Direct Marketing Campaigns Regression Anaylsis

Jessica Gorr

October 24, 2024

Table of Contents

# Introduction

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required. The data collected from these marketing capaigns was collected from May 2008 to November 2010. There is a total number of 45,211 observation in this data set. The data set consists of 17 variables, including the response variable with the name ‘y’. A description of the predictor and outcome features are below:

1 - age (numeric)

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

3 - marital : Marital status (categorical): “married”, “divorced”, “single”

4 - education (categorical): “unknown”,“secondary”,“primary”,“tertiary”

5 - default: Does the client have credit in default? (binary: “yes”,“no”)

6 - balance: Average yearly balance (numeric, in euros)

7 - housing: Does the client have a housing loan? (binary: “yes”,“no”)

8 - loan: Does the client have a personal loan? (binary: “yes”,“no”)

9 - contact: Contact communication type (categorical): “unknown”,“telephone”,“cellular”

10 - day: Last contact day of the month (numeric, discrete)

11 - month: Last contact month of year (categorical): “jan”, “feb”, “mar”, “apr”, “may”, “jun”, “jul”, “aug”, “sep”, “oct”, “nov”, “dec”

12 - duration: Last contact duration (numeric, in seconds)

13 - campaign: The number of contacts performed during this campaign and for this client (numeric, discrete)

14 - pdays: The number of days after the client was last contacted from a previous campaign (numeric, discrete)

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

16 - poutcome: The outcome of the previous marketing campaign (categorical): “unknown”, “other”, “failure”, “success”

17 - y oOutcome class variable): Has the client subscribed a term deposit? (binary: “yes”,“no”)

A public link to the data can be found here: <https://archive.ics.uci.edu/dataset/222/bank+marketing>

When looking at this dataset, two areas we would want to explore through linear and logistic regression.

1. If there is a association with any of the variables and how long of the duration of a call (Linear).
2. Predict if a client has subscribed a term deposit after direct marketing campaigns (Logistic).

#Load the sample data  
BankMarketing = read.csv("https://pengdsci.github.io/datasets/BankMarketing/BankMarketingCSV.csv")[, -1]

## Handling Missing Values

This dataset does not contain any missing values, therefore we do not have to drop any rows or input any values.

# EDA and Feature Engineering

In oder to perfom so EDA, we must have a basic understanding of the data. A summary of teh data is printed below.

#Summarized descriptive statistics for all variables in the data set  
summary(BankMarketing)

## age job marital education   
## Min. :18.00 Length:45211 Length:45211 Length:45211   
## 1st Qu.:33.00 Class :character Class :character Class :character   
## Median :39.00 Mode :character Mode :character Mode :character   
## Mean :40.94   
## 3rd Qu.:48.00   
## Max. :95.00   
## default balance housing loan   
## Length:45211 Min. : -8019 Length:45211 Length:45211   
## Class :character 1st Qu.: 72 Class :character Class :character   
## Mode :character Median : 448 Mode :character Mode :character   
## Mean : 1362   
## 3rd Qu.: 1428   
## Max. :102127   
## contact day month duration   
## Length:45211 Min. : 1.00 Length:45211 Min. : 0.0   
## Class :character 1st Qu.: 8.00 Class :character 1st Qu.: 103.0   
## Mode :character Median :16.00 Mode :character Median : 180.0   
## Mean :15.81 Mean : 258.2   
## 3rd Qu.:21.00 3rd Qu.: 319.0   
## Max. :31.00 Max. :4918.0   
## campaign pdays previous poutcome   
## Min. : 1.000 Min. : -1.0 Min. : 0.0000 Length:45211   
## 1st Qu.: 1.000 1st Qu.: -1.0 1st Qu.: 0.0000 Class :character   
## Median : 2.000 Median : -1.0 Median : 0.0000 Mode :character   
## Mean : 2.764 Mean : 40.2 Mean : 0.5803   
## 3rd Qu.: 3.000 3rd Qu.: -1.0 3rd Qu.: 0.0000   
## Max. :63.000 Max. :871.0 Max. :275.0000   
## y   
## Length:45211   
## Class :character   
## Mode :character   
##   
##   
##

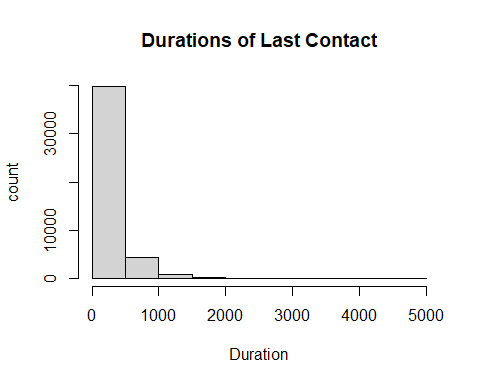
We can see from the above summary table that the distribution of some of the numeric variables is skewed and contains outliers that need to be further explored.

## Single Variable Distribution

The distribution for our continuous numerical variables for average\_monthly\_hours and time\_spend\_company are shown below. Time spent is skewed to the right, and is not normal. Average monthly hours so not too bad, and can be deemed approxiatly normal.

In order to examine and determine the outcome of any of the outliers and looking at the skewness of certain numerical variables, such as “duration”, discretization will be used to divide the different categorical varibles into groups. This variable should be discretized due to thenumber of high outliers it contains. In looking at this variable’s distribution, the three groups that were created (0-180, 181-319, and 320+) seem similar enough in the frequency of client observations. This new variable will now be used for building future models. The histogram can be seen below.

# histogram showing the distribution of the duration variable  
hist(BankMarketing$duration, xlab = "Duration", ylab = "count", main = "Durations of Last Contact")



# New grouping variable for duration  
BankMarketing$grp.duration <- ifelse(BankMarketing$duration <= 180, '0-180',  
 ifelse(BankMarketing$duration >= 320, '320+', '181-319'))

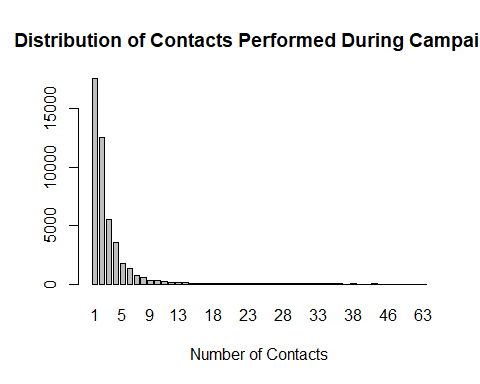
Now we want to look at some of the categorical and binary features. When looking at the distribution of contacts performed during the campain, it is Skewed to the right. This means that groups should be created. For campaign, the value of 1 contact should be its own group since it has the highest frequency of observations. Values 2 & 3 number of contacts combined have close to the same frequency, so they should be paired together in their own group. The remaining observations should be combined into the final group.

When looking at pdays, the value of -1 is an indicator that a client was not previously contacted before the campaign. Since this makes up most of the obersvations, it will become its own group. The rest of the observations were split into groups of 1-200 days and 200 days or more.

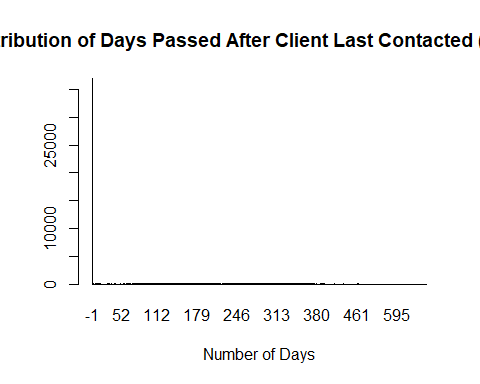
The previous variable was also split into 3 groups. The value of 0 contacts is one group since it has the most observations. The values of 1 to 3 contacts is another category since they both make a fair amount of the observations. Same goes for observations with 4 or more contacts.

All of these bar plots can be seen below.

