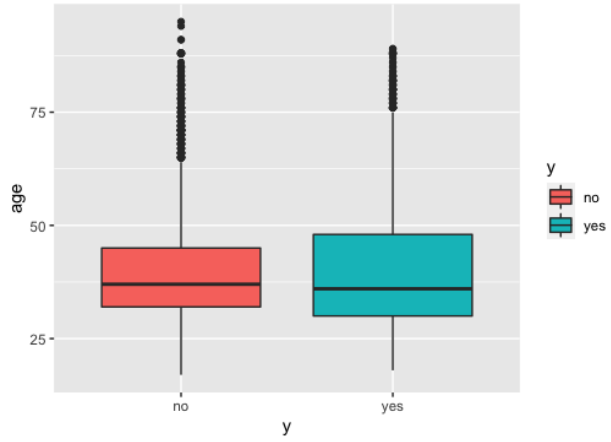


Figure 3: Boxplot of Client Age by Response

euribor3m: Euribor 3 month interest rate is a daily indicator that denotes the rate of interest used in lending between banks within the European Union Interbank Market. In Figure 4, the median differentiates greatly amongst both classes showing that it can be a very useful feature for determining whether a client will sign on to a long-term deposit or not. However, this isn't conclusive and applying of models is important.

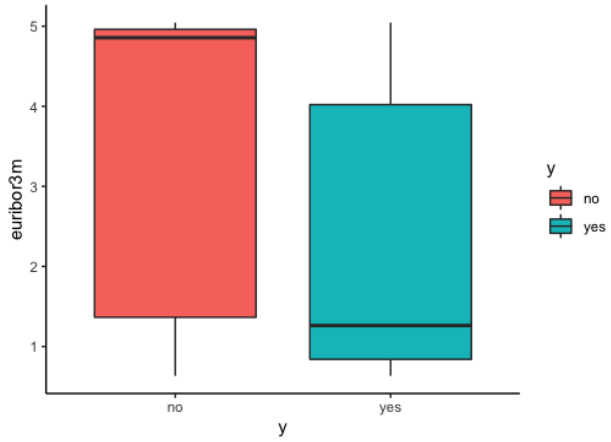


Figure 4: Boxplot of Euribor 3 Month Rate by Response

emp.var.rate: Employee variance rate is a quarterly indicator. In Figure 5, the distribution and medians between the two responses for emp.var.rate differ. We can guess that this feature can be useful in predicting client response.

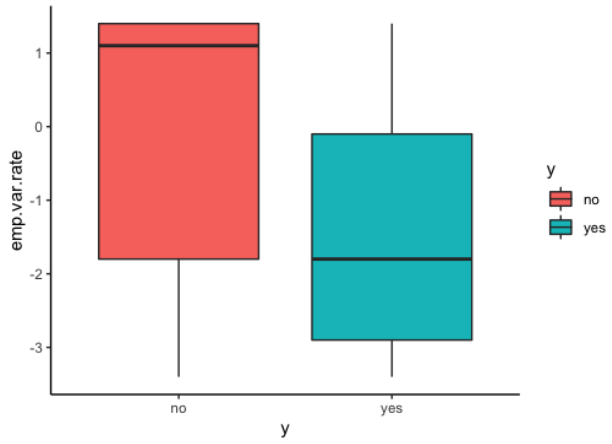


Figure 5: Boxplot of Employee Variation Rate

nr.employed Number of employees is a quarterly indicator. Similar to emp.var.rate, in Figure 6, the distribution and median between the two responses differ. We can guess that this feature can be useful in predicting client response.

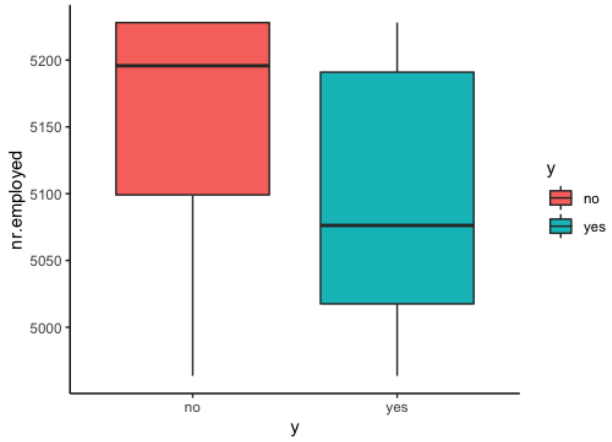


Figure 6: Boxplot of Number of Employees

Duration: Duration is the number of seconds spent on the last contact with the client. Duration fails to be a useful predictor because it can't be known how long the duration of the call will be beforehand, therefore duration is removed from the dataset.

Job: In Figure 7, the type of job of the clients is listed. In this figure, it seems that for both responses(yes or no), a job as an admin is the highest in both. This means that the admin job is simply the most common job amongst the clients. Further modeling will be needed to

determine the importance of this feature.

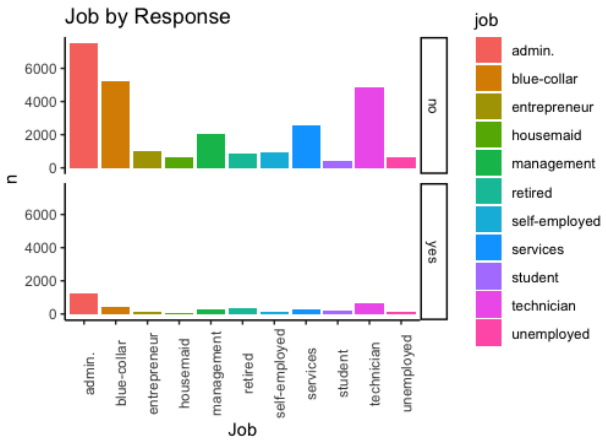


Figure 7: Job by Response

Marital Response: Lists the clients marital status. Figure 8 shows that majority of customers are married.

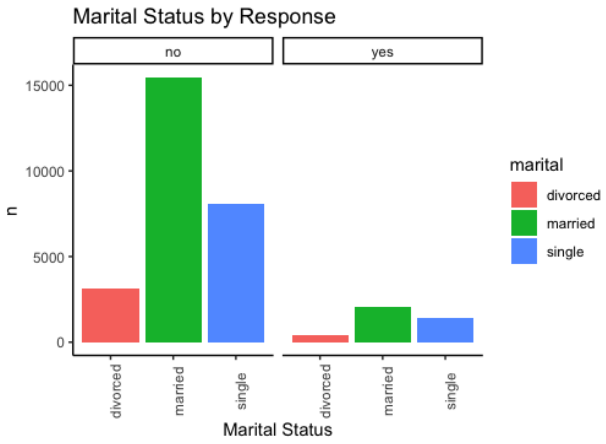


Figure 8: Marital Status by Response

Day of the Week: Denotes which day of the week the client was contacted. In Figure 9, between both responses, the distributions are similar. Therefore, it seems like it will not be an effective feature in determining a client's response. Further predictive modeling could prove otherwise.

