Statisical Analysis of Bank Data

Team ID: Group 29

Name (tasks): Amay Kharbanda (LASSO, Logistic Regression, parts of the report)

Name (tasks): David Wong (Decision Tree Classification, Literature Review, parts of the report)

Name (tasks): Alejandro Medina (Neural Networks, Random Forest, parts of the report)

Introduction

In this project, our goal is to analyze data collected from 2008-2010 of a Portuguese bank marketing campaign containing client descriptions and engagement. We are provided a large dataset labeled "bank-additional-full.csv" which contains the data collected in its entirety. We are also provided a smaller dataset containing randomly sampled data from "bank-additional-full.csv" which is labeled as "bank-additional.csv". This smaller dataset allows us to run computationally intensive algorithms and will be known as our training data. The larger dataset will be used to test any models derived from our smaller dataset and will be known as our testing data.

Our ultimate goal for this analysis is to answer our primary statistical question of interest. We want to develop a statistical model that accurately predicts whether a client will sign on to a long-term deposit or not. This begs the question, how would we formulate an accurate model and what attributes would it consist of? Within this analysis, we will develop a range of statistical models using different methods learned in class. We will compare these methods and build three different classification models which include logistic regression, classification tree, and random forest. These models will then be used against our testing data. This will allow us to find model accuracy and in turn, will help decide which model is best at predicting who will sign onto a long-term deposit.

Statistical questions of interest

Our primary statistical question of interest is determining an accurate prediction model and evaluating what significant attributes it would utilize. The secondary question we seek to answer is to find the model with the highest accuracy and prediction rate.

Population and study design

The target population is clients who will sign on to a long-term deposit for the bank. We will be utilizing a training dataset and a full dataset. The training data is randomly sampled data pulled from the full dataset containing around 41,000 different subjects. Below we have included the input and output variables within this dataset along with helpful descriptions to interpret each one.

Variables in the Data Set

Input Varibles:

# bank client data:

1 - age (numeric)

2 - job : type of job (categorical: “admin.”,“blue-collar”,“entrepreneur”,“housemaid”,“management”,“retired”,“self-employed”,“services”,“student”,“technician”, “unemployed”,“unknown”)

3 - marital : marital status (categorical: “divorced”,“married”,“single”,“unknown”; note: “divorced” means divorced or widowed)

4 - education (categorical: “basic.4y”,“basic.6y”,“basic.9y”,“high.school”,“illiterate”,“professional.course”, “university.degree”,“unknown”)

5 - default: has credit in default? (categorical: “no”,“yes”,“unknown”)

6 - housing: has housing loan? (categorical: “no”,“yes”,“unknown”)

7 - loan: has personal loan? (categorical: “no”,“yes”,“unknown”)

# related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: “cellular”,“telephone”)

9 - month: last contact month of year (categorical: “jan”, “feb”, “mar”, . . . , “nov”, “dec”)

10 - day_of_week: last contact day of the week (categorical: “mon”,“tue”,“wed”,“thu”,“fri”)

11 - duration: last contact duration, in seconds (numeric).

# other attributes:

12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

14 - previous: number of contacts performed before this campaign and for this client (numeric)

15 - poutcome: outcome of the previous marketing campaign (categorical: “failure”,“nonexistent”,“success”)

# social and economic context attributes

16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)

17 - cons.price.idx: consumer price index - monthly indicator (numeric)

18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)

19 - euribor3m: euribor 3 month rate - daily indicator (numeric)

20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

21 - y - has the client subscribed a term deposit? (binary: “yes”,“no”)

Analysis Plan

Our analysis plan consists of feature selection, model design, and model comparisons. Our ultimate goal is to implement a model that will accurately predict whether or not a client will sign onto a long term deposit. For feature selection, we plan to implement LASSO to reduce the number of predictors within our models. Our goal is to avoid overfitting our models by overloading them with too many predictor variables. Once feature reduction has been achieved we plan on fitting two different models using the predictors selected. These models will be classification based as we are working with many categorical variables and binary output. The first model we plan to implement is a logistic regression model, and the second model we plan to implement is a tree classification model. We also plan on fitting a random forest model which deals with feature selection within its function, so it will be fit separately from the predictors used for our first two models. Once each model has been implemented, we will compare the models using the AUC statistic taken from the ROC curve to conclude which model is best suited for predicting whether or not someone will sign onto a long term deposit. Also, we plan on exploring and implementing a model using neural networks.

Analysis

LASSO and Feature Selection

The Least Absolute Shrinkage and Selection Operator(LASSO) is a method used to perform regularization and feature selection for a given dataset. The operation of LASSO penalizes complexity in β, and often this leads to values of β minimizing exactly to 0. This means that the minimization of β produces a sparse model which can be modulated by changing λ. The mathematical interpretation of the penalization is given below for reference:

For LASSO we choose $\beta_0, \beta_1, ..., \beta_k$ to minimize:

$$avg[L(outcome, \beta_0 + \beta_1 x_1 + ... + \beta_k x_k)] + \lambda P(\beta_1, ..., \beta_k)\ where$$

$$P(\beta_1, ..., \beta_k) = |\beta_1| + ... + |\beta_k|$$

In regards to this project, we will utilize LASSO to minimize our predictor variables and disregard predictors that minimize to 0. We will use the remaining predictors as a reference for our model production. When fitting our training dataset using the LASSO function in R, we will also make use of cross-validation. Cross-validation is used to assess the predictive performance of the models produced, especially on unseen data.

Logistic Regression

Logistic regression is what is considered to be a likelihood-based machine learning method. When fitting a logistic regression model we generally want to choose parameters in a model to maximize the likelihood of observing a collection of data. This is the equivalent of maximizing the log-likelihood or minimizing the negative-log-likelihood. For a model with binary outcome y we often model the probability y = 1 as $f(x)$. Below is a mathematical interpretation of minimizing the negative-log-likelihood in terms of a model, known as logistic regression:

$L(y, f(x)) = -y \cdot log(\frac{f(x)}{1-f(x)}) + log(1 - f(x))$

$log(\frac{f(x)}{1-f(x)}) = x_1\beta_1 + ... + x_p\beta_p$

equivalently

$f(x) = \frac{e^{x_1\beta_1 + \cdots + x_p\beta_p}}{1+e^{x_1\beta_1 + \cdots + x_p\beta_p}}$

Logistic Regression is a type of classification algorithm that can be used to predict a binary outcome, given a set of predictors. In regards to this project, we will implement a logistic regression model using our training data to predict the probabilities of given binary outcomes. The model will be fit using predictors obtained from our LASSO feature selection method. We will test the model against our testing data to determine the accuracy of the model.