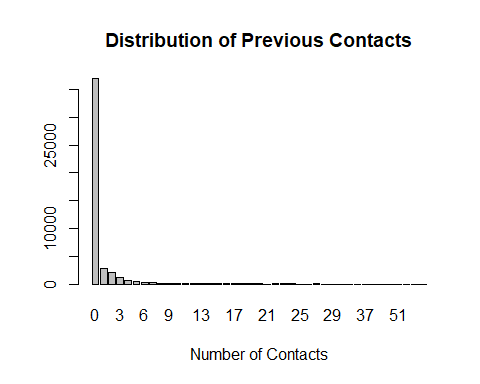
# barplot showing the distribution of the campaign variable  
marketcampaigns = table(BankMarketing$campaign)  
barplot(marketcampaigns, main = "Distribution of Contacts Performed During Campaign", xlab = "Number of Contacts")



# barplot showing the distribution of the pdays variable  
dayspassed = table(BankMarketing$pdays)  
barplot(dayspassed, main = "Distribution of Days Passed After Client Last Contacted (Pdays) ", xlab = "Number of Days")



# barplot showing the distribution of the previous variable  
prev = table(BankMarketing$previous)  
barplot(prev, main = "Distribution of Previous Contacts", xlab = "Number of Contacts")

 These new grouped variables will be used in future model build. The categories for each variable are as follows:

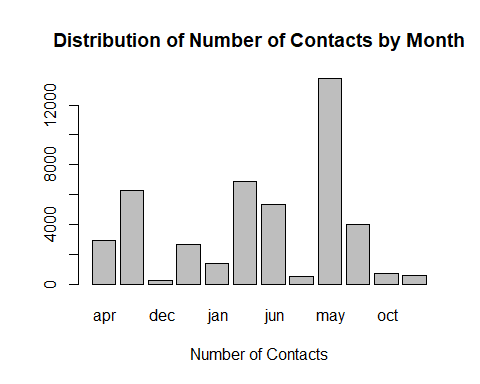
campaign: 1, 2-3, 4+ pdays: -1, 1-199, 200+ previous: 0, 1-3, 4+

# New grouping variable for campaign  
BankMarketing$grp.campaign <- ifelse(BankMarketing$campaign <= 1, '1',  
 ifelse(BankMarketing$campaign >= 4, '4+', '2-3'))  
  
# New grouping variable for pdays  
BankMarketing$grp.pdays <- ifelse(BankMarketing$pdays <= -1, 'Client Not Previously Contacted', ifelse(BankMarketing$pdays >= 200, '200+', '1-199'))  
  
# New grouping variable for previous  
BankMarketing$grp.previous <- ifelse(BankMarketing$previous <= 0, '0',  
 ifelse(BankMarketing$previous > 4, '4+', '1-3'))

Now we move onto categorical varibles.

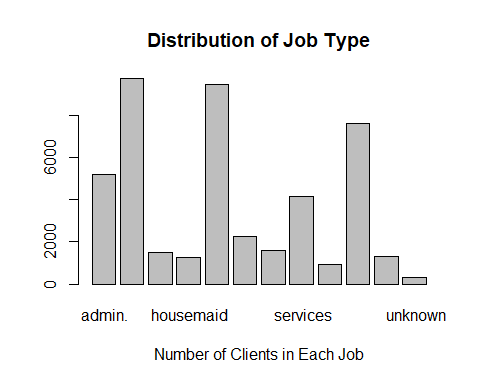
The categorical variable of month has also been discretized by seasons since the bar plot below is also skewed for certain months. Also, handling smaller groups into seasons is easier than hanling 12 months. A new feature called “seasons” was created.

# barplot showing the distribution of the month variable  
seasons = table(BankMarketing$month)  
barplot(seasons, main = "Distribution of Number of Contacts by Month", xlab = "Number of Contacts")



# New grouping variable for month  
BankMarketing$grp.month = ifelse(BankMarketing$month == " jan", "winter", ifelse(BankMarketing$month == " feb", "winter", ifelse(BankMarketing$month == " mar", "spring", ifelse(BankMarketing$month == " apr", "spring", ifelse(BankMarketing$month == " may", "spring", ifelse(BankMarketing$month == " jun", "summer", ifelse(BankMarketing$month == " jul", "summer", ifelse(BankMarketing$month == " aug", "summer", ifelse(BankMarketing$month == " sep", "fall", ifelse(BankMarketing$month == " oct", "fall", ifelse(BankMarketing$month == " nov", "fall", "winter")))))))))))

# barplot showing the distribution of the job variable  
jobcategory = table(BankMarketing$job)  
barplot(jobcategory, main = "Distribution of Job Type", xlab = "Number of Clients in Each Job")



# New grouping variable for job  
BankMarketing$grp.job = ifelse(BankMarketing$job == " unknown", "not working", ifelse(BankMarketing$job == " unemployed", "not working", ifelse(BankMarketing$job == " retired", "not working", ifelse(BankMarketing$job == " blue-collar", "workers", ifelse(BankMarketing$job == " entrepreneur", "bosses", ifelse(BankMarketing$job == " housemaid", "workers", ifelse(BankMarketing$job == " management", "bosses", ifelse(BankMarketing$job == " self-employed", "bosses", ifelse(BankMarketing$job == " services", "white-collar", ifelse(BankMarketing$job == " technician", "white-collar", ifelse(BankMarketing$job == " student", "not working", "white-collar")))))))))))

Now that we have Now that the variables have been discretized, and created new variables, the dataset now must be cleaned to reflect those changes.

# Assembling the discretized variables and other variables to make the modeling data set  
var.names = c("age", "balance", "day", "grp.job", "marital", "education", "default", "housing", "loan", "contact", "grp.month", "grp.duration", "grp.campaign", "grp.pdays", "grp.previous", "poutcome", "y")   
BankMarketingCampaign = BankMarketing[, var.names]

## Assessing Pairwised Relationship

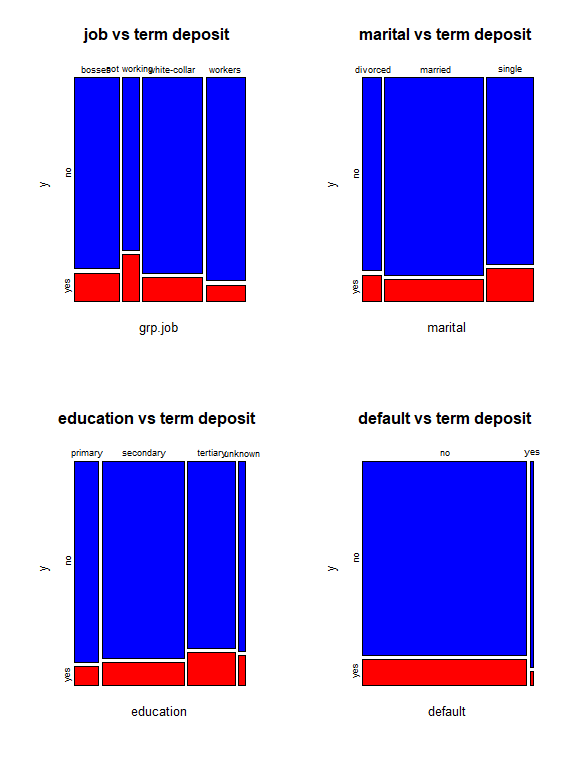
We want to see if there is any linear association between the numeric variables. A Pairwise scatter plot like below, is a good visual. This scatterplot blow looks at the data.

# Pair-wise scatter plot for numeric variables  
ggpairs(BankMarketingCampaign, # Data frame  
 columns = 1:3, # Columns  
 aes(color = y, # Color by group (cat. variable)  
 alpha = 0.5))

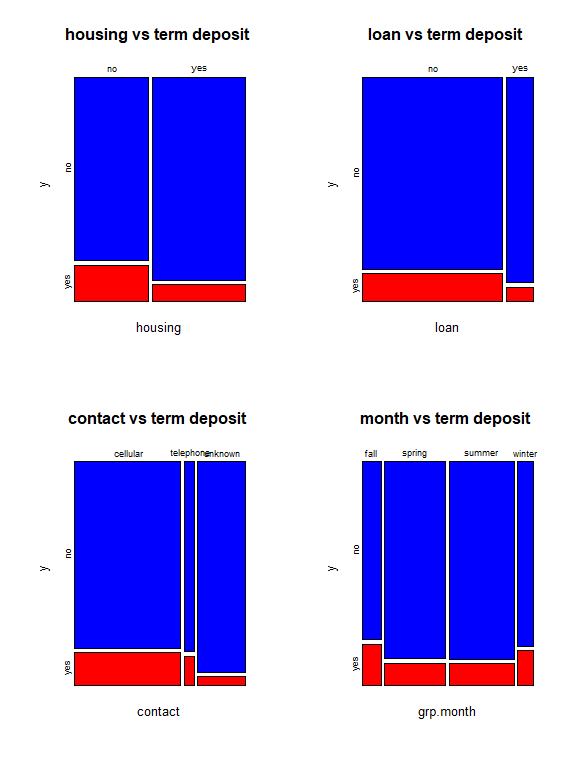
 We see that none of the numeric variables appear to be significantly correlated when looking at the numbers. But, the stacked density cures and no completely overlapping, indicating a week correlation between the numeric variables and our response ‘y’. The strongest but ’weak correlation we can see through the plots is age and balance.