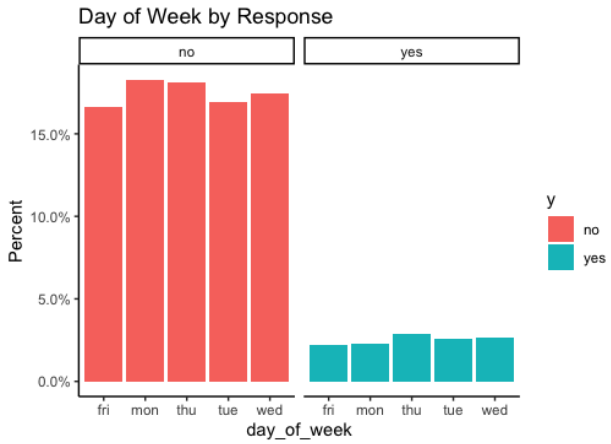


Figure 9: Day of the Week Contacted by Response

poutcome: Previous outcome of the last marketing campaign. In Figure 10, it is shown that regardless of the response, most customers are new customers and therefore didn't exist in the previous campaign. However, we can see that a majority of customers who subscribed in last years campaign, also subscribed for the term deposit. Therefore, this feature can help predict if a client will subscribe to the long-term deposit.

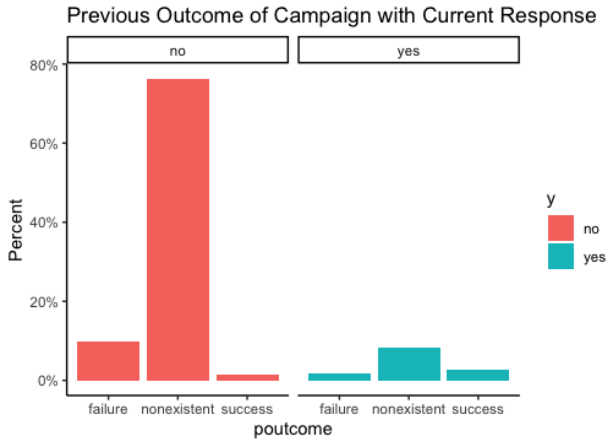


Figure 10: Previous Outcome with Response

Pearson Correlation: In Figure 11, we measure the Pearson Correlation of all the features. The more red, the more it is positively correlated whereas the more blue, the more negatively it is correlated. This figure helps in finding collinearity within our features.

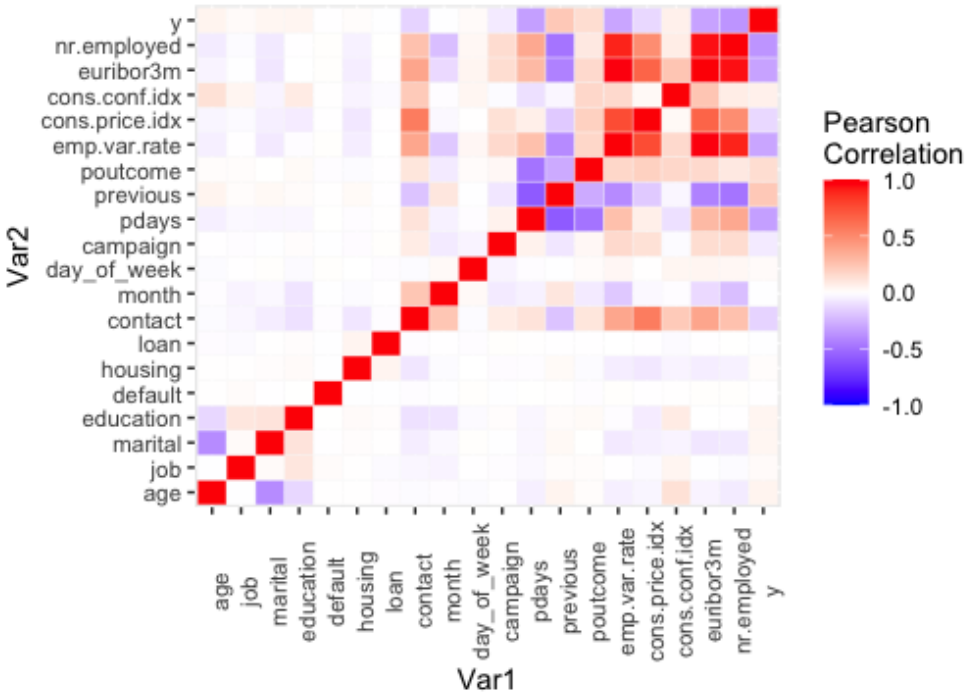


Figure 11: Pearson Correlation Heatmap

2.2 Data Preprocessing and Cleaning

2.2.1 Checking for NA/Missing/Unknown Values

This dataset does not contain any 'NA' values. If the row contained a 'NA', it would be removed from the dataset. Despite having no missing values, there are however 'unknown' values amongst the categorical variables. If any row contained an 'unknown' value, the row is removed. This removed a large amount of datapoints reducing the number of datapoints from $41,888$ to $30,488$.

2.2.2 Transforming Categorical Variables into Numerical Variables

Since there are 10 categorical variables, it is important to change them into numerical variables when using logistic regression.

2.2.3 Splitting Data Into Training/Test Sets

We apply a random 80/20 split of our original data set, which contains $30,488$ observations after data cleaning. In our train set, we observe $24,390$ samples which will be used in model building. In our test set, we observe $6,098$ samples which will be used to measure the performance of our trained models on "unseen" data.

3 Generalized Linear Model: Logistic Regression

3.1 Theory

In logistic regression, what we are interested in is the conditional probability that our response variable belongs to class 1, given a linear combination of features and their weights, or coefficients. The output of logistic regression function always depends on a set of linear inputs, like a linear regression model, hence logistic regression is considered a generalized linear model. The probability distribution of the response variable, y, is a binomial distribution, which is what the glm() function calculates. It does not give us predicted values, instead it gives us probability estimates for each observation.

Let β denote the vector of true parameters coefficients we want to estimate from our data length p such that $\beta = \beta_0, \beta_1, ..., \beta_p$. Similarly, let x denote the vector of features (predictor variables) from our data that we will use to predict the binary classifications of each sample, n such that $x = x_1, ..., x_p$. Let θ be the linear combination of predictor variables that will give us an estimated probability $f(\theta)$ that our response variable is classified as class 1, for each sample n, where $f(\theta) = \frac{e^\theta}{1+e^\theta}$ is the logistic function, or sigmoid function, that can take any value on the real line as input, but it's output is always a range between 0 and 1. The Axiom of Probability says that a probabilitity is defined to be between the values of 0 and 1, hence why $f(\theta)$ is used to model the estimated probabilities. Note that there will be n such estimated probabilities for each sample of our given data.

Then we have,

$P(Y = 1 \mid X) \approx f(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p) = f(\theta)$ for each n observations. With some algebraic manipulation, we equivalently model:

$\log_{(\frac{f(\theta)}{1-f(\theta)})} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$, note that $\beta_0 + \beta_1 x_1 + ... + \beta_p x_p = \theta$.

This is known as the log of odds, or logit function, which is our link function for the logistic regression algorithm. The second equation is the one that we model, where we give a linear combination of the predictor variables as input, and the algorithm returns the left hand side of the equation, the log of odds. Note that the log of odds does not directly This algorithm attempts to find the best estimates of our parameters of interest, β possible linear combination of θ from our given data.

Y is our binary categorical response variable that has levels "yes" or "no". Logistic regression cannot handle categorical variables, so we convert them into dummy numeric features, for the purposes of running our algorithm. In particular, note that the response Y is converted into a numerical variable so that it is either in class 1 for yes, or class 0 for no. After preprocessing and cleaning, our given data contains $n = 30488$ observations of the response Y, where each Y has a set of $p = 20$ parameters of interest, β, and a set of $p = 20$ predictor variables x. We randomly split our data set into 2 subsets, one containing 24390 training observations used to build our model and the other containing 6098 test observations used to evaluate the performance of our model on unseen data in an attempt to minimize overfitting. Our target class is class 1, so we want $Y = 1$. The logistic regression algorithm will estimate each β_i from the n true observations from the train set.