Decision Tree Classification

Decision tree learning is one of the predictive modeling approaches used in statistics and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value or response (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs. Further classification is conducted through a process known as binary recursive partitioning. This approach is also commonly known as divide and conquer because it splits the data into subsets, which are then split repeatedly into even smaller subsets, and so on until the process stops when the algorithm determines the data within the subsets are sufficiently homogeneous, or another stopping criterion has been met. In other words, we can then repeat this splitting procedure at each child node until the leaves are pure. This means that the samples at each leaf node all belong to the same class. In practice, we may set a limit on the depth of the tree to prevent overfitting. We compromise on purity here somewhat as the final leaves may still have some impurity.

Random Forests

Random forests are an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set. They provide an improvement over bagged trees by way of a small tweak that de-correlates the trees. Bagging (Bootstrap Aggregation) is a technique used to reduce the variance when we average the trees. Due to this, random forests generally outperform decision trees.

The training algorithm - Given a training set $X = x_1, \ldots, x_n$ with responses $Y = y_1, \ldots, y_n$ bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, ..., B$,

- Sample, with replacement, n training examples from X, Y call these X_b, Y_b.
- Train a classification or regression tree $f(b)$ on X_b, Y_b.
- After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

  $$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f_b(x')$$

  or by taking the majority vote in the case of classification trees.

Data Manipulation

Categorical Transformation

To build statistical models, we need to be able to run our data through computationally intensive algorithms, but large datasets don't tend to do well in regards to running time. This is where the functionality of our training dataset comes into play. Given a smaller dataset containing sampled data from our full dataset, we can utilize the data in algorithms that are much more computationally intensive. From this point forward we will be working with our training dataset for model fabrication, and our full dataset will be used for testing. For the functionality of a few functions, inputs need to be provided as numerical values. Given the fact that we are working with categorical variables, our first step is to build and transform our training data into a new dataset. It is a simple process of data manipulation where we transform our categorical values into numerical values. As an example, say a categorical variable labeled color has three possible outcomes: blue, yellow, and red. Transformation of this data would simply change each outcome to a numerical value. Given this, blue would transform to 1, yellow to 2, and red to 3. Let's observe a sample of our variables to display the data before manipulation. (Note: it is also worth mentioning that all values labeled "unknown" in our training dataset will be transformed to NULL)

```
##   age         job marital   education default housing loan   contact month
## 1  30 blue-collar married    basic.9y      no     yes   no  cellular   may
## 2  39    services  single high.school      no      no   no telephone   may
## 3  25    services married high.school      no     yes   no telephone   jun
## 4  38    services married    basic.9y      no    <NA> <NA> telephone   jun
##   day_of_week duration campaign pdays previous    poutcome emp.var.rate
## 1         fri      487        2   999        0 nonexistent         -1.8
## 2         fri      346        4   999        0 nonexistent          1.1
## 3         wed      227        1   999        0 nonexistent          1.4
## 4         fri       17        3   999        0 nonexistent          1.4
##   cons.price.idx cons.conf.idx euribor3m nr.employed  y
## 1         92.893         -46.2     1.313      5099.1 no
## 2         93.994         -36.4     4.855      5191.0 no
## 3         94.465         -41.8     4.962      5228.1 no
## 4         94.465         -41.8     4.959      5228.1 no
```

Now let's go ahead and observe our variables after the numerical transformation of our categorical values and removal of NA values

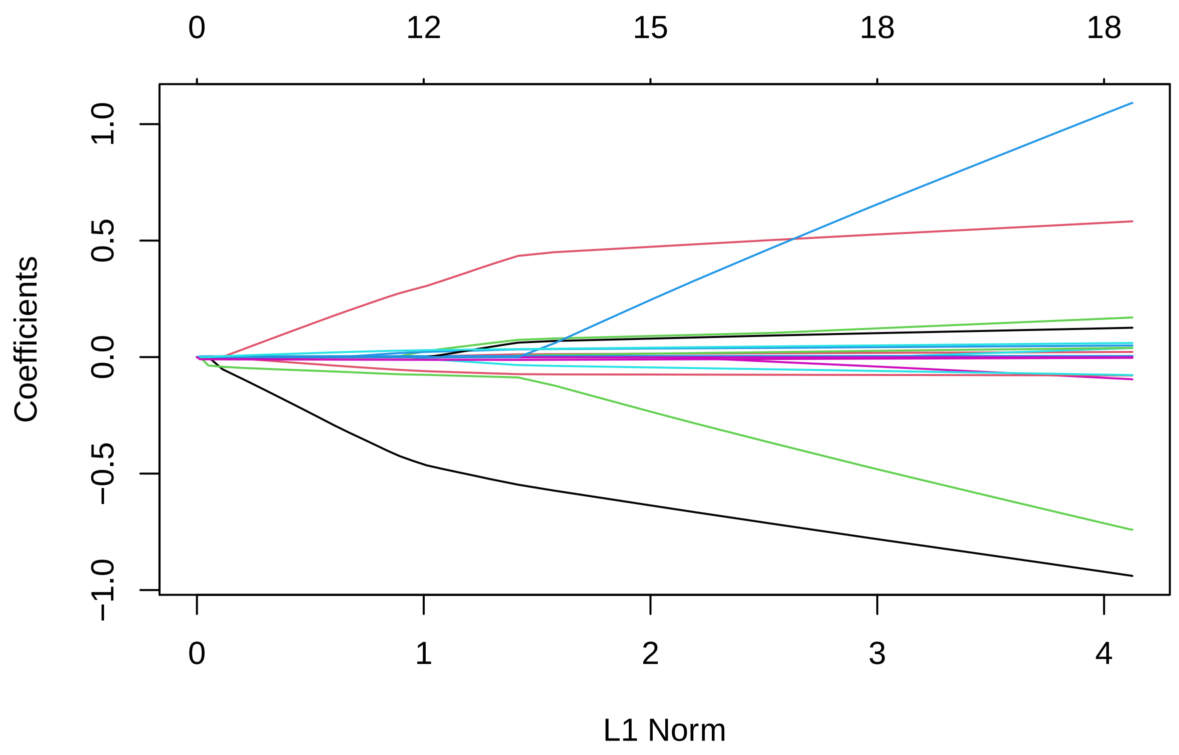
```
##   age job marital education default housing loan contact month day_of_week
## 1  30   2       2         3       1       2    1       1     7           1
## 2  39   8       3         4       1       1    1       2     7           1
## 3  25   8       2         4       1       2    1       2     5           5
## 5  47   1       2         7       1       2    1       1     8           2
##   duration campaign pdays previous poutcome emp.var.rate cons.price.idx
## 1      487        2   999        0        2         -1.8         92.893
## 2      346        4   999        0        2          1.1         93.994
## 3      227        1   999        0        2          1.4         94.465
## 5       58        1   999        0        2         -0.1         93.200
##   cons.conf.idx euribor3m nr.employed y
## 1         -46.2     1.313      5099.1 0
## 2         -36.4     4.855      5191.0 0
## 3         -41.8     4.962      5228.1 0
## 5         -42.0     4.191      5195.8 0
```

As can be observed our training data has now been transformed from categorical variables to numerical. This same process will be performed on our full dataset as the testing functionality of a few of the functions used within the analysis rely on numerical values. Given that both the datasets contain the same predictors and response variables, we will refrain from outputting the full dataset to avoid redundancy.