Now that we looked at the numeric variables, we can turn the attention to categorical variables and dependency. In order to see dependency, mosaic plots will be shown below.

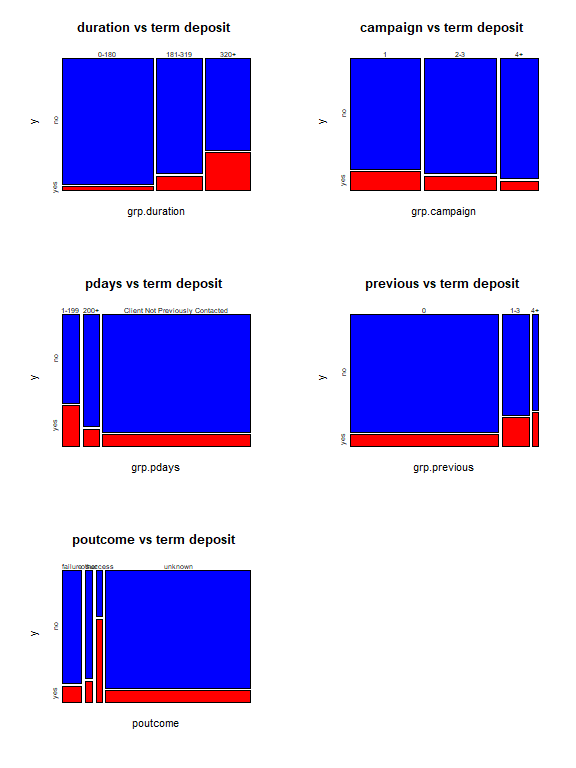
# Mosaic plots to show categorical variable dependency to the response.  
par(mfrow = c(2,2))  
mosaicplot(grp.job ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="job vs term deposit ")  
mosaicplot(marital ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="marital vs term deposit ")  
mosaicplot(education ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="education vs term deposit ")  
mosaicplot(default ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="default vs term deposit ")



# Mosaic plots to show categorical variable dependency to the response.  
par(mfrow = c(2,2))  
mosaicplot(housing ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="housing vs term deposit ")  
mosaicplot(loan ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="loan vs term deposit ")  
mosaicplot(contact ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="contact vs term deposit ")  
mosaicplot(grp.month ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="month vs term deposit ")



# Mosaic plots to show categorical variable dependency to the response.  
par(mfrow = c(3,2))  
mosaicplot(grp.duration ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="duration vs term deposit ")  
mosaicplot(grp.campaign ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="campaign vs term deposit ")  
mosaicplot(grp.pdays ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="pdays vs term deposit ")  
mosaicplot(grp.previous ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="previous vs term deposit ")  
mosaicplot(poutcome ~ y, data=BankMarketingCampaign,col=c("Blue","Red"), main="poutcome vs term deposit ")



# Predictive Modeling with Logistic Regression

The revised data from the EDA done above will be used to run different logistic regression models. The optimal final model will be found from canidate models, which will be used to calculate probabilities for predicting whether or not a client has subscribed after direct marketing campaigns.

We can use a logistical model for predicting whether or not a client has subscribed a term deposit after direct marketing campaigns.The variable, y, which tells whether a client has subscribed a term deposit, acts as the binary response variable of all the logistic models. The rest of the variables, including the new discretized variables, of the revised data set act as the predictor variables that will possibly affect the response ‘y’.

In order to perform proper modeling, some categorical and binary variables had to be changed, including the response (y) and the ones changed in the EDA, to have numerical labels, thereby making them easier to use for modeling.

The first logistic regression model that will be built is an initial full model that contains all predictors variables of the data set. Automatic variable selection will then be used to find a final model. In looking at the p-values of the variables in the initial model, those that are insignificant at the 0.05 level will be dropped.

The variables remaining that are either statistically significant or important for the model will be used to create a sort of reduced model. A third and final model, that is between the full and reduced models, will then be found. Performance of predictive power will be analyzed for all predictor variables as well as their association to the response.

Finally, this final model will be used to calculate predictive probability for values of the response variable. When values of predictor values are entered, the predicted value of whether or not a client has subscribed a term deposit (either Yes or No) is given.

## Turning features in Discrete Variables

All the categorical and binary varibles including the response must be changed into discrete numerical varibles. It will be as as follows:

y (response): 0=no, 1=yes

grp.job: 0=not working, 1=workers, 2=bosses, 3=white-collar

marital: 0=divorced, 1=single, 2=married

education: 0=unknown, 1=primary, 2=secondary, 3=tertiary

housing: 0=no, 1=yes

loan: 0=no, 1=yes

contact: 0=unknown, 1=telephone, 2=cellular

grp.month: 0=winter, 1=spring, 2=summer, 3=fall

grp.duration: 0=(0-180), 1=(181-319), 2=320+

grp.campaign: 0=1, 1=(2-3), 2=4+

grp.pdays: 0=Client Not Previously Contacted, 1=(1-199), 2=200+

grp.previous: 0=0, 1=(1-3), 2=4+

# Create numerical value labels for categorical variables  
BankMarketingCampaign$y <- factor(BankMarketingCampaign$y, levels = c(" no", " yes"), labels = c("0", "1"))  
  
BankMarketingCampaign$grp.job <- factor(BankMarketingCampaign$grp.job, levels = c("not working", "workers", "bosses", "white-collar"), labels = c("0", "1", "2", "3"))  
  
BankMarketingCampaign$marital <- factor(BankMarketingCampaign$marital, levels = c(" divorced", " single", " married"), labels = c("0", "1", "2"))  
  
BankMarketingCampaign$education <- factor(BankMarketingCampaign$education, levels = c(" unknown", " primary", " secondary", " tertiary"), labels = c("0", "1", "2", "3"))  
   
BankMarketingCampaign$housing <- factor(BankMarketingCampaign$housing, levels = c(" no", " yes"), labels = c("0", "1"))  
   
BankMarketingCampaign$loan <- factor(BankMarketingCampaign$loan, levels = c(" no", " yes"), labels = c("0", "1"))  
  
BankMarketingCampaign$contact <- factor(BankMarketingCampaign$contact, levels = c(" unknown", " telephone", " cellular"), labels = c("0", "1", "2"))  
  
BankMarketingCampaign$grp.month <- factor(BankMarketingCampaign$grp.month, levels = c("winter", "spring", "summer", "fall"), labels = c("0", "1", "2", "3"))  
  
BankMarketingCampaign$grp.duration <- factor(BankMarketingCampaign$grp.duration, levels = c("0-180", "181-319", "320+"), labels = c("0", "1", "2"))  
   
BankMarketingCampaign$grp.campaign <- factor(BankMarketingCampaign$grp.campaign, levels = c("1", "2-3", "4+"), labels = c("0", "1", "2"))  
   
BankMarketingCampaign$grp.pdays <- factor(BankMarketingCampaign$grp.pdays, levels = c("Client Not Previously Contacted", "1-199", "200+"), labels = c("0", "1", "2"))  
   
BankMarketingCampaign$grp.previous <- factor(BankMarketingCampaign$grp.previous, levels = c("0", "1-3", "4+"), labels = c("0", "1", "2"))

## Create Candidate Models

The full/initial model containing all of the predictor variables will be made first, with ‘y’ (whether or not a client has subscribed a term deposit) as the response. The variables balance and default are not included since our EDA showed that removing them from the model might help the results.