3.2 Model Building

3.2.1 Logistic Regression using all features

First, we will run a logistic regression model using all 20 features in the dataset. We use the generalized linear model function, glm(), from the R base package, and specify the family to be binomial for a logistic regression to be done.

We can see that many coefficient estimates were not significant. The predictor age for example, was not particularly useful probabilities because the mean age was 39, and a majority of samples were in the age group of $30 - 40$, hence not a very useful predictor. Since the glm function returns the log of odds, or logit, the coefficient estimates don't tell us much about the class predictions. The relationship between the predictor variables and

the response is not directly linear, it is only linear by it's link function, log of odds. This implies that a unit increase in a predictor variable will change the log of odds of the response, hence the estimates only give us insight to the log of the odds. For example, the coefficient estimate for the predictor variable "default" is $\hat{\beta}_5 = -7.43$. What this implies is for a unit increase in the predictor "default", with all other predictor variables held constant, the decrease in the log of odds that our response variable is classified as 1 is -8.79. We can also see that the p-values of the coefficient estimates of contact, month, dayofweek, pdays, poutcome, emp.var.rate, euribor3m, and nr.employed predictor variables are statistically significant with an alpha of 0%, but this only tells us about the significance of the log odds. Furthermore, a statistically significant coefficient estimate does not gurantee those variables will help our classification accuracy, so we will not interpret them for logistic regression.

To measure the performance of our full logistic regression model, we can look at the Akaike Information Criteria (AIC) value. The AIC value is the residual deviance adjusted for the number of parameters used in the model and gives us some intuition of model fit. Unlike the R-squared value in linear regression, a low AIC value is ideal. Note that AIC of the full model is 11587. However, this does not measure much performance of our model directly, it is rather a measure of performance for comparing and selecting models.

```
Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     2.31e+01   2.44e+01    0.95  0.34384
age             2.96e-03   2.34e-03    1.27  0.20505
job             1.26e-02   6.76e-03    1.86  0.06327 .
marital         9.22e-02   4.40e-02    2.10  0.03611 *
education       5.32e-02   1.27e-02    4.19  2.7e-05 ***
default        -7.43e+00   1.39e+02   -0.05  0.95742
housing        -3.93e-02   4.93e-02   -0.80  0.42540
loan           -7.95e-02   6.83e-02   -1.16  0.24489
contact        -6.62e-01   7.85e-02   -8.43  < 2e-16 ***
month          -1.15e-01   1.14e-02  -10.09  < 2e-16 ***
day_of_week     6.69e-02   1.78e-02    3.76  0.00017 ***
duration        4.38e-03   9.09e-05   48.22  < 2e-16 ***
campaign       -3.47e-02   1.44e-02   -2.41  0.01579 *
pdays          -9.34e-04   1.88e-04   -4.98  6.5e-07 ***
previous       -3.61e-02   6.61e-02   -0.55  0.58424
poutcome        4.82e-01   9.06e-02    5.32  1.0e-07 ***
emp.var.rate   -9.48e-01   8.16e-02  -11.62  < 2e-16 ***
cons.price.idx  5.66e-01   1.51e-01    3.76  0.00017 ***
cons.conf.idx   5.17e-03   8.24e-03    0.63  0.53030
euribor3m       8.05e-01   1.25e-01    6.42  1.3e-10 ***
nr.employed    -1.59e-02   2.29e-03   -6.92  4.5e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 18597  on 24389  degrees of freedom
Residual deviance: 11545  on 24369  degrees of freedom
```

```
AIC: 11587

Number of Fisher Scoring iterations: 10
```

We then use our trained model to estimate probabilities on the test set, our model's unseen data. We get returned, a vector containing the probability estimates for each observation in the test set belonging to class 1. We then set a probability threshold of 0.5, so for each predicted probability, if it is greater than 0.5 the sample will be classified as class 1, otherwise it will be classified as class 0.

```
## Confusion Matrix
          actual
predicted    0    1
        0 5211  453
        1  133  301
```

Summing up the true positives and the true negatives on the main diagonal, we get the proportion of samples that were correctly classified. Our classification accuracy for logistic regression using all predictor variables is: 90.4%, which is fairly high accuracy. However, classification accuracy may not be the best measure of performance because we have imbalanced classes, so our accuracy rate could be optimistically high due to the majority class votes of "no". Thus, we plot an ROC curve and calculate AUC value. True Positive Rate (TPR), or Sensitivity, 1− False Negative Rate, indicates how many samples were correctly classified as class 1, out of all the true class 1 observations. Our True Positive Rate is 39.92%. False Positive Rate (FPR), or 1−Specificity, indicates how many samples were incorrectly classified as class 1, when their true observation is class 0, out of all the true class 0 observations. Our False Positive Rate is 2.49%.

Plotting the ROC Curve on the full model, we can an AUC of 92.7% which our model had a strong ability to classify samples between classes.

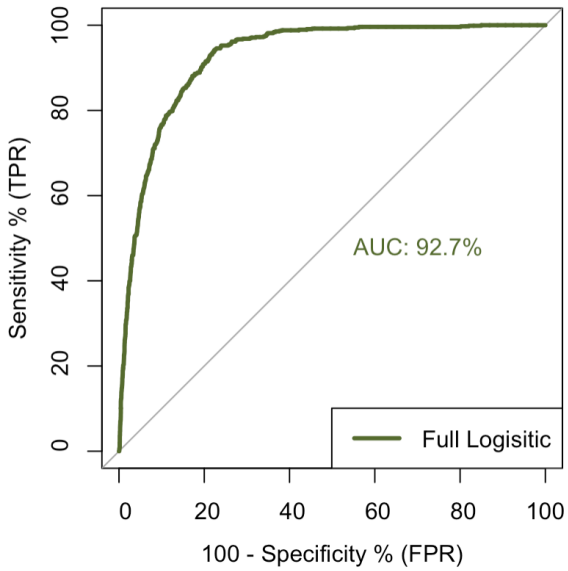


Figure 12: ROC Curve for Logistic Regression with full set of features

3.3 Model Selection

For model selection, we will perform Stepwise Regression Algorithm using the stepAIC() function. It takes our full glm model with all predictor variables, and runs a logistic regression step by step removing a predictor variable each time (backward regression), until it reaches the null model. Finding the lowest AIC value helps avoid overfitting when training our model, because AIC penalizes the model when adding more parameters (coefficients) [2]. The stepwise algorithm chose the predictor variables: job, marital, education, contact, month, daysofweek, duration, campaign, pdays, poutcome, emp.var.rate, cons.price.idx, euribor3m, and nr.employed as the most significant. It removed 6 variables: age, default, housing, loan, previous, and cons.conf.idx because these variables did not contribute much at all to lowering the AIC. The result is the combination of predictor variables that gave gives the lowest AIC of 11579, compared to the full model, with an AIC of 11587, the AIC value did not differ significantly.

```
Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     3.73e+01   1.56e+01    2.39  0.01696 *
job             1.24e-02   6.75e-03    1.84  0.06595 .
marital         6.75e-02   3.93e-02    1.72  0.08559 .
education       5.02e-02   1.25e-02    4.01  6.1e-05 ***
contact        -6.42e-01   7.38e-02   -8.69  < 2e-16 ***
month          -1.19e-01   1.02e-02  -11.64  < 2e-16 ***
day_of_week     6.65e-02   1.78e-02    3.74  0.00018 ***
duration        4.38e-03   9.09e-05   48.25  < 2e-16 ***
campaign       -3.43e-02   1.44e-02   -2.39  0.01705 *
pdays          -8.59e-04   1.08e-04   -7.97  1.6e-15 ***
poutcome        5.15e-01   6.37e-02    8.08  6.4e-16 ***
emp.var.rate   -9.55e-01   8.15e-02  -11.72  < 2e-16 ***
cons.price.idx  4.88e-01   1.09e-01    4.49  7.0e-06 ***
euribor3m       8.84e-01   7.42e-02   11.91  < 2e-16 ***
nr.employed    -1.73e-02   1.33e-03  -13.01  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 18597  on 24389  degrees of freedom
Residual deviance: 11549  on 24375  degrees of freedom
AIC: 11579

Number of Fisher Scoring iterations: 6
```