LASSO for Feature Selection

LASSO using Training data

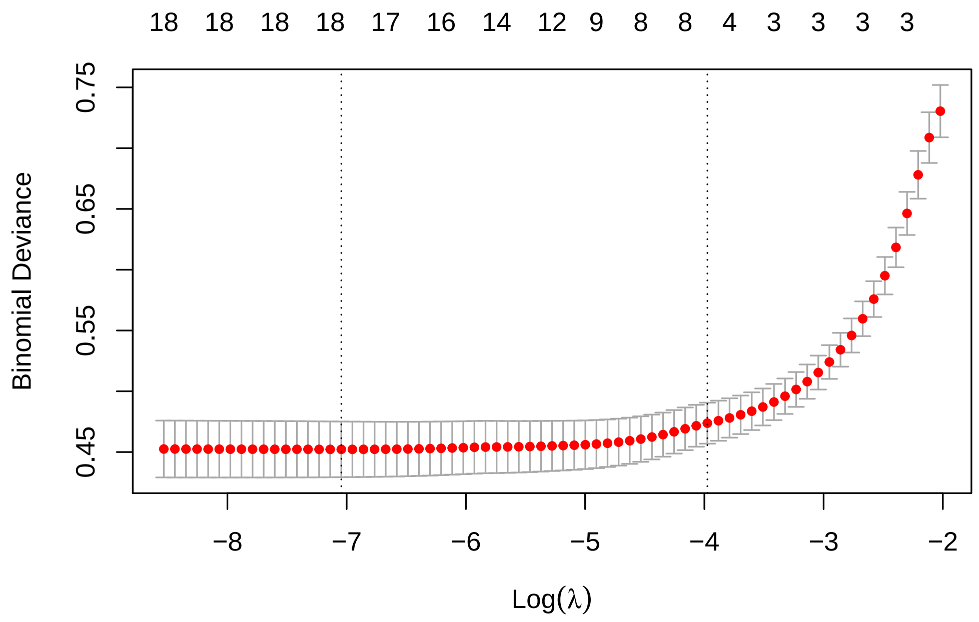
In this section, we will focus on the implementation of LASSO. We will utilize the glmnet package which allows for the implementation of both ridge and LASSO within the same function. To fit the LASSO model we will be using our training dataset which will then be tested against using the full dataset using the *predict* function. To implement LASSO rather than Ridge Regression, we will set our alpha = 1 within our function. We also have to take into consideration that since our response variable produces a binary output we must set family = binomial. We will also allow the function to produce its own λ for now. Utilizing these settings for our function, we fit our LASSO model using our training dataset. We then plot our model below:



The given plot gives us the L1 Regularization of our variables. As can be observed, a few of our predictors are minimizing to zero. We want to be able to extract this information but before we predict our coefficient values our next step is to find an optimal value of λ using cross-validation. This will ensure that the LASSO model will perform well with data that has not been seen.

Cross-validation of our LASSO model

Here we fit the same model as we did before, but instead use the cv.glmnet function to perform cross-validation on our training set. We then plot the results of cross-validation for observational interpretation. The plot is given below:



What is shown in this plot is the values for log(λ). As the values move towards the right of the plot, the penalization increases, meaning more of the predictor variables are being excluded from our model. We want to pick the lambda value that reduces over-fitting for our model yet has enough predictors to build an accurate model. The process of picking a lambda will be discussed next.

Choosing Lambda and Predicting Coefficients

Our goal is to find the smallest value of lambda to predict our coefficients using the full dataset. This can easily be achieved as the function cv.glmnet stores the most optimal lambda under the label *lambda.min*. The next step is to pinpoint exactly what coefficients converged to 0 utilizing the full dataset. We do this by fitting a LASSO model on our full dataset and utilizing our predict function. Our function takes in the full dataset model, the type of values we want to predict(coefficients in this case), and the optimal lambda we extracted from our cross-validated training model. Below are the coefficient values that were produced:

```
##  Best Lambda
## 0.0008718337
```

```
##    (Intercept)           age           job       marital     education
##  -7.7611878241  0.0003576056  0.0006089235  0.0042729923  0.0044779431
##        default       housing          loan       contact         month
##   0.0000000000  0.0000000000 -0.0019955321 -0.0707156807 -0.0079745425
##    day_of_week      duration      campaign         pdays      previous
##   0.0040192333  0.0004645425  0.0006103663 -0.0002293269  0.0000000000
##       poutcome  emp.var.rate cons.price.idx cons.conf.idx     euribor3m
##   0.0603673025 -0.0710177762  0.1204127136  0.0070674014  0.0087903734
##    nr.employed
##  -0.0005935114
```

We can interpret the variables default, housing, and previous as the predictors that we may remove from our statistical models. Moving forward with this analysis we will be excluding these variables from our predictive models.

Logistic Regression

Fitting the Model and Model Accuracy

After running our feature selection method and obtaining the unimportant predictors, we can proceed with eliminating them and creating our model using Logistic Regression.

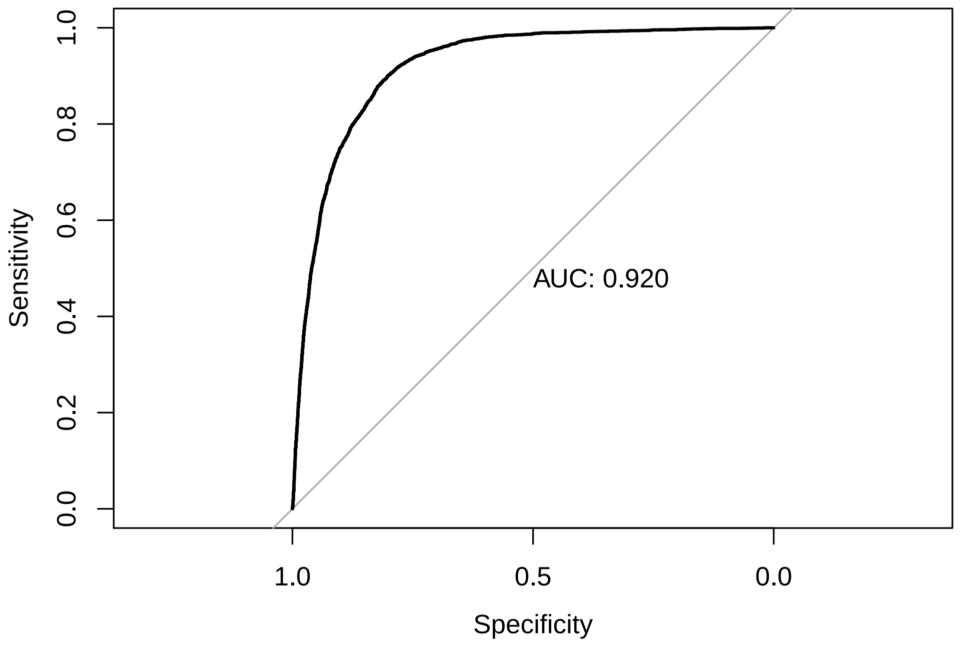
```
## Model Accuracy
##       0.899895
```

Using the glm() function, we compute our logistic regression model. We apply our binary response variable to all the predictors and calculate coefficients for each.