# Create the initial full model  
# Create the initial full model  
initial.model = glm(y ~ age + day + grp.job + marital + education + housing + loan + contact +grp.duration +grp.campaign +grp.pdays +grp.previous, family = binomial, data = BankMarketingCampaign)  
coefficient.table = summary(initial.model)$coef  
kable(coefficient.table, caption = "Significance tests of logistic regression model")

Significance tests of logistic regression model

|  | Estimate | Std. Error | z value | Pr(>|z|) |
| --- | --- | --- | --- | --- |
| (Intercept) | -3.5018560 | 0.1521815 | -23.011052 | 0.0000000 |
| age | 0.0037427 | 0.0017733 | 2.110553 | 0.0348108 |
| day | -0.0055270 | 0.0020224 | -2.732896 | 0.0062780 |
| grp.job1 | -0.5356687 | 0.0623519 | -8.591057 | 0.0000000 |
| grp.job2 | -0.4527472 | 0.0599608 | -7.550719 | 0.0000000 |
| grp.job3 | -0.3669178 | 0.0546304 | -6.716368 | 0.0000000 |
| marital1 | 0.1692006 | 0.0610871 | 2.769826 | 0.0056086 |
| marital2 | -0.1806493 | 0.0536963 | -3.364279 | 0.0007674 |
| education1 | -0.2609743 | 0.0938475 | -2.780833 | 0.0054220 |
| education2 | -0.1221068 | 0.0829586 | -1.471899 | 0.1410481 |
| education3 | 0.1775338 | 0.0866458 | 2.048960 | 0.0404660 |
| housing1 | -0.7303848 | 0.0366351 | -19.936773 | 0.0000000 |
| loan1 | -0.5591949 | 0.0540381 | -10.348154 | 0.0000000 |
| contact1 | 0.9550897 | 0.0817113 | 11.688589 | 0.0000000 |
| contact2 | 1.0053064 | 0.0523061 | 19.219696 | 0.0000000 |
| grp.duration1 | 1.3339942 | 0.0503148 | 26.512978 | 0.0000000 |
| grp.duration2 | 2.7044986 | 0.0458332 | 59.007422 | 0.0000000 |
| grp.campaign1 | -0.3253327 | 0.0364818 | -8.917665 | 0.0000000 |
| grp.campaign2 | -0.5201170 | 0.0505253 | -10.294181 | 0.0000000 |
| grp.pdays1 | 1.4641585 | 0.0758033 | 19.315236 | 0.0000000 |
| grp.pdays2 | 0.6043423 | 0.0869219 | 6.952701 | 0.0000000 |
| grp.previous1 | -0.2153398 | 0.0777604 | -2.769273 | 0.0056181 |

We can see that there are some insignificant predictor variables, and they should be dropped from the model to create a reduced model. Using the step() function, we will now find reduced and final models. The final best model will be a model that is between the full and reduced models.

# Creating the reduced and final models  
full.model = initial.model # the \*biggest model\* that includes all predictor variables  
reduced.model = glm(y ~ day + grp.job + marital + housing + loan + contact + grp.duration + grp.campaign + grp.previous, family = binomial, data = BankMarketingCampaign)  
final.model = step(full.model,   
 scope=list(lower=formula(reduced.model),upper=formula(full.model)),  
 data = BankMarketingCampaign,   
 direction = "backward",  
 trace = 0) # trace = 0: suppress the detailed selection process  
final.model.coef = summary(final.model)$coef  
kable(final.model.coef, caption = "Summary table of significant tests")

Summary table of significant tests

|  | Estimate | Std. Error | z value | Pr(>|z|) |
| --- | --- | --- | --- | --- |
| (Intercept) | -3.5018560 | 0.1521815 | -23.011052 | 0.0000000 |
| age | 0.0037427 | 0.0017733 | 2.110553 | 0.0348108 |
| day | -0.0055270 | 0.0020224 | -2.732896 | 0.0062780 |
| grp.job1 | -0.5356687 | 0.0623519 | -8.591057 | 0.0000000 |
| grp.job2 | -0.4527472 | 0.0599608 | -7.550719 | 0.0000000 |
| grp.job3 | -0.3669178 | 0.0546304 | -6.716368 | 0.0000000 |
| marital1 | 0.1692006 | 0.0610871 | 2.769826 | 0.0056086 |
| marital2 | -0.1806493 | 0.0536963 | -3.364279 | 0.0007674 |
| education1 | -0.2609743 | 0.0938475 | -2.780833 | 0.0054220 |
| education2 | -0.1221068 | 0.0829586 | -1.471899 | 0.1410481 |
| education3 | 0.1775338 | 0.0866458 | 2.048960 | 0.0404660 |
| housing1 | -0.7303848 | 0.0366351 | -19.936773 | 0.0000000 |
| loan1 | -0.5591949 | 0.0540381 | -10.348154 | 0.0000000 |
| contact1 | 0.9550897 | 0.0817113 | 11.688589 | 0.0000000 |
| contact2 | 1.0053064 | 0.0523061 | 19.219696 | 0.0000000 |
| grp.duration1 | 1.3339942 | 0.0503148 | 26.512978 | 0.0000000 |
| grp.duration2 | 2.7044986 | 0.0458332 | 59.007422 | 0.0000000 |
| grp.campaign1 | -0.3253327 | 0.0364818 | -8.917665 | 0.0000000 |
| grp.campaign2 | -0.5201170 | 0.0505253 | -10.294181 | 0.0000000 |
| grp.pdays1 | 1.4641585 | 0.0758033 | 19.315236 | 0.0000000 |
| grp.pdays2 | 0.6043423 | 0.0869219 | 6.952701 | 0.0000000 |
| grp.previous1 | -0.2153398 | 0.0777604 | -2.769273 | 0.0056181 |

## Prediction

## Predictive Probability Analysis for Clients Subscribing Term Deposits

Now that a final model has been created, we can predict whether or not a client has subscribed a term deposit based on given values of the predictor variables in the final model associated with two clients. A threshold probability of 0.5 is used to predict the response value.

# Predicting Response Value for Banking Client Given Variable Values for the Final Model  
mynewdata = data.frame(age=c(34,64),  
 day = c(3,5),  
 grp.job = c("1","1"),  
 marital = c("1","1"),  
 education = c("1","3"),  
 housing = c("1","1"),  
 loan = c("0","0"),  
 contact = c("1","0"),  
 grp.month = c("3","1"),  
 grp.duration = c("2","1"),  
 grp.campaign = c("0","1"),  
 grp.pdays = c("2","2"),  
 grp.previous = c("1","0"))  
pred.success.prob = predict(final.model, newdata = mynewdata, type="response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

## threshold probability  
cut.off.prob = 0.5  
pred.response = ifelse(pred.success.prob > cut.off.prob, 1, 0) # This predicts the response  
  
# Add the new predicted response to Mynewdata  
mynewdata$Pred.Response = pred.response  
kable(mynewdata, caption = "Predicted Value of Response with given cut-off")

Predicted Value of Response with given cut-off

| age | day | grp.job | marital | education | housing | loan | contact | grp.month | grp.duration | grp.campaign | grp.pdays | grp.previous | Pred.Response |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 34 | 3 | 1 | 1 | 1 | 1 | 0 | 1 | 3 | 2 | 0 | 2 | 1 | 0 |
| 64 | 5 | 1 | 1 | 3 | 1 | 0 | 0 | 1 | 1 | 1 | 2 | 0 | 0 |

Looking at the predicted response, we can see that neither of these clients will suscribe to the marketing campaign. ## Model Selection

Now that we have our canidate models. It is time to perform some model selction using the ROC curve and AUC.

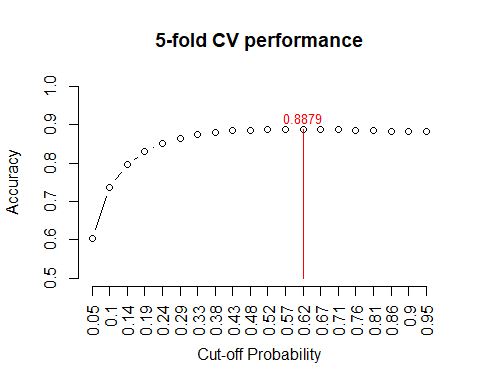
Since our sample is relatively large, we will randomly split the overall data set into two data sets. 70% of the data will be put in a training data set for training and validating models. The other 30% goes into a testing data set used for testing the final model. The value labels of the response (yes/no) used for testing and validation data will be removed when calculating the accuracy measures later.