4 Model Performance

4.1 Logistic Regression using a reduced feature set

Now we will calculate classification accuracy on the reduced model and AUC value. Rounded to the first decimal place, the classification accuracy rate achieved from the reduced model and full model are both 90.4%. Thus, we will select the reduced model for a simpler model. We also note that the AUC was approximately the same, rounding to the first decimal place again both the reduced model and full model give an AUC value of 92.7%, implying the reduced model is best choice in terms of classification accuracy and simplicity. The reduced model with 14 variables is simpler and gives about the same classification accuracy. since it gave us about the same performance both on the train set and the test set, we can safely remove 6 predictor variables and choose the simpler model.

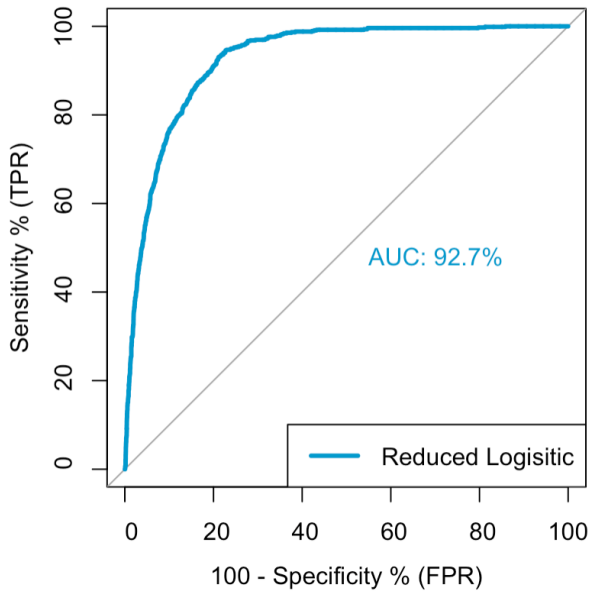


Figure 13: ROC Curve for Logistic Regression with reduced set of features

5 Random Forest

5.1 Theory

Below we will be implementing the *random forest method* for the purpose of our predictive classification model. For the *random forest method* we build a specific number of decision trees on our bootstrapped training data. What distinguishes *random forest* when building the decision trees, is that each time we consider a split, we carry out a random selection of m predictors from our set of predictors in our data.The split then employs one of those m predictors.

5.2 Model Building

Conventionally, we choose m using the following calculation, $m \approx \sqrt{(p)}$. Upon the calculation of $\sqrt{(20)}$, we get 4.472. So, we can carry out the *random forest* method using both, $m = 4$ and 5.

5.2.1 Random Forest with $m = 4$

```
## OOB estimate of  error rate: 9.65%
## Confusion matrix:
Confusion matrix:
       no  yes class.error
no  20384  901      0.0423
yes  1452 1653      0.4676
```

We plot the following graph to find the number of trees that we will be considering when we implement random forest, and our main objective is to minimize error. From the following graph we choose ntree = 50 because we find that there is no clear "elbow" in the plot, and it appears to be stable throughout, from $0 - 500$, on the x axis. Furthermore, we choose ntree = 50 because it yields a relatively similar error when compared to higher values, and is computationally less intensive.

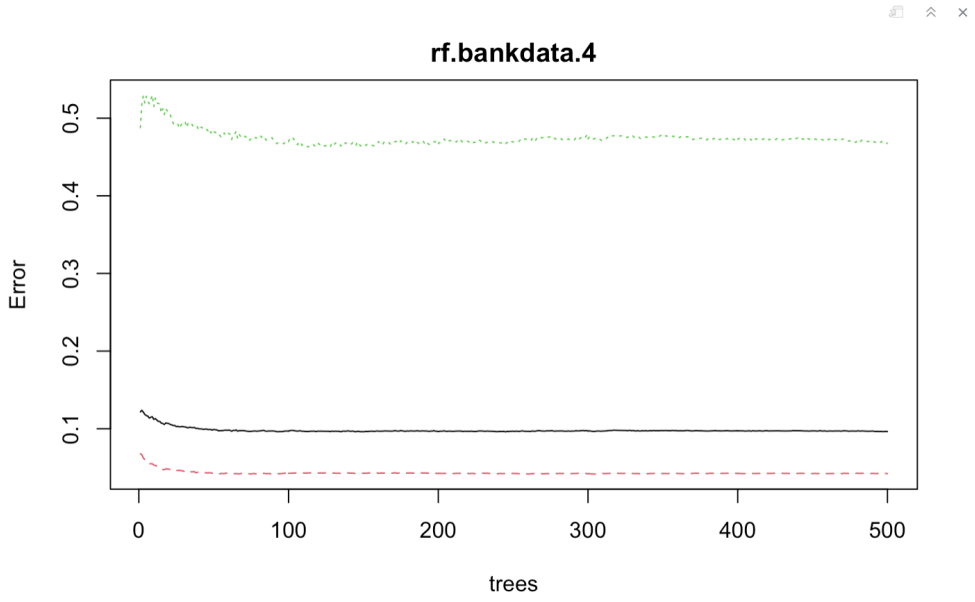


Figure 14: Random Forest with $m = 4$ splits

5.2.2 Random Forest with $m = 5$

We, now, carry out the **random forest method** and implement it using $m = 5$.

```
## OOB estimate of  error rate: 9.75%
## Confusion matrix:
       no  yes class.error
no  20300  985      0.0463
yes  1394 1711      0.4490
```

We find that we yield similar results to the plot mentioned above for $m = 4$. Therefore, we can assume that ntrees $= 50$ is a good estimation for the number of tress that should be considered when employing the **random forest method** on our data.

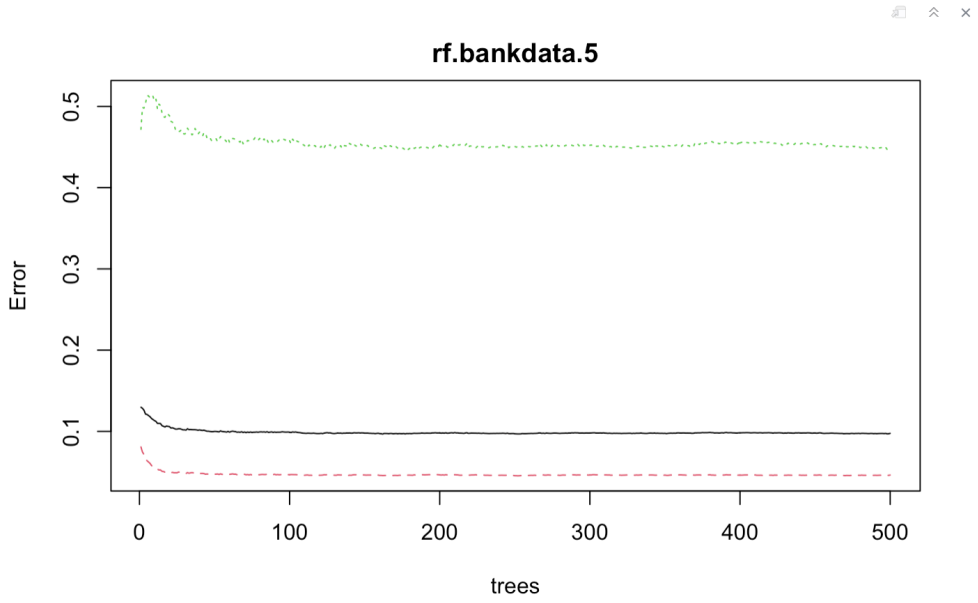


Figure 15: Random Forest with $m = 5$ splits

We will now compute the confusion matrix that will ultimately help us calculate the predictive accuracy of our model when $m = 4$.

```
## Confusion matrix:
          actual
predicted   no  yes
      no  5157  329
      yes  187  425
```