We proceed to calculate the probabilities of each observation from the data set in terms of our logistic model (after removing the unimportant variables). Then we apply the condition of “probabilities > 0.5” to segregate the model probabilities and obtain the model’s predicted values. Finally, we take the average of the results from the condition to determine the final model accuracy which comes up to 0.899 or approximately 90 percent.

ROC and AUC for Logistic Model

Now, we find the ROC (Receiver Operating Characteristics) curve. For that, we use our predicted response calculated from earlier and apply our roc() function. From this, we obtain our ROC curve.

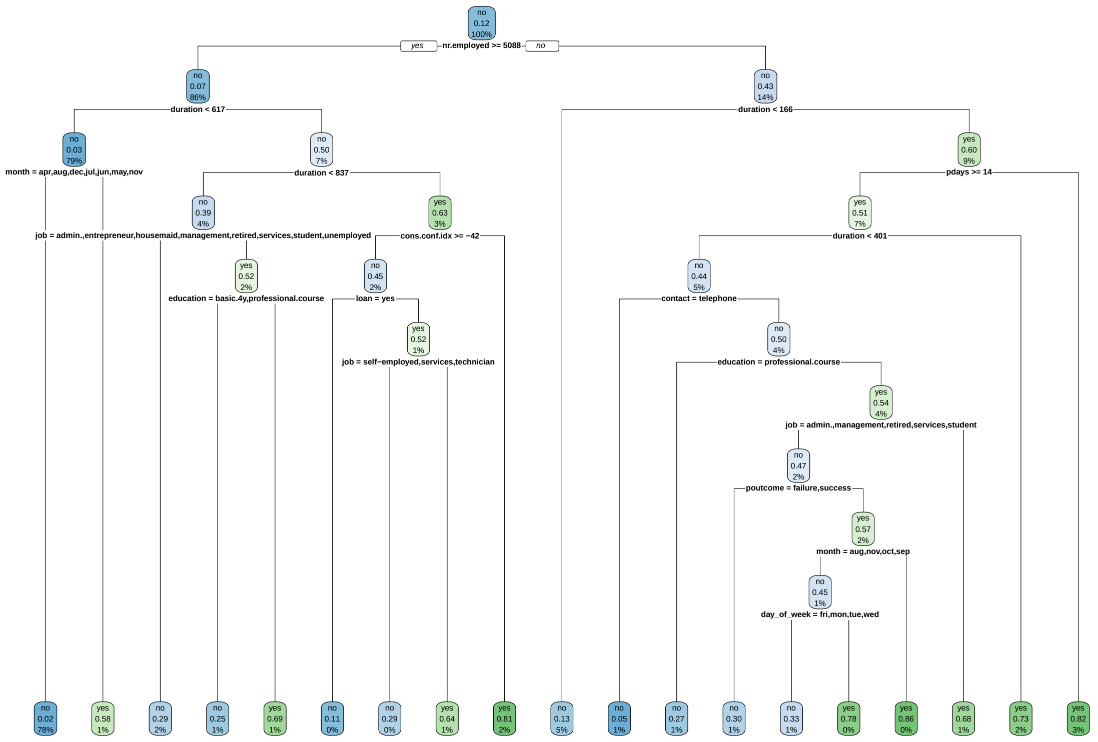


Since the ROC plot shows the trade-off between sensitivity and specificity, it curves closer to the top - left of the graph thus indicating better performance. This information is complemented by the Area Under the Curve (AUC) value, which, in this case, is 0.92 on a scale of 0 to 1. This reflects high accuracy.

Tree Classification

For this model, we decided to implement the feature selection criteria from LASSO and apply the concept of classification tree models to construct the final model for predictions. Our decision variable (or response) is of binary output making the classification categorical or discrete. The key idea is to use a decision tree to partition the data space into cluster (or dense) regions and empty (or sparse) regions.