## Recode response variable: yes = 1 and no = 0  
yes.id = which(BankMarketingCampaign$y == "1")   
no.id = which(BankMarketingCampaign$y == "0")  
  
## Creating the training and testing data sets  
BankMarketingCampaign$y.subscribe = 1  
BankMarketingCampaign$y.subscribe[no.id] = 0  
var.names = c("age", "day", "grp.job", "marital","education","housing","loan","contact","grp.month", "grp.duration","grp.campaign","grp.pdays","grp.previous", "y.subscribe" )  
BankMarketingCampaign = BankMarketingCampaign[, var.names]  
nn = dim(BankMarketingCampaign)[1]  
train.id = sample(1:nn, round(nn\*0.7), replace = FALSE)   
####  
training = BankMarketingCampaign[train.id,]  
testing = BankMarketingCampaign[-train.id,]

## Cut-off Probability Search and Accuracy Score

In order to find an optimal cut-off probability, a sequence of 20 candidate cut-off probabilities will be defined. Then, a 5-fold cross-validation will be performed to find the optimal cut-off probability of the final model. All three models created will be used to find the optimal cut-off. This is shown below.

n0 = dim(training)[1]/5  
  
# candidate cut off prob  
cut.0ff.prob = seq(0,1, length = 22)[-c(1,22)]  
  
# null vector for storing prediction accuracy  
pred.accuracy = matrix(0,ncol=20, nrow=5, byrow = T)  
  
## 5-fold CV  
for (i in 1:5){  
 valid.id = ((i-1)\*n0 + 1):(i\*n0)  
 valid.data = training[valid.id,]  
 train.data = training[-valid.id,]  
 train.model = glm(y.subscribe ~ age + day + grp.job + marital + education + housing + loan + contact + grp.duration + grp.campaign + grp.pdays + grp.previous, family = binomial(link = logit), data = train.data)  
####  
 pred.prob = predict.glm(train.model, valid.data, type = "response")  
 # define confusion matrix and accuracy  
 for(j in 1:20){  
 #pred.subscribe = rep(0,length(pred.prob))  
 valid.data$pred.subscribe = as.numeric(pred.prob > cut.0ff.prob[j])  
 a11 = sum(valid.data$pred.subscribe == valid.data$y.subscribe)  
 pred.accuracy[i,j] = a11/length(pred.prob)  
 }  
}  
##  
avg.accuracy = apply(pred.accuracy, 2, mean)  
max.id = which(avg.accuracy ==max(avg.accuracy))  
  
### visual representation  
tick.label = as.character(round(cut.0ff.prob,2))  
plot(1:20, avg.accuracy, type = "b",  
 xlim=c(1,20),   
 ylim=c(0.5,1),   
 axes = FALSE,  
 xlab = "Cut-off Probability",  
 ylab = "Accuracy",  
 main = "5-fold CV performance"  
 )  
axis(1, at=1:20, label = tick.label, las = 2)  
axis(2)  
segments(max.id, 0.5, max.id, avg.accuracy[max.id], col = "red")  
text(max.id, avg.accuracy[max.id]+0.03, as.character(round(avg.accuracy[max.id],4)), col = "red", cex = 0.8)



5-fold CV performance plot

The above figure indicates that the optimal cut-off probability that yields the best accuracy is 0.52.

test.model = glm(y.subscribe ~ age + day + grp.job + marital + education + housing + loan + contact + grp.month + grp.duration + grp.campaign + grp.pdays + grp.previous, family = binomial(link = logit), data = training)  
newBankingTestingData = data.frame(age= testing$age, day= testing$day, grp.job= testing$grp.job, marital= testing$marital, education= testing$education, housing= testing$housing, loan= testing$loan, contact= testing$contact, grp.month= testing$grp.month, grp.duration= testing$grp.duration, grp.campaign= testing$grp.campaign, grp.pdays= testing$grp.pdays, grp.previous= testing$grp.previous)  
  
pred.prob.test = predict.glm(test.model, newBankingTestingData, type = "response")  
  
## Assessing Model Accuracy  
testing$test.subscribe = as.numeric(pred.prob.test > 0.22)  
a11 = sum(testing$test.subscribe == testing$y.subscribe)  
test.accuracy = a11/length(pred.prob.test)  
kable(as.data.frame(test.accuracy), align='c')

| test.accuracy |
| --- |
| 0.8355821 |

Here in our accuracy test we find that it is accurate 84.1% of the time. This indcates there is no under-fitting for our model.

## Local and Global ROC Metrics

Using the optimal cut-off probability of 0.57, which was found above, we will now report the local measures using our testing data. This includes specificity and sensitivity based on each of these cut-offs for the 20 sub-intervals.

# Looking at sensitivity and specificity performance measurements  
testing$test.subscribe = as.numeric(pred.prob.test > 0.52)  
### components for defining various measures  
p0.a0 = sum(testing$test.subscribe ==0 & testing$y.subscribe ==0)  
p0.a1 = sum(testing$test.subscribe ==0 & testing$y.subscribe ==1)  
p1.a0 = sum(testing$test.subscribe ==1 & testing$y.subscribe ==0)  
p1.a1 = sum(testing$test.subscribe ==1 & testing$y.subscribe ==1)  
###  
sensitivity = p1.a1 / (p1.a1 + p0.a1)  
specificity = p0.a0 / (p0.a0 + p1.a0)  
###  
precision = p1.a1 / (p1.a1 + p1.a0)  
recall = sensitivity  
F1 = 2\*precision\*recall/(precision + recall)  
metric.list = cbind(sensitivity = sensitivity,   
 specificity = specificity,   
 precision = precision,  
 recall = recall,  
 F1 = F1)  
kable(as.data.frame(metric.list), align='c', caption = "Local performance metrics")

Local performance metrics

| sensitivity | specificity | precision | recall | F1 |
| --- | --- | --- | --- | --- |
| 0.1705475 | 0.9792049 | 0.5211538 | 0.1705475 | 0.2569938 |

The sensitivity indicates the probability of those clients who are said to have subscribed a term deposit at the banking institution out of those who actually did is about 8-12%. The specificity indicates the probability of those clients who are said to have not subscribed a term deposit at the banking institution out of those who actually did not is about 97.3%.

### ROC Global Measure Analysis

For the last part of this section, a ROC (receiver operating characteristic) curve will be plotted by selecting a sequence of decision thresholds and calculating corresponding sensitivity and specificity.

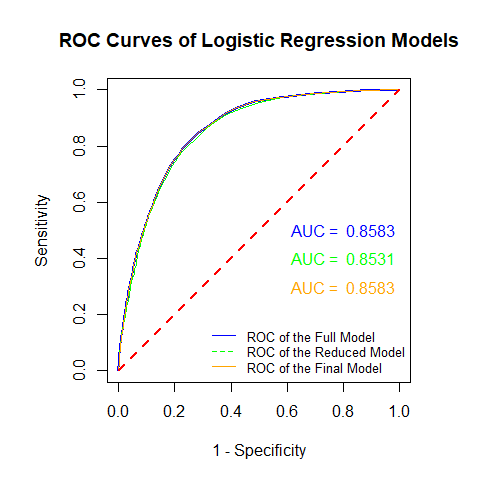
# Creating ROC curves for sensitivity and (1-specificity) for all 3 models  
## Full Model  
cut.off.seq = seq(0, 1, length = 100)  
sensitivity.vec.full = NULL  
specificity.vec.full = NULL  
###   
training.model.full = glm(y.subscribe ~ age + day + grp.job + marital + education + housing + loan + contact + grp.month + grp.duration + grp.campaign + grp.pdays + grp.previous, family = binomial(link = logit), data = training)  
newBankTrainingData.full = data.frame(age= training$age, day= training$day, grp.job= training$grp.job, marital= training$marital, education= training$education, housing= training$housing, loan= training$loan, contact= training$contact, grp.month= training$grp.month, grp.duration= training$grp.duration, grp.campaign= training$grp.campaign, grp.pdays= training$grp.pdays, grp.previous= training$grp.previous)  
pred.prob.train.full = predict.glm(training.model.full, newBankTrainingData.full, type = "response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

for (i in 1:100){  
 training$train.subscribe = as.numeric(pred.prob.train.full > cut.off.seq[i])  
### components for defining various measures  
TN = sum(training$train.subscribe == 0 & training$y.subscribe == 0)  
FN = sum(training$train.subscribe == 0 & training$y.subscribe == 1)  
FP = sum(training$train.subscribe == 1 & training$y.subscribe == 0)  
TP = sum(training$train.subscribe == 1 & training$y.subscribe == 1)  
###  
sensitivity.vec.full[i] = TP / (TP + FN)  
specificity.vec.full[i] = TN / (TN + FP)  
}  
one.minus.spec.full = 1 - specificity.vec.full  
sens.vec.full = sensitivity.vec.full  
## A better approx of ROC, need library {pROC}  
 prediction.full = pred.prob.train.full  
 category.full = training$y.subscribe == 1  
 ROCobj.full <- roc(category.full, prediction.full)  
 AUCfull = round(auc(ROCobj.full),4)  
   