We observe a classification accuracy of 91.5% when $m = 4$. We will also compute the confusion matrix that will ultimately help us calculate the predictive accuracy of our model when $m = 5$. We observe a classification accuracy of 91.3% when $m = 5$.

```
## Confusion matrix:
          actual
predicted   no  yes
      no  5135  319
      yes  209  435
```

5.3 Model Selection

We now compute the Out-of-bag error and the test error for various values of m, and our objective is to find the optimal value of m while simultaneously trying to avoid overfitting our model. We utilize the optimal value of ntree we calculated above to further fine tune our model and it's predictive abilities. We check for overfitting by observing the two graphs and making sure that the Out-of-bag and test errors for the m value have a relatively low difference. From this graph, we find that a m value of 2 appears to have a relatively low error while avoiding the issues of over and under fitting.

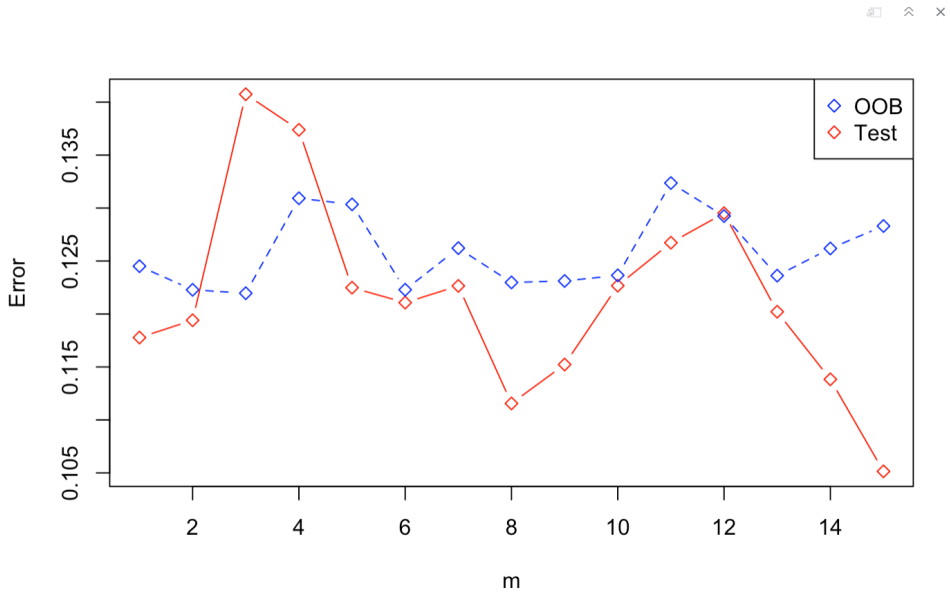


Figure 16: OOB Error on training data vs error on test data

We will now employ the **random forest method** using he following values: $\mathbf{m = 2}$ and $\mathbf{ntrees = 50}$.

```
## OOB estimate of  error rate: 10.5%
## Confusion matrix:
       no  yes class.error
no  20749  536      0.0252
yes  2015 1090      0.6490

## Confusion matrix:
         actual
predicted   no  yes
      no  5253  477
      yes   91  277
```

We observe that the classification accuracy with $m = 2$ is 90.7% which was lower than at $m = 4$ or 5. Although the model trained with $m = 2$ splits is less computationally intensive, ultimately our goal is to achieve the best accuracy. Thus, we will choose the model trained with $m = 4$ splits, because it gave the best classification accuracy and is still relatively simple to compute.

5.4 Model Performance

Finally, we will calculate the roc and, ultimately, the auc values to find out how our *random forest* model performed on the test data. Given the calculations, we find that we obtain an auc value of 94.8%, which indicates our random forest model has excellent performance at predicting correct classifications.

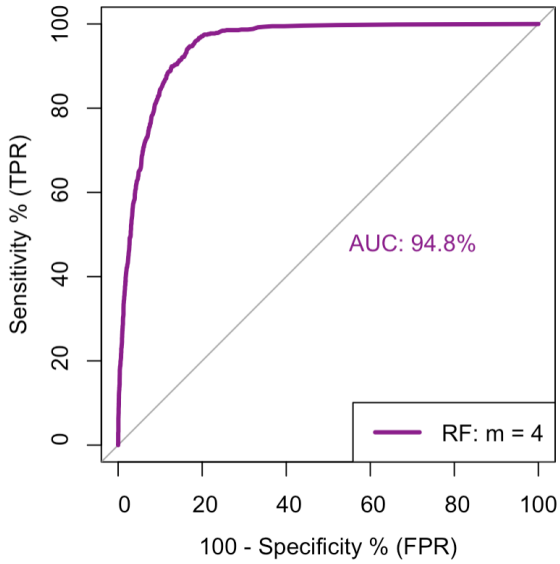


Figure 17: ROC Curve for Random Forest Model

We ran a predictor variable importance plot on our trained random forest model, with $m = 4$ splits. We observe the predictor variable "duration" is ranked very high with respect to both mean decrease in accuracy and gini index. The importance plot can be naively interpreted as a method of feature selection. However, variables that have high cardinality run the risk of the multiple testing problem and other underlying bias can occur from potentially colinear variables. Therefore, we decide to construct new model logistic regression and random forest models with the "duration" variable removed how much it affects our classification accuracy and AUC values.

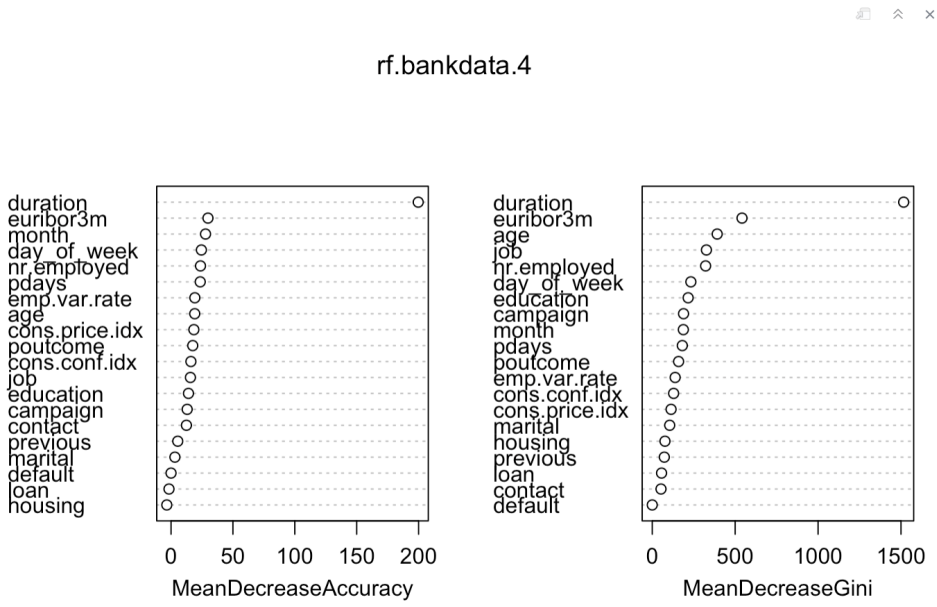


Figure 18: Variable Importance Plot from Random Forest with $m = 4$

6 Build new LR and RF models with the duration variable removed

We trained a new logistic regression model on the full set of predictors, excluding the duration variable. Then, we implemented backwards stepwise regression to choose the best linear combination of predictor variables that gives us the lowest AIC value, while not compromising on classification accuracy. We note the new full model trained on 19 predictor variables, and stepwise regression chose the same 13 predictor variables as our reduced model containing the duration variable.