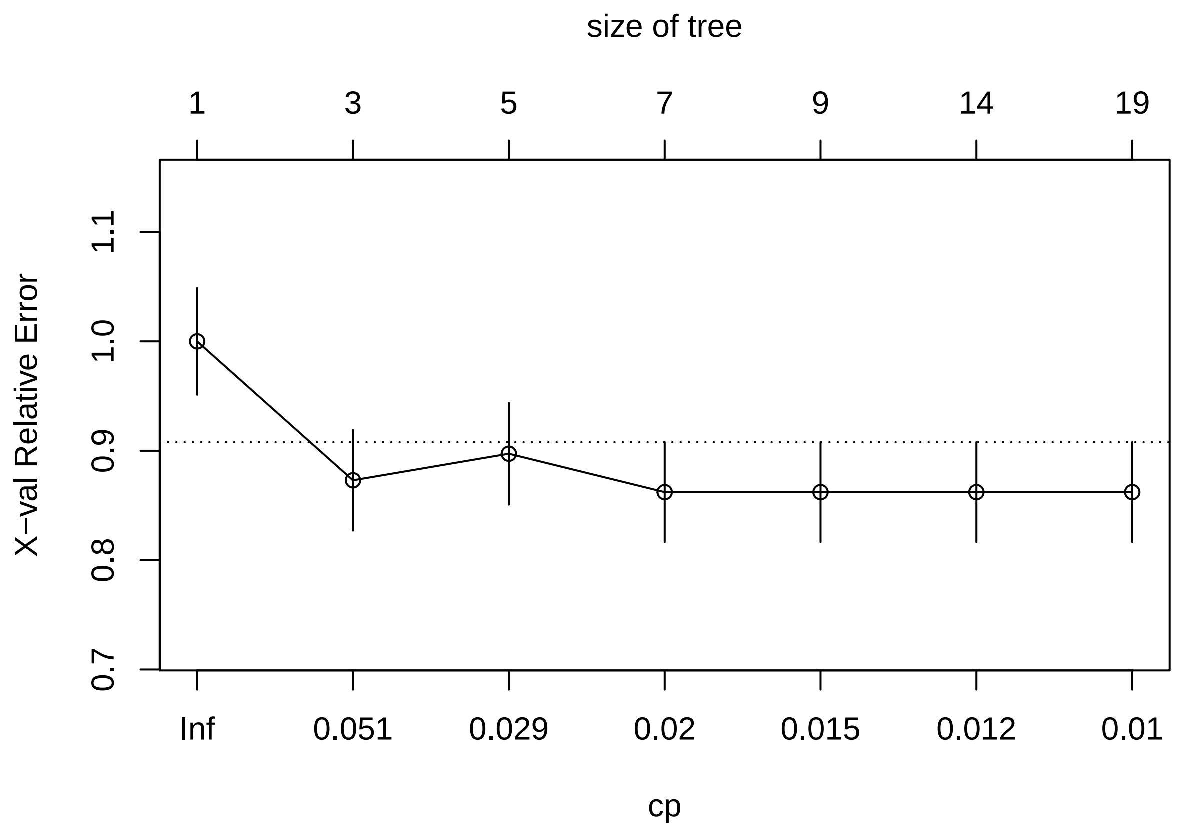
Fitting the Tree



yes
nr.employed >= 5088
no
duration < 617
duration < 166
month = apr,aug,dec,jul,jun,may,nov
duration < 837
pdays >= 14
job = admin.,entrepreneur,housemaid,management,retired,services,student,unemployed
cons.conf.idx >= -42
duration < 401
education = basic.4y,professional.course
loan = yes
contact = telephone
job = self-employed,services,technician
education = professional.course
job = admin.,management,retired,services,student
poutcome = failure,success
month = aug,nov,oct,sep
day_of_week = fri,mon,tue,wed

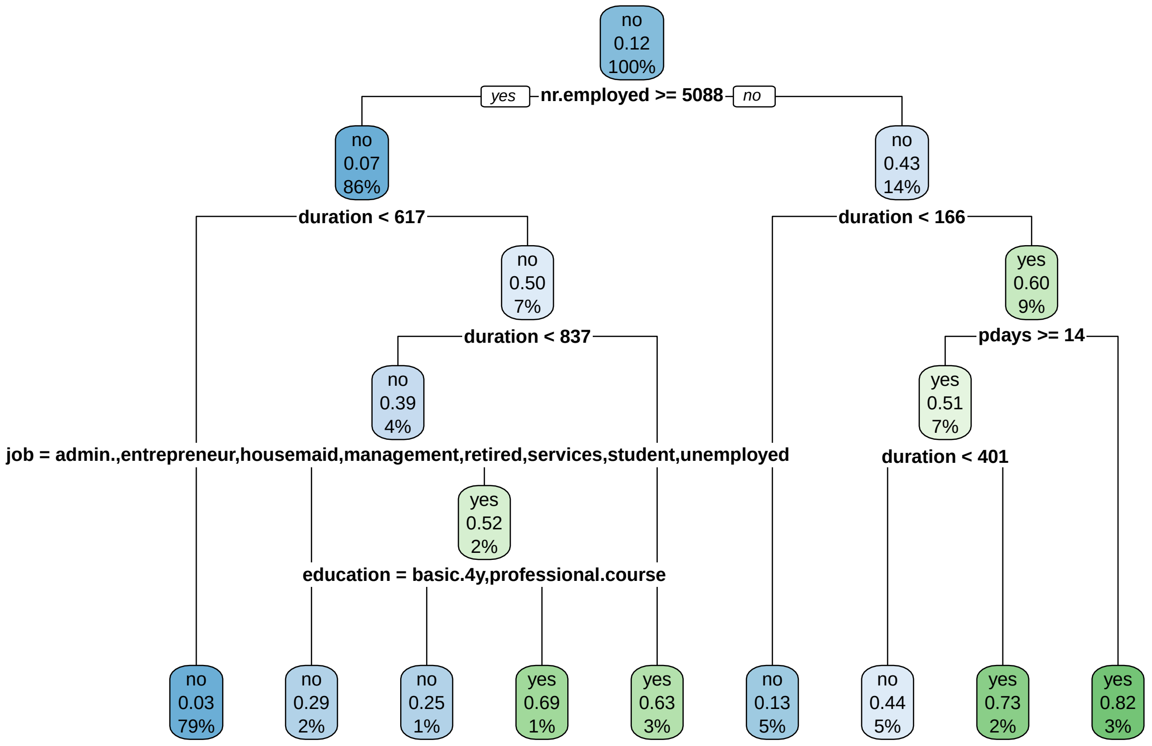
Here the rpart function computes the complete classification tree and the tree is displayed using the rpart.plot() function.

So, to classify our decision tree, we calculate our complexity parameter (CP) to select the optimal tree size. If the cost of adding another variable to the decision tree from the current node is above the value of CP, then tree building does not continue. Additionally, CP is used to validate our decision tree along with cross-validation error. To do so, we use the printcp() and plotcp() functions. We then examine and plot the resulting CP and other errors from each split



```
##
## Classification tree:
## rpart(formula = y ~ ., data = training_tree_data[, -rem_feat],
##     method = "class")
##
## Variables actually used in tree construction:
##  [1] cons.conf.idx contact       day_of_week   duration      education
##  [6] job           loan          month         nr.employed   pdays
## [11] poutcome
##
## Root node error: 370/3090 = 0.11974
##
## n= 3090
##
##         CP nsplit rel error  xerror     xstd
## 1 0.078378      0   1.00000 1.00000 0.048776
## 2 0.033784      2   0.84324 0.87297 0.045965
## 3 0.025676      4   0.77568 0.89730 0.046525
## 4 0.016216      6   0.72432 0.86216 0.045712
## 5 0.013514      8   0.69189 0.86216 0.045712
## 6 0.010811     13   0.62162 0.86216 0.045712
## 7 0.010000     18   0.56757 0.86216 0.045712
```

From the output of the printcp() function, we can determine the optimal CP value for our model by selecting the least amount of “xerror” which is the cross-validation error. We can see that the best CP has CP = 0.013514, xerror = 0.84054, and number of splits = 8. This CP is chosen for our model.



Tree Interpretation

Here similar to the previous rpart() function above, we compute the new tree classification with 8 splits.