## Reduced Model  
cut.off.seq = seq(0, 1, length = 100)  
sensitivity.vec.reduced = NULL  
specificity.vec.reduced = NULL  
###   
training.model.reduced = glm(y.subscribe ~ day + grp.job + marital + housing + loan + contact + grp.duration + grp.campaign + grp.previous, family = binomial(link = logit), data = training)  
newBankTrainingData.reduced = data.frame(day= training$day, grp.job= training$grp.job, marital= training$marital, housing= training$housing, loan= training$loan, contact= training$contact, grp.duration= training$grp.duration, grp.campaign= training$grp.campaign, grp.previous= training$grp.previous)  
pred.prob.train.reduced = predict.glm(training.model.reduced, newBankTrainingData.reduced, type = "response")  
for (i in 1:100){  
 training$train.subscribe = as.numeric(pred.prob.train.reduced > cut.off.seq[i])  
### components for defining various measures  
TN.reduced = sum(training$train.subscribe == 0 & training$y.subscribe == 0)  
FN.reduced = sum(training$train.subscribe == 0 & training$y.subscribe == 1)  
FP.reduced = sum(training$train.subscribe == 1 & training$y.subscribe == 0)  
TP.reduced = sum(training$train.subscribe == 1 & training$y.subscribe == 1)  
###  
sensitivity.vec.reduced[i] = TP.reduced / (TP.reduced + FN.reduced)  
specificity.vec.reduced[i] = TN.reduced / (TN.reduced + FP.reduced)  
}  
one.minus.spec.reduced = 1 - specificity.vec.reduced  
sens.vec.reduced = sensitivity.vec.reduced  
## A better approx of ROC, need library {pROC}  
 prediction.reduced = pred.prob.train.reduced  
 category.reduced = training$y.subscribe == 1  
 ROCobj.reduced <- roc(category.reduced, prediction.reduced)  
 AUCreduced = round(auc(ROCobj.reduced),4)  
   
# Final Model  
cut.off.seq = seq(0, 1, length = 100)  
sensitivity.vec.final = NULL  
specificity.vec.final = NULL  
###   
training.model.final = glm(y.subscribe ~ age + day + grp.job + marital + education + housing + loan + contact + grp.month + grp.duration + grp.campaign + grp.pdays + grp.previous, family = binomial(link = logit), data = training)  
newBankTrainingData.final = data.frame(age= training$age, day= training$day, grp.job= training$grp.job, marital= training$marital, education= training$education, housing= training$housing, loan= training$loan, contact= training$contact, grp.month= training$grp.month, grp.duration= training$grp.duration, grp.campaign= training$grp.campaign, grp.pdays= training$grp.pdays, grp.previous= training$grp.previous)  
pred.prob.train.final = predict.glm(training.model.final, newBankTrainingData.final, type = "response")

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
## prediction from a rank-deficient fit may be misleading

for (i in 1:100){  
 training$train.subscribe = as.numeric(pred.prob.train.final > cut.off.seq[i])  
### components for defining various measures  
TN.final = sum(training$train.subscribe == 0 & training$y.subscribe == 0)  
FN.final = sum(training$train.subscribe == 0 & training$y.subscribe == 1)  
FP.final = sum(training$train.subscribe == 1 & training$y.subscribe == 0)  
TP.final = sum(training$train.subscribe == 1 & training$y.subscribe == 1)  
###  
sensitivity.vec.final[i] = TP.final / (TP.final + FN.final)  
specificity.vec.final[i] = TN.final / (TN.final + FP.final)  
}  
one.minus.spec.final = 1 - specificity.vec.final  
sens.vec.final = sensitivity.vec.final  
## A better approx of ROC, need library {pROC}  
 prediction.final = pred.prob.train.final  
 category.final = training$y.subscribe == 1  
 ROCobj.final <- roc(category.final, prediction.final)  
 AUCfinal = round(auc(ROCobj.final),4)  
   
## Visual Representation of all 3 ROCs  
par(pty = "s") # make a square figure  
plot(one.minus.spec.full, sens.vec.full, type = "l", xlim = c(0,1), xlab ="1 - Specificity", ylab = "Sensitivity", main = "ROC Curves of Logistic Regression Models", lwd = 2, col = "blue")  
lines(one.minus.spec.reduced, sens.vec.reduced, col = "green")  
lines(one.minus.spec.final, sens.vec.final, col = "orange")  
segments(0,0,1,1, col = "red", lty = 2, lwd = 2)  
  
#AUCfull = round(sum(sens.vec.full\*(one.minus.spec.full[-101]-one.minus.spec.full[-1])),4)  
#AUCreduced = round(sum(sens.vec.reduced\*(one.minus.spec.reduced[-101]-one.minus.spec.reduced[-1])),4)  
#AUCfinal = round(sum(sens.vec.final\*(one.minus.spec.final[-101]-one.minus.spec.final[-1])),4)  
text(0.8, 0.5, paste("AUC = ", AUCfull), col = "blue")  
text(0.8, 0.4, paste("AUC = ", AUCreduced), col = "green")  
text(0.8, 0.3, paste("AUC = ", AUCfinal), col = "orange")  
legend("bottomright", c("ROC of the Full Model", "ROC of the Reduced Model", "ROC of the Final Model"), lty=c(1,2), col = c("blue", "green", "orange"), bty = "n", cex = 0.8)



The area under the curve (AUC) for the reduced model and the ROC curve is less than the other two graphs. Higher AUC indicates the model for that curve is better. Therefore, the reduced model is not the best model to use and is no longer considered. But, it should be noted that it is not far off the other 2 models.

Looking at the initial and final models, they have the same curve since both models contain all feature variables used in the initial model. Out of these other two models, the final model works better compared to the initial model. It is the parsimonious model. It has been proven to be accurate in modeling performance, has high specificity, and its ROC curve is remaining away from the 45 degrees mark. Plus, the AUC is fairly high at .8541, even though the initial model has the same score.

# Neural Network Modeling

Now we will create another model to predict using neural networks.

Training and testing data sets and canidate models will be made once again for the neural network model. From them, a final model will be constructed and plotted to show backpropagation of the neural network

Cross-validation will also be used be used for finding an optimal cut-off probability for assessing model performance and accuracy. Fianlly, an ROC curve will be made for the model to look at predictive power and compare it to the other model just like logistic regression.