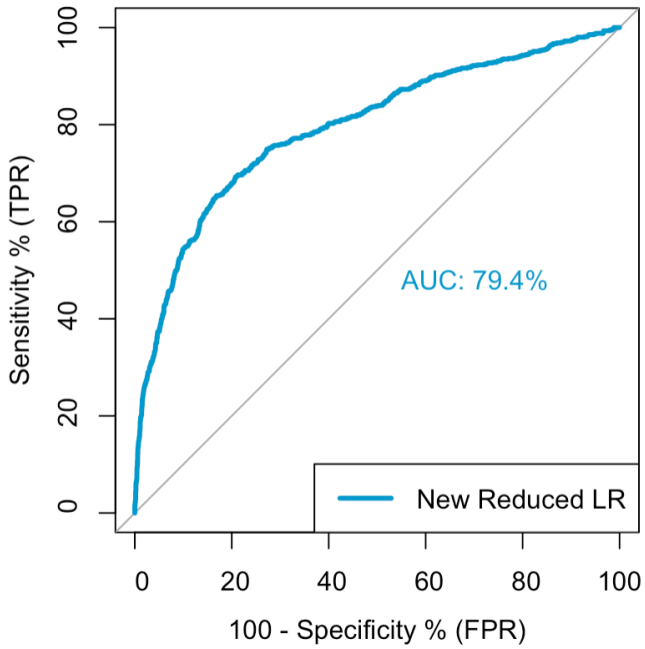


Figure 19: ROC Curve with Reduced Logistic Regression with duration variable removed

Both the full and reduced models, and our new random forest model gave us a lower classification accuracy rate of about 89.0%, and a much lower AUC value of 79.4% for LR and 79.0% for RF, than our full and reduced models implemented with the duration variable. This new accuracy rate and AUC value may be accounting for the optimistically high accuracy and AUC values containing the duration variable, which implies that duration was giving our models false prediction performance. If our models were to be used in real life for the purpose of accurate predictions and classifications, they would have given very inaccurate results and been highly unrealistic. It is also observed that our classification accuracy and AUC values with the duration variable removed, are very close to the values found in the original paper using the bank data set [1].

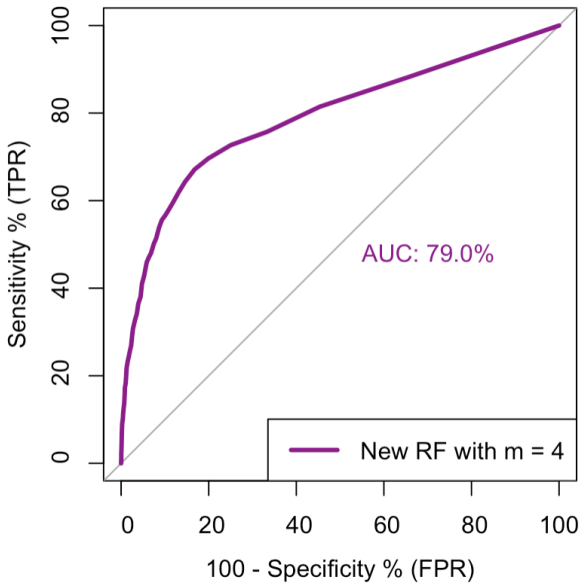


Figure 20: ROC Curve with Random Forest at $m = 4$ splits, duration variable removed

6.1 Interpretation

Given our high dimensional data, we have 20 predictor variables and 30488 observations which makes it challenging to visually infer whether our data is linear or nonlinear. However, the results of the Random Forest model on all of the features out performed both of the logistic regression models in terms of accuracy and area under the curve. From this result, we infer that Random Forest performed better because the algorithm can handle nonlinearity and potential overfitting in the data.

After removing the "duration" predictor variable, logistic regression contrarily out performed Random Forest (only slightly). This indicates that "duration" was contributing to nonlinearity in our data. If this holds true, given that we removed it, it would make sense that the logistic method performed better because it is a linear model. It could also mean that duration was a dependent variable, and removing it caused our feature inputs to be linearly independent, which the logistic method assumes in order to yield accurate classification results.

7 Extra Credit: h2o

7.1 Background and Theory

To go beyond the scope of the class and to implement techniques that help us predict our data better, we have applied a h2o package, which allows us to implement deep learning and run various ML methods, with the important advantage that they are capable of parallel computation. h2o is an open source distributor, scalable framework used to train Machine Learning and Deep Learning models as well as helping in data analysis. It can handle large datasets with ease of use by creating a cluster from the available nodes. It provides an API to reap benefits when it comes to large datasets with which R has its most limitations. h2o used Java virtual machine.The h2o package adopts the "triple cross-validation" approach, which partitions the data into training, validation and test sets. The training and validation sets are used for the package's internal cross-validation and thus are essential. In most algorithms, the more iterations the better, but in NNs the fear is that running the algorithm too long may result in convergence to a poor solution. h2o.deeplearning() comes in handy here, it can help us stop if the accuracy in the validation set begins to deteriorate, even if we have not performed the number of iterations decided.

According to the description of the data we have a response variable 'y' with two values "yes" for that a client will sign on to a long-term deposit and "no" for the client would not sign up for a long term lease. Now, for this portion of the project, we have identified that our data has a serious class imbalance problem. Class imbalance occurs when there is an underrepresentation of a particular class to be predicted. This occurs when we have more instances of a particular class in a training set which largely outnumbers other classes within the dataset. When a high level of imbalance occurs within a dataset there tends to be issues with creating an effective classification model which can be used to accurately make class predictions. In class imbalance, the class instances belong to either the majority or the minority class. The minority class refers to the class within the dataset which have very few instances represented while the majority class refers to the class whose instances out number others with several occurrences. A classifier which is biased towards the majority class could record a high accuracy for its overall performance but record a very poor performance for

the minority instances.

We also recognize that AUC model metric, which used to evaluate how well a binary classification model is able to discriminate between True Positives and False Positives and AUC of 1 indicates a perfect classifier, while an AUC of 0.95 indicates a poor classifier. For imbalance data a large quantity of True Negatives usually overshadows the effect of changes in other metrics like False Positives. So, to account for the imbalance in the data,we need to balance the data to know the correct AUC.

The most commonly used method for solving class imbalance is the data sampling method. The data sampling method consists of two major categories. This is recognized as undersampling and oversampling of the dataset. Oversampling is the process of increasing the minority class instances while undersampling is the process of reducing the majority class instances.

In our model, we have set 80% of the data in the train set and 19% in the validation set. We checked the imbalance in the data with the following command "h2o.table(bankdata$y)". This was out result:

	y <dbl>	Count <dbl>
1	0	36548
2	1	4640

As expected the majority of the cases are of class label "0" , any Machine Learning model fitted to this data without correcting this problem will be dominated by the label "0" and will hardly correctly predict if the client would sign up for a long term lease (Label = "1") - which is our main interest.