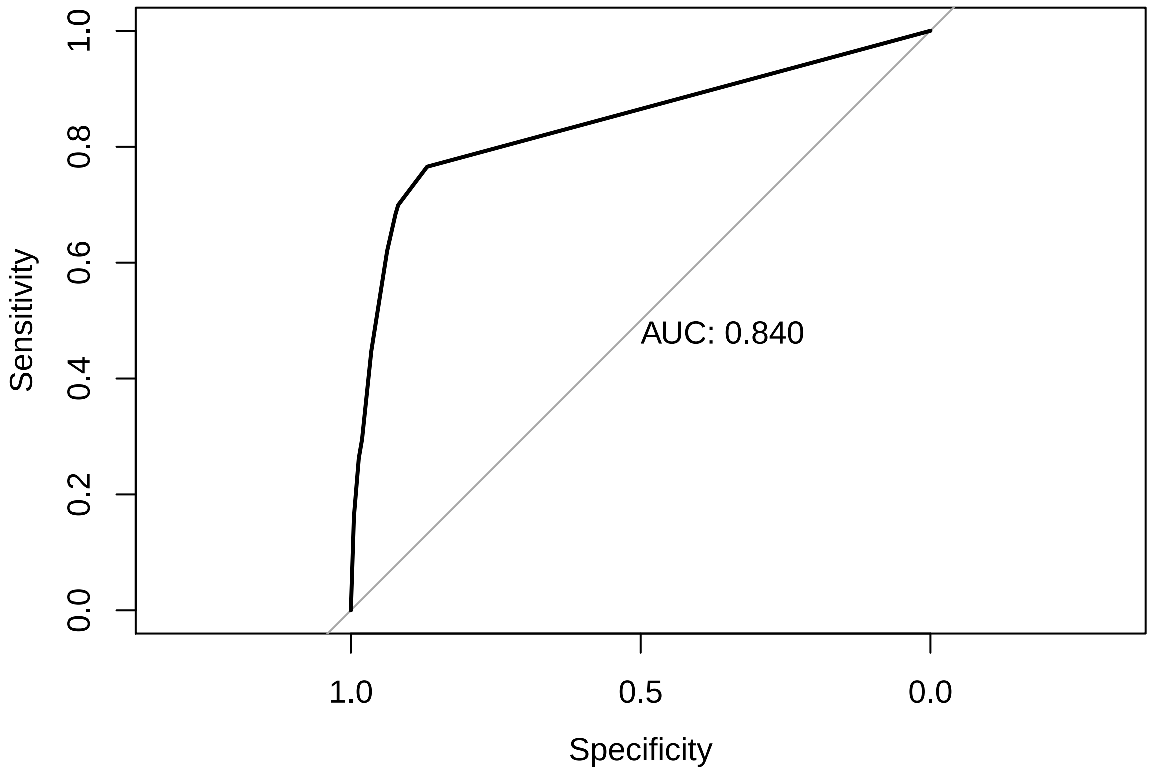
We find that the clients who are most likely to sign onto a long term deposit are those who have more than 5088 employed and who were in contact with for a duration greater than 14 minutes, but if their duration is less than 14 minutes but greater than 10 minutes and they are not employed in admin, housemaid, management, retired, service job, a student, or unemployed, and do not have an education that is basic, 4years, or professional course they are more likely to respond yes to the response variable. Of the people who have less than 5088 employed they are more likely to respond yes if their pdays (number of days since the last contact) is less than 14, or if it is then they are more likely if their duration is greater than 7 minutes.

```
## Model Accuracy
##      0.8992718
```

Here we use the predict function to return the predicted response of our Decision Tree model. We compare and compute these responses to our complete dataset to find our final accuracy which turns out to be 0.899, which indicates high accuracy.

ROC and AUC for Classification Tree

Now, we find the ROC (Receiver Operating Characteristics) curve. For that, we use our predicted response calculated from earlier and apply our roc() function. From this, we obtain our ROC curve.



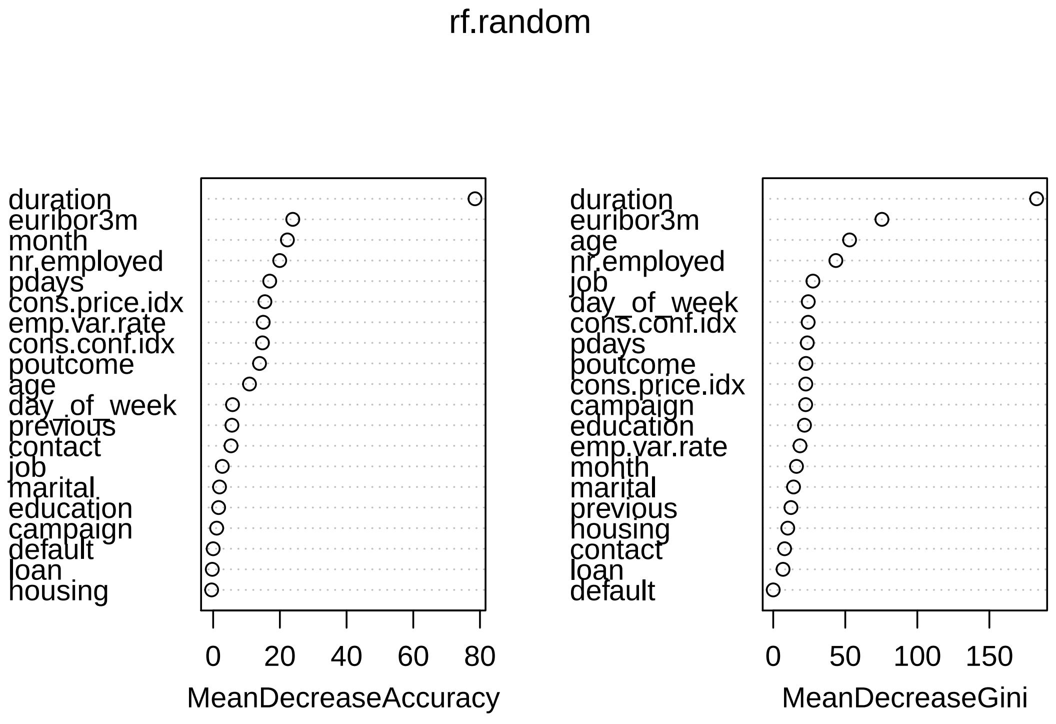
Since the ROC plot shows the trade-off between sensitivity and specificity, it curves closer to the top - left of the graph thus indicating a good performance. This information is complemented by the Area Under the Curve (AUC) value, which in this case, is 0.84 on a scale of 0 to 1. This reflects decent accuracy.

Random Forest

Fitting a Random Forest

For Random Forest, we do not require a feature selection procedure as the algorithm sorts the predictors based on importance by itself. Here, we run our function, randomForest(), to fit the model. We input our binary response variable along with all the observations from our training dataset. Below we have a few outputs derived from our fitted model:

```
##                         no        yes MeanDecreaseAccuracy MeanDecreaseGini
## age             13.3445033 -1.0730366           10.8892936     5.298391e+01
## job              2.7025228  0.9556516            2.7418424     2.756835e+01
## marital          4.0099117 -2.6678138            1.9047996     1.409760e+01
## education        1.8722817  0.1032315            1.6370515     2.171381e+01
## default          0.0000000  0.0000000            0.0000000     1.205362e-03
## housing          1.9883079 -3.8392276           -0.4670191     1.009217e+01
## loan             0.1361756 -0.6832927           -0.2702844     6.820584e+00
## contact         -2.2612750 13.8218175            5.3594699     7.893172e+00
## month           22.3356509 -1.6679959           22.2601610     1.613709e+01
## day_of_week      7.6450590 -0.6961552            5.8024263     2.432759e+01
## duration        48.2817450 72.4891370           78.5097046     1.827752e+02
## campaign         1.0879884  0.2714298            1.0635706     2.251930e+01
## pdays            5.2675648 19.3422532           16.9556172     2.363724e+01
## previous         5.9805232  0.4577690            5.6281052     1.231978e+01
## poutcome         7.1739086 13.6854408           13.8970177     2.273163e+01
## emp.var.rate    14.1680543  9.5032672           15.0113654     1.859492e+01
## cons.price.idx  15.8715002 -3.0992104           15.5088148     2.260720e+01
## cons.conf.idx   13.9105600  2.2031425           14.7799168     2.424884e+01
## euribor3m       22.5662843  1.6751937           23.8664274     7.543782e+01
## nr.employed     16.5353534 16.5694302           19.9525561     4.347839e+01
```