## Changing varibles back to numeric form

Unlike in our logistic regression, the neuralnet library in R needs all feature variables to be in numeric form. Categorical variables become encoded into dummy variables. We want to be able to find all names for the feature variables and write them in the neural network model formula. The data set here will not any numerical labels for the categorical variables as with the previous model. The code is below:

# Assembling the discretized variables and other variables to make the modeling data set  
var.names = c("age", "day", "grp.job", "marital", "education", "housing", "loan", "contact", "grp.month", "grp.duration", "grp.campaign", "grp.pdays", "grp.previous", "y")   
BankMarketingNeural = BankMarketing[, var.names]

The numerical varibles must be rescaled and the code is below:

#re-scaling the numerical variables  
BankMarketingNeural$age = (BankMarketingNeural$age-min(BankMarketingNeural$age))/(max(BankMarketingNeural$age)-min(BankMarketingNeural$age))  
BankMarketingNeural$day = (BankMarketingNeural$day-min(BankMarketingNeural$day))/(max(BankMarketingNeural$day)-min(BankMarketingNeural$day))

Now we must create dummy variables for all categorical features:

# Creating a model.matrix() and looking at the names of the variable columns  
BankingMtx = model.matrix(~ ., data = BankMarketingNeural)  
colnames(BankingMtx)

## [1] "(Intercept)"   
## [2] "age"   
## [3] "day"   
## [4] "grp.jobnot working"   
## [5] "grp.jobwhite-collar"   
## [6] "grp.jobworkers"   
## [7] "marital married"   
## [8] "marital single"   
## [9] "education secondary"   
## [10] "education tertiary"   
## [11] "education unknown"   
## [12] "housing yes"   
## [13] "loan yes"   
## [14] "contact telephone"   
## [15] "contact unknown"   
## [16] "grp.monthspring"   
## [17] "grp.monthsummer"   
## [18] "grp.monthwinter"   
## [19] "grp.duration181-319"   
## [20] "grp.duration320+"   
## [21] "grp.campaign2-3"   
## [22] "grp.campaign4+"   
## [23] "grp.pdays200+"   
## [24] "grp.pdaysClient Not Previously Contacted"  
## [25] "grp.previous1-3"   
## [26] "grp.previous4+"   
## [27] "y yes"

#Renaming the categorical variable columns in the model matrix  
colnames(BankingMtx)[4] <- "grp.jobNotWorking"  
colnames(BankingMtx)[5] <- "grp.jobWhiteCollar"  
colnames(BankingMtx)[6] <- "grp.jobWorkers"  
colnames(BankingMtx)[7] <- "maritalMarried"  
colnames(BankingMtx)[8] <- "maritalSingle"  
colnames(BankingMtx)[9] <- "education2nd"  
colnames(BankingMtx)[10] <- "education3rd"  
colnames(BankingMtx)[11] <- "educationUnknown"  
colnames(BankingMtx)[12] <- "housingYes"  
colnames(BankingMtx)[13] <- "loanYes"  
colnames(BankingMtx)[14] <- "contactTelephone"  
colnames(BankingMtx)[15] <- "contactUnknown"  
colnames(BankingMtx)[16] <- "grp.monthSpring"  
colnames(BankingMtx)[17] <- "grp.monthSummer"  
colnames(BankingMtx)[18] <- "grp.monthWinter"  
colnames(BankingMtx)[19] <- "grp.duration0To180"  
colnames(BankingMtx)[20] <- "grp.duration320Plus"  
colnames(BankingMtx)[21] <- "grp.campaign1"  
colnames(BankingMtx)[22] <- "grp.campaign4Plus"  
colnames(BankingMtx)[23] <- "grp.pdays200Plus"  
colnames(BankingMtx)[24] <- "grp.pdaysCNPC"  
colnames(BankingMtx)[25] <- "grp.previous0"  
colnames(BankingMtx)[26] <- "grp.previous4Plus"  
colnames(BankingMtx)[27] <- "yYes"

## Create Candidate Model

Here the neural network model is defined using the changed names of the variables in the matrix.

# Defining the neural network model  
columnNames = colnames(BankingMtx)  
columnList = paste(columnNames[-c(1,length(columnNames))], collapse = "+")  
columnList = paste(c(columnNames[length(columnNames)],"~",columnList), collapse="")  
modelNeuralFormula = formula(columnList)  
modelNeuralFormula

## yYes ~ age + day + grp.jobNotWorking + grp.jobWhiteCollar + grp.jobWorkers +   
## maritalMarried + maritalSingle + education2nd + education3rd +   
## educationUnknown + housingYes + loanYes + contactTelephone +   
## contactUnknown + grp.monthSpring + grp.monthSummer + grp.monthWinter +   
## grp.duration0To180 + grp.duration320Plus + grp.campaign1 +   
## grp.campaign4Plus + grp.pdays200Plus + grp.pdaysCNPC + grp.previous0 +   
## grp.previous4Plus

Now we will split the data into two datasets. 70% is for training the neural network model and 30% is for testing.

# Creating the training and testing data sets for creating the model  
n = dim(BankingMtx)[1]  
testID = sample(1:n, round(n\*0.7), replace = FALSE)  
testData = BankingMtx[testID,]  
trainData = BankingMtx[-testID,]

Now that we have a training and testing dataset, a single-layer neural network model is created below.

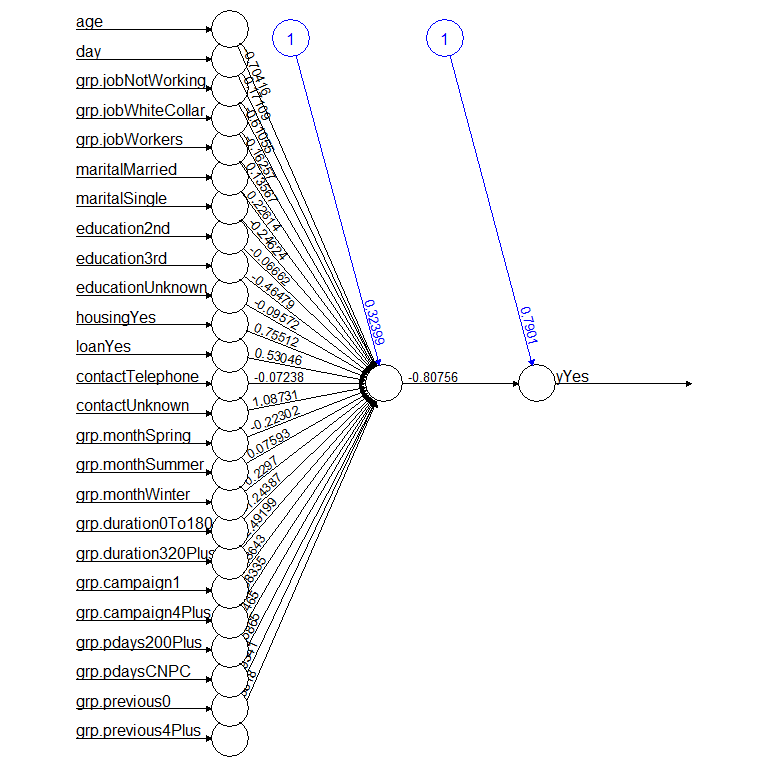
# Creating the single-layer neural network and model  
NetworkModel = neuralnet(modelNeuralFormula,  
 data = trainData,  
 hidden = 1, # single layer neural network  
 rep = 1, # number of replicates in training neural network  
 threshold = 0.01, # threshold for partial derivatives as stopping criteria.  
 learningrate = 0.1, # user selected rate  
 algorithm = "rprop+"  
 )  
kable(NetworkModel$result.matrix)

|  |  |
| --- | --- |
| error | 548.0981858 |
| reached.threshold | 0.0098849 |
| steps | 4902.0000000 |
| Intercept.to.1layhid1 | 0.3239860 |
| age.to.1layhid1 | -0.7041552 |
| day.to.1layhid1 | 0.1710929 |
| grp.jobNotWorking.to.1layhid1 | -0.6105496 |
| grp.jobWhiteCollar.to.1layhid1 | -0.1625685 |
| grp.jobWorkers.to.1layhid1 | 0.1356713 |
| maritalMarried.to.1layhid1 | 0.2261372 |
| maritalSingle.to.1layhid1 | -0.2462395 |
| education2nd.to.1layhid1 | -0.0666175 |
| education3rd.to.1layhid1 | -0.4647886 |
| educationUnknown.to.1layhid1 | -0.0957183 |
| housingYes.to.1layhid1 | 0.7551178 |
| loanYes.to.1layhid1 | 0.5304649 |
| contactTelephone.to.1layhid1 | -0.0723821 |
| contactUnknown.to.1layhid1 | 1.0873060 |
| grp.monthSpring.to.1layhid1 | -0.2230193 |
| grp.monthSummer.to.1layhid1 | 0.0759259 |
| grp.monthWinter.to.1layhid1 | 0.2296993 |
| grp.duration0To180.to.1layhid1 | -1.2438683 |
| grp.duration320Plus.to.1layhid1 | -2.4919857 |
| grp.campaign1.to.1layhid1 | 0.3642997 |
| grp.campaign4Plus.to.1layhid1 | 0.3833546 |
| grp.pdays200Plus.to.1layhid1 | 0.9464996 |
| grp.pdaysCNPC.to.1layhid1 | 2.2586451 |
| grp.previous0.to.1layhid1 | 1.1934083 |
| grp.previous4Plus.to.1layhid1 | 0.8581818 |
| Intercept.to.yYes | 0.7900995 |
| 1layhid1.to.yYes | -0.8075564 |