7.2 GLM via h2o

Now, to identify the error for the minor class, we find the confusion matrix for imbalanced data using model GLM. We get the following results:

For Training Set :-

	0 <chr>	1 <chr>	Error <chr>	Rate <chr>
0	27250	2079	0.070885	=2079/29329
1	1148	2585	0.307527	=1148/3733
Totals	28398	4664	0.097605	=3227/33062

For Validation Set :-

	0 <chr>	1 <chr>	Error <chr>	Rate <chr>
0	6356	516	0.075087	=516/6872
1	271	587	0.315851	=271/858
Totals	6627	1103	0.101811	=787/7730

For Test Set :-

	0 <chr>	1 <chr>	Error <chr>	Rate <chr>
0	333	14	0.040346	=14/347
1	19	30	0.387755	=19/49
Totals	352	44	0.083333	=33/396

So, we get the error rates for Minor class in the Training set as: 0.307, Validation Set: 0.315, for Test Set: 0.3877 - which are too high.

The h2o package provides a way to correct this imbalance problem. We have an argument called "balancedclasses " for this purpose. If "balancedclasses" is set to "TRUE" then it performs a sub-sampling method by default. For this problem, we use the GLM model using h2o with a balanced factor. In GLM, we have parameters like nfolds (the number of folds to use for cross-validation for hyper parameter tuning) , family (many distributions are provided for binary - we have binomial). h2o provides a bunch of matrices - we can get access to them by calling "h2o.performance()" [3]. Here are our results:

We get a **confusion matrix for Training Set** as follows :-

0 <chr>	**1** <chr>	**Error** <chr>	**Rate** <chr>	
0	27117	2212	0.075420	=2212/29329
1	1096	2637	**0.293598**	=1096/3733
Totals	28213	4849	0.100054	=3308/33062

And we again check the **confusion matrix for the Validation Set**

0 <chr>	1 <chr>	Error <chr>	Rate <chr>	
0	6261	611	0.088912	=611/6872
1	235	623	**0.273893**	=235/858
Totals	6496	1234	0.109444	=846/7730

And, now, checking for the most important metric - **The Test Set**

	0 <chr>	**1** <chr>	**Error** <chr>	**Rate** <chr>
0	333	14	0.040346	=14/347
1	19	30	**0.387755**	=19/49
Totals	352	44	0.083333	=33/396

Note the AUC for the test set: 0.92319.

From the above results, we can see that our error for the minor class in the Training set is 0.29, error for the minor in validation set is 0.27 and error for the minor class in test set is 0.38. We note that the error rate for minor classes in the Training set and validation set reduces when we use the Balance factor, but we also note that the Test set error did not reduce.

7.3 Random Forest via h2o

Now, to reduce the minor class error rate, we implement a second model - Random Forest via h2o package. This Random Forest model will help us in capturing the complex patterns in

data which may have negative effects as it exceedingly memorizes data including the noise, which gives rise to the overfitting problem. To tackle this, Random Forest has a large number of hyper-parameters to control the training process. H2o provides these parameters, helping the Random Forest. We also note that a large change in the number of false positives can lead to a small change in the false positive rate, but by comparing false positives to true positives rather than true negatives we would be able to capture the effect of the large number of negative examples. Our goal is also to reduce the error for the minor class, which Random Forest helps in by setting a parameter of "balancedclasses = TRUE". We again created a confusion matrix for random forest for training set and our results:

We again created a confusion matrix for random forest for **training set** and got :-

	0 <chr>	**1** <chr>	**Error** <chr>	**Rate** <chr>
0	24932	4397	0.149920	=4397/29329
1	1522	27815	**0.051880**	=1522/29337
Totals	26454	32212	0.100893	=5919/58666

We now form a confusion matrix for the **Validation set** :-

	0 <chr>	**1** <chr>	**Error** <chr>	**Rate** <chr>
0	6356	516	0.075087	=516/6872
1	218	640	**0.254079**	=218/858
Totals	6574	1156	0.094955	=734/7730

Now, checking for the **Test Set**

	0 <chr>	1 <chr>	Error <chr>	Rate <chr>
0	333	14	0.040346	=14/347
1	17	32	**0.346939**	=17/49
Totals	350	46	0.078283	=31/396

AUC increases to 0.92430.

This model is the best until now as we can see that our Error Rate for Minor Class in Training Set is 0.051, Error rate for minor class in Validation set is 0.2540, Error rate for minor class in Test set is 0.34. We note that we were successful in reducing the error rate, as the results show significantly low minor class error rate. In order to balance the class more

accurately, we implement another model with a different set of parameters.

7.4 Deep Learning via h2o

Next, we implement a deep learning model. This is known for its high accuracy at predicting very large and complex data sets. They have a large number of hyper parameters that can be tuned to efficiently handle a wide range of dataset. Using a large number of hyper parameters with large datasets requires very large hardware resources and time, which is expensive and hardly available. This is where deep learning comes into play. H2o provides feed-forward neural networks, which are densely connected layers, helping it supplement the model. We also specify the number of hidden layers, which applies a non-linearity to its input, and the more hidden layers we stack together, the more complex functions we will be able to model. We then specify the number of nodes in each layer. We have provided nfolds for cross validation. We also have implemented activation - "Rectifier", which leads to much faster training speed. One strategy to improve our model is to remove the less important variables using a threshold. h2o provides a function to list the predictors and their importance in predicting the response variable - so we can think about removing the less important variables, with the hope to reduce the error rate. In this model, we have also used Variable importance which is used for a better model.

For this we get the confusion matrix on the **Training set** as :-

0 <chr>	**1** <chr>	**Error** <chr>	**Rate** <chr>	
0	4704	437	0.085003	=437/5141
1	144	4807	0.029085	=144/4951
Totals	4848	5244	0.057570	=581/10092

We get the following confusion matrix for the **Validation set** as :-

	0 <chr>	**1** <chr>	**Error** <chr>	**Rate** <chr>
0	6145	727	0.105792	=727/6872
1	206	652	0.240093	=206/858
Totals	6351	1379	0.120699	=933/7730

Now, checking for **the Test Set :-**

0 <chr>	1 <chr>	Error <chr>	Rate <chr>	
0	305	42	0.121037	=42/347
1	8	41	**0.163265**	=8/49
Totals	313	83	0.126263	=50/396

This result suggests that the DeepLearning model was supported by parameters which helped it perform better than other models. One of the parameters that made the difference was the Activation - Rectifier. The reason for this is that H2O Deep Learning is optimized to take advantage of sparse activation of the neurons, and the Rectifier activation function ($max(0, x)$),which is sparse and leads to less computation especially - leading to a low test set error. One of the other parameters was "nfolds", when we specify nfolds, the algorithm builds nfolds + 1 models. In our program, we set nfolds = 5, which means it creates 6 models. First 5 models are created on 80% of the training data, and a different 20% is held out for each 5 models. The main model is built on 100% of the training data - this model is which we get back. Since it ensures that every observation from the dataset has the chance of appearing in training and test set, this method results in a less biased model.