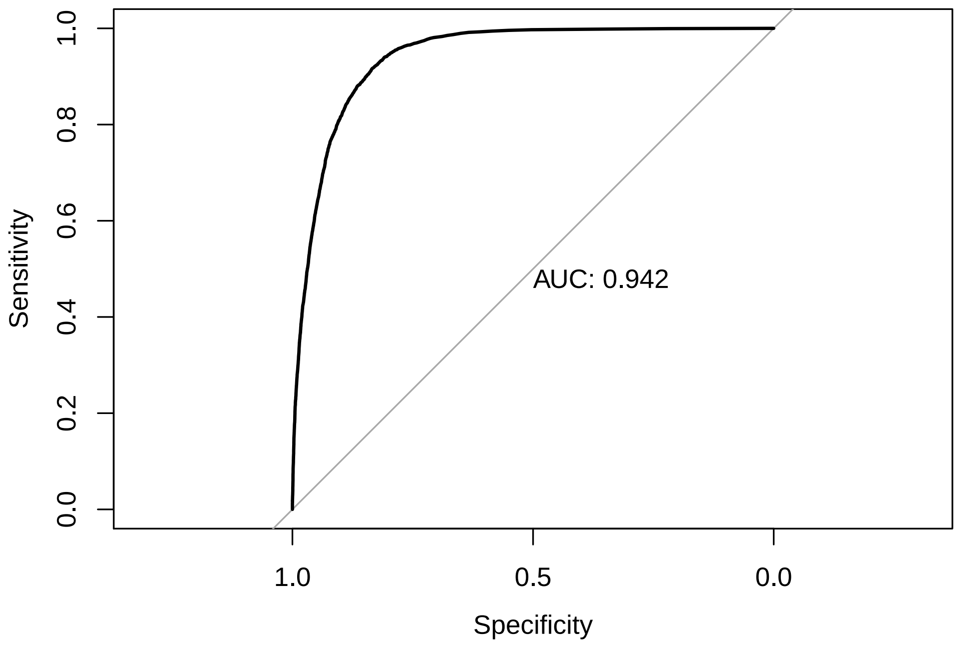


We apply importance() and varImpPlot() functions to assess and plot the usefulness of certain predictors. They are ranked in terms of most important to least. Here you can see that predictor "duration" is the most important according to the Mean Decrease Accuracy (MDA) as well as the Mean Decrease Gini (MDG) graphs. MDA gives a rough estimate of the loss in prediction performance when that particular variable is omitted from the training set whereas MDG is a measure of node impurity. According to this, "duration" has a high loss in prediction performance thereby becoming an essential predictor. On the other hand, it is high on the MDG graph implying a high node impurity, i.e. it is a good feature for the data to be split on thereby playing an important role in classification.

```
##
## Call:
##  randomForest(formula = as.factor(y) ~ ., data = training_tree_data,      importance = T)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 9.9%
## Confusion matrix:
##       no yes class.error
## no  2623  97  0.03566176
## yes  209 161  0.56486486
```

Moreover, from the output of our print() function, we can see an Out-Of-Bag (OOB) error estimate which is 9.9 percent. This is a low value for an error estimate implying a highly accurate model of about 90.1 percent accuracy.

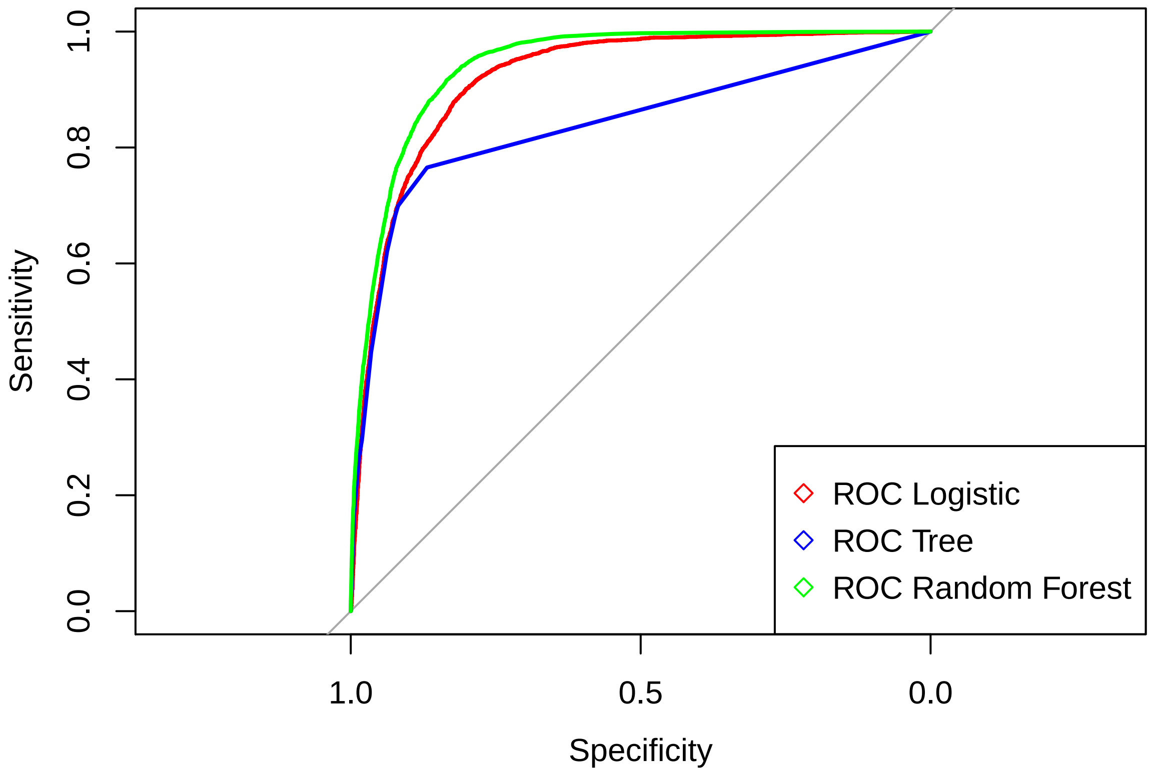
ROC and AUC for Random Forest



Since the ROC plot shows the trade-off between sensitivity and specificity, it curves closer to the top - left of the graph thus indicating better performance. This information is complemented by the Area Under the Curve (AUC) value, which is 0.942 on a scale of 0 to 1. This reflects high precision and a great ability for our model to predict our output variable.