### Perceptron

In order to get a sense of what is going on, a map of the single-layer neural network is shown below:

# Plot for neural network model  
plot(NetworkModel, rep="best")



Single-layer backpropagation Neural network model for Banking Institution Client Term Deposit Subscription

logiModel = glm(factor(y) ~., family = binomial, data = BankMarketingNeural)  
pander(summary(logiModel)$coefficients)

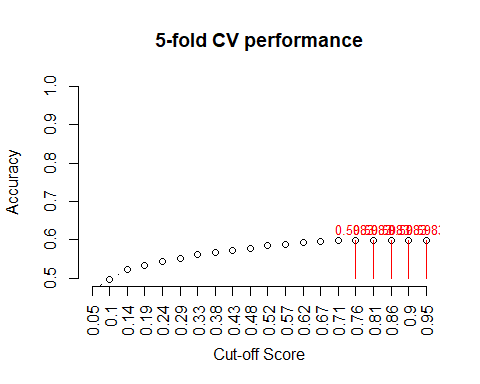
|  | Estimate | Std. Error | z value | Pr(>|z|) |
| --- | --- | --- | --- | --- |
| **(Intercept)** | -1.588 | 0.1378 | -11.52 | 1.009e-30 |
| **age** | 0.2957 | 0.1367 | 2.163 | 0.03051 |
| **day** | -0.2017 | 0.06161 | -3.274 | 0.001062 |
| **grp.jobnot working** | 0.451 | 0.06017 | 7.496 | 6.574e-14 |
| **grp.jobwhite-collar** | 0.08888 | 0.04938 | 1.8 | 0.07187 |
| **grp.jobworkers** | -0.09262 | 0.06162 | -1.503 | 0.1328 |
| **marital married** | -0.1794 | 0.05379 | -3.336 | 0.0008495 |
| **marital single** | 0.1635 | 0.06121 | 2.672 | 0.007548 |
| **education secondary** | 0.1383 | 0.05898 | 2.344 | 0.01906 |
| **education tertiary** | 0.4419 | 0.06776 | 6.522 | 6.923e-11 |
| **education unknown** | 0.2613 | 0.094 | 2.78 | 0.005434 |
| **housing yes** | -0.8169 | 0.03936 | -20.76 | 1.034e-95 |
| **loan yes** | -0.5359 | 0.05421 | -9.886 | 4.792e-23 |
| **contact telephone** | -0.0627 | 0.0684 | -0.9167 | 0.3593 |
| **contact unknown** | -1.054 | 0.0537 | -19.63 | 7.983e-86 |
| **grp.monthspring** | 0.07709 | 0.05479 | 1.407 | 0.1594 |
| **grp.monthsummer** | -0.1858 | 0.05287 | -3.513 | 0.0004429 |
| **grp.monthwinter** | -0.2336 | 0.06407 | -3.646 | 0.0002669 |
| **grp.duration181-319** | 1.328 | 0.05036 | 26.37 | 2.931e-153 |
| **grp.duration320+** | 2.699 | 0.04588 | 58.82 | 0 |
| **grp.campaign2-3** | -0.3073 | 0.03671 | -8.37 | 5.749e-17 |
| **grp.campaign4+** | -0.4754 | 0.05162 | -9.21 | 3.275e-20 |
| **grp.pdays200+** | -0.8797 | 0.06761 | -13.01 | 1.043e-38 |
| **grp.pdaysClient Not Previously Contacted** | -1.423 | 0.0767 | -18.56 | 7.193e-77 |
| **grp.previous1-3** | -0.2155 | 0.07785 | -2.768 | 0.005647 |

## Model Performance Testing

## Cut-off Probability Search and Accuracy Score

Cross-validation can be used in order to find the optimal cut-off scores because sigmoid perceptron is used, Hyper-parameteres will also be used to assess the performance of the neural network. The code is shown below:

# Looking at optimal cut-off probabilities with cross-validation  
n0 = dim(trainData)[1]/5  
cut.off.score = seq(0,1, length = 22)[-c(1,22)] # candidate cut off prob  
pred.accuracy = matrix(0,ncol=20, nrow=5, byrow = T) # null vector for storing prediction accuracy  
  
## 5-fold CV  
for (i in 1:5){  
 valid.id = ((i-1)\*n0 + 1):(i\*n0)  
 valid.data = trainData[valid.id,]  
 train.data = trainData[-valid.id,]  
 ####  
 train.model = neuralnet(modelNeuralFormula,  
 data = train.data,  
 hidden = 1, # single layer NN  
 rep = 1, # number of replicates in training NN  
 threshold = 0.01, # threshold for partial derivatives as stopping criteria.  
 learningrate = 0.1, # user selected rate  
 algorithm = "rprop+"  
 )  
 pred.nn.score = predict(NetworkModel, valid.data)  
 for(j in 1:20){  
 #pred.status = rep(0,length(pred.nn.score))  
 pred.status = as.numeric(pred.nn.score > cut.off.score[j])  
 a11 = sum(pred.status == valid.data[,17])  
 pred.accuracy[i,j] = a11/length(pred.nn.score)  
 }  
}  
###   
avg.accuracy = apply(pred.accuracy, 2, mean)  
max.id = which(avg.accuracy ==max(avg.accuracy ))  
  
### visual representation  
tick.label = as.character(round(cut.off.score,2))  
plot(1:20, avg.accuracy, type = "b",  
 xlim=c(1,20),   
 ylim=c(0.5,1),   
 axes = FALSE,  
 xlab = "Cut-off Score",  
 ylab = "Accuracy",  
 main = "5-fold CV performance"  
 )  
axis(1, at=1:20, label = tick.label, las = 2)  
axis(2)  
segments(max.id, 0.5, max.id, avg.accuracy[max.id], col = "red")  
text(max.id, avg.accuracy[max.id]+0.03, as.character(round(avg.accuracy[max.id],4)), col = "red", cex = 0.8)

 The above figure indicates that the optimal cut-off probability that yields the best accuracy is around 0.86.

Zooming in on the graphic above, we can see that the optimal cut-off is around .86. ## Results and Conclusion Now that we have chosed the best model out of the three candidate models, we can use it to predict whether or not a client has subscribed a term deposit. Using out optimal cut-off of .57

# Predicting Response Value for Banking Client Given Variable Values for the Final Model  
pdata = data.frame(age=c(25,64),  
 day = c(5,5),  
 grp.job = c("1","2"),  
 marital = c("0","1"),  
 education = c("2","0"),  
 housing = c("1","1"),  
 loan = c("0","1"),  
 contact = c("0","0"),  
 grp.duration = c("0","1"),  
 grp.campaign = c("1","0"),  
 grp.pdays = c("1","2"),  
 poutcome=c("0","0"),  
 grp.previous = c("1","2"))  
   
  
  
  
pred.success.prob = predict(final.model, newdata = pdata, type="response")  
  
## threshold probability  
cut.off.prob = 0.57  
pred.response = ifelse(pred.success.prob > cut.off.prob, 1, 0) # This predicts the response  
pred.response

## 1 2   
## 0 0

# Add the new predicted response to pdata  
  
#pdata$pred.response <- predict(final.model, newdata = pdata, type = "response")  
pdata$Pred.Response = pred.response  
kable(pdata, caption = "Predicted Value of response variable with the given cut-off probability")

Predicted Value of response variable with the given cut-off probability

| age | day | grp.job | marital | education | housing | loan | contact | grp.duration | grp.campaign | grp.pdays | poutcome | grp.previous | Pred.Response |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 25 | 5 | 1 | 0 | 2 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 64 | 5 | 2 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 2 | 0 | 2 | 0 |

You can see that neither of the two observations will be subscribing. More testing can be done with the testing dataset.

## Testing Model Accuracy and Performance

This optimal cut-off probability and testing data will now be used to look at the accuracy and confusion matrix of the model.

#Testing resulting output and finding model accuracy and confusion matrix  
nn.results <- predict(NetworkModel, testData)  
results <- data.frame(actual = testData[,27], prediction = nn.results > .86)  
confMatrix = table(results$prediction, results$actual) # confusion matrix  
accuracy=sum(results$actual == results$prediction)/length(results$prediction)  
list(confusion.matrix = confMatrix