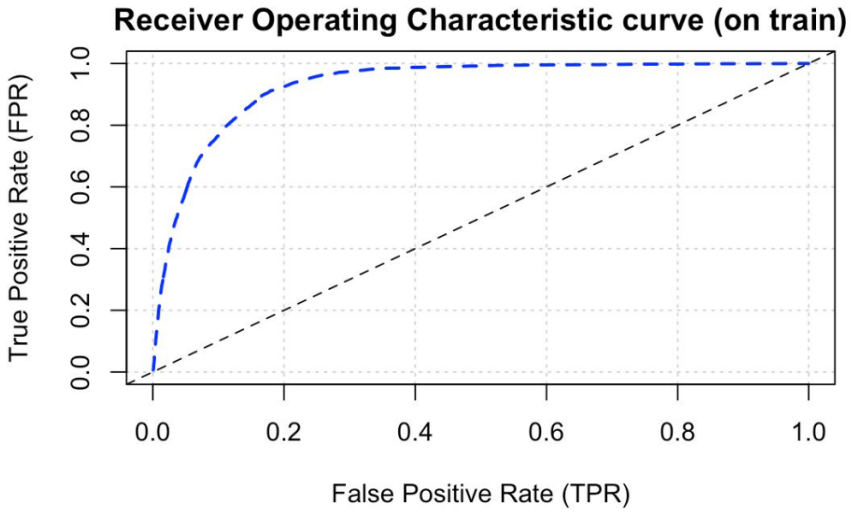
Going one step further in order to ensure the performance, we try to handle the under-fitting/overfitting problem. We set the number of epochs in deep learning - An optimization procedure which updates the weights in the neural network in an iterative fashion. If we

train the network for a few epochs, it leads to underfitting of the data. This means it cannot capture the underlying trend of the data. Now, as we increase the number of epochs, it will reach an optimal situation where we will get maximum accuracy on the training set. Beyond this stage, if we keep increasing the number of epochs, it will lead to overfitting of the data. That is the accuracy in the training set increases whereas that of validation set decreases. This means the network does not reflect the reality of the data because it captures the noise in the data as well. Having set the number of epochs to "25", and running the model multiple times, we have ensured that we are covered from the underfitting/overfitting problem resulting in superior results.

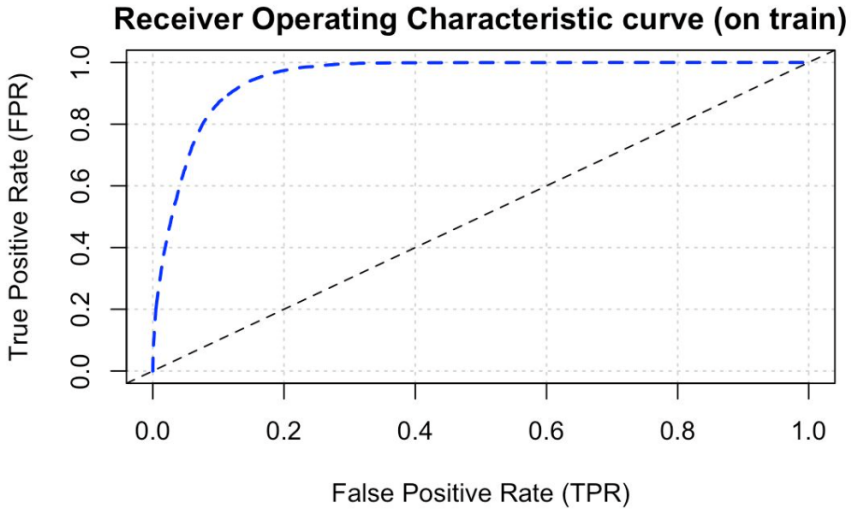
Taking all these factors into account, we get minor class error for Training set 0.029, minor class error for Validation set is 0.2400, minor class error for Test Set is - 0.16. The error for the minor class as seen in the deep learning model, suggests that our Error for Minor class has significantly reduced from the previous models - GLM and Random Forest. We also see that Deep Learning has an AUC value of 0.93512, higher than RandomForest and GLM.

ROC Curves help us understand the trade off between sensitivity (TPR) and specificity (1-FPR). Classifiers that give curves closer to the top-left corner indicate a better performance.

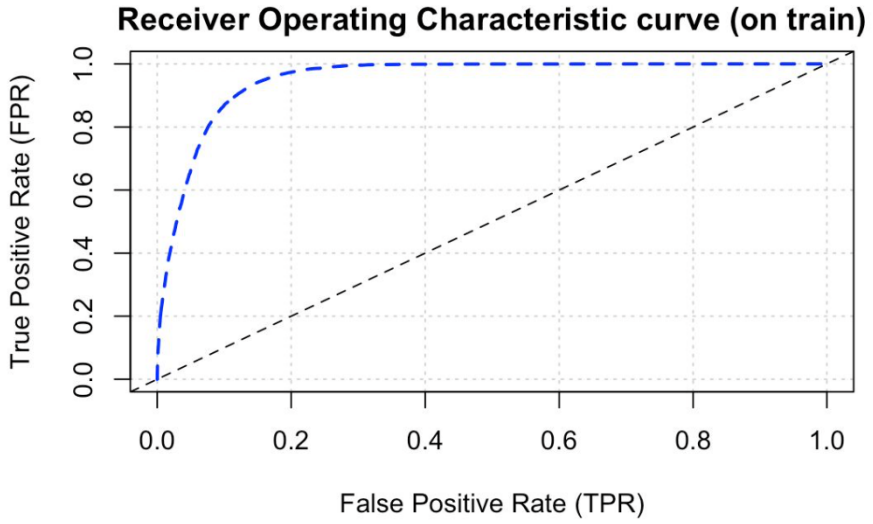
GLM ROC CURVE :-



DEEP LEARNING ROC CURVE :-



RANDOM FOREST ROC CURVE :-



7.5 Interpretation

From the above results, we see that in order to balance the classes accurately we need precise parameters. For large and complex data, the most powerful models are in order: **DeepLearning** > **Random Forest** > **GLM**. Deep Learning not only gives us the lowest minor class error from the confusion matrix which helps us in predicting the label 1, but also the highest AUC value which helps us in predicting 0's as 0's and 1's as 1's.

8 Summary

The features euribor3m and nr.employed consistently have significant impact on client response according to our univariate analysis and predictive models. We implemented a full and reduced logistic regression model, and chose the reduced model for simplicity. Then, we fine tuned the number of splits and the number of trees for Random forest by plotting the OOB training error vs test error. We found that $m = 2$ gave good accuracy results, but $m = 4$ gave the best accuracy and AUC value. So we chose a Random forest model with $m = 4$. From the variable importance plot, we noticed the duration feature appeared to be too strong of a predictor, and was giving us optimstically high performance. After removing

this feature, logistic regression and random forest gave an AUC value of approximately 79%, which was consistent with the AUC results found in the original paper on this bank dataset [1].

9 Extra Credit Summary

We implemented stepwise regression as a method of model selection in logistic regression. We found that a reduced set of feature inputs yielded about the same accuracy results and AUC values. Thus, we decided to select the logistic model with a reduced set of features because it is simpler. Implemented h2o package to solve class imbalance problem by comparing three different models :- GLM , Random Forest and Deep Learning based on the confusion matrix of Training set, Validation set and Test set. Found that Deep Learning model worked the best out of three. Deep Learning not only helped in significantly reducing the minority class error, but also it gave the highest AUC value. Created a heatmap for Pearson Correlation in order to help determine any collinear variables within our dataset.

References

[1] Sérgio Moro, Paulo Cortez, and Paulo Rita. "A data-driven approach to predict the success of bank telemarketing". In: *Decision Support Systems* 62 (2014), pp. 22–31.

[2] Zhongheng Zhang. "Variable selection with stepwise and best subset approaches". In: *Annals of translational medicine* 4.7 (2016).

[3] H2O.ai. *Tree Class in H2O*. 2020 (accessed December 1, 2020). URL: http://docs.h2o.ai/h2o/latest-stable/h2o-py/docs/tree.html.