ROC and AUC Model Comparisons

In this section, we will go through the process of selecting an optimal model from the ones that we have implemented. The statistic we will be using for comparisons will be the AUC(Area under the curve) extracted from our ROC(Receiver Operator Characteristics) curves. To reiterate ROC is what is known as a probability curve, while AUC can be represented as a measurement of separability. The higher the AUC the better the model is at predicting binary outcomes. Now that we understand our test measurements let's take a look at the collection of ROC curves that have been produced throughout the analysis.



For convenience we will also output the AUC values for each curve below:

```
##          Logistic AUC                Tree AUC      Random Forest AUC
##             0.9200604               0.8400715              0.9418105
```

The higher the AUC value, the better a model is at distinguishing between classes. In our case, the AUC values collected tell us how well the models can distinguish between whether or not a client will sign onto a long term deposit. We are comparing three different classification models. The first is logistic, which was fitted with the predictors chosen from our LASSO implementation. Our second is tree classification, which was also fit using the predictors chosen from LASSO. The third is Random Forest which uses its method for feature reduction. We are looking for a model that has the highest AUC value. According to our AUC values the model that is best fit to predict our client outcomes would be Random Forest, thus based on AUC, our model of choice will be the Random Forest model.

Results and Conclusion

The overall objective of this analysis was to determine if we could construct a model that could accurately predict client decisions on long term deposit accounts. We took advantage of algorithms to achieve feature selection to avoid overfitting and we utilized different libraries and functions to build three different predictive statistical models. Two of the models used predictors chosen from the LASSO method, while the third performed its own feature selection. We were able to construct ROC curves for each of these statistical models which in turn gave us the AUC values needed to compare the accuracy of our models. We concluded that our random forest model was the most accurate out of all three, having an AUC value of 0.9418105.

Extra Credit Neural Networks

Neural Network

Extra Credit Neural Networks

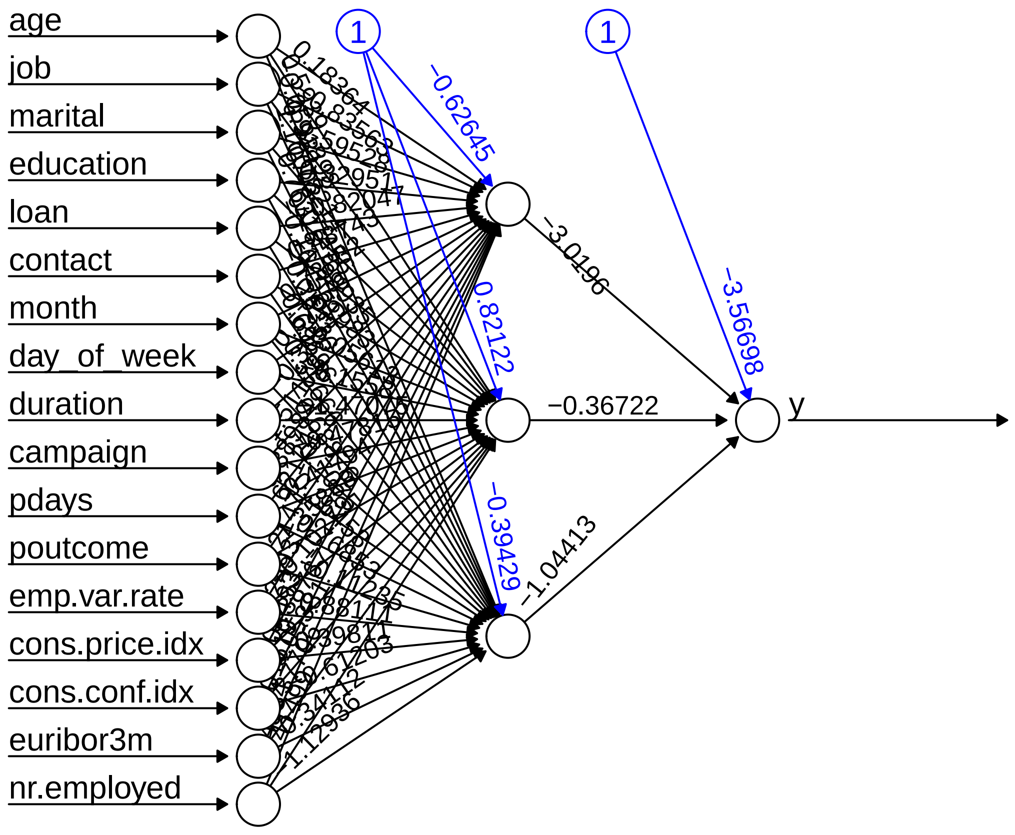
Neural Network

This portion of the project has been reserved for a method of analysis not taught within the course. The method chosen to be implemented here is neural networks.

Neural networking is an unsupervised machine learning model that employs a Backpropagation or feed-forward algorithm that is described by some to go “beyond regression”. A neural network is an information processing structure that connects elements. Each element has a connection that branches out into as many connections as is necessary. A neural network is a universal function approximator, so we can use it for any function with more prediction power compared to our logistic regression model. In this specific study, we can use neural networks to predict the willingness of clients to sign on to a long-term deposit based on many predictors. We chose this as an alternative method of a regression model with more power that works efficiently in this scenario due to the large training data provided (Bergmeir & Benitez, 2012).

Fitting the Neural Network

We first fit the model using the function neuralnet. Within this function, we’re fitting our training data along with the predictors chosen from LASSO. We set the linear.output = FALSE and our act.fct = logistic because we are working with binary outputs and we want our predict function to return probabilities. Below we have plotted the produced neural network. Along with the model accuracy.



```
## Model Accuracy
##      0.8734256
```

On the far left of our plot, we have our predictors as input. The arrows with numerical value are weights and can be seen as how much the variable contributes to the next node. The blue node and arrows contribute to the bias weights. The nodes in the middle of the plot are what is considered our hidden nodes, which are used for the learning process of our neural network. The last node on the neural network is our response variables or our output. We can also see from above that our model accuracy is quite good at 87.34% accuracy, yet when compared to the accuracy of the other models used in this analysis it falls a little short. So we can conclude that although the model is still fairly accurate, in terms of our analysis and data, one of our other models would be preferred.

References

Bergmeir, C., & Benitez, J. M. (2012). Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS. Journal of Statistical Software, 46(7). https://doi.org/10.18637/jss.v046.i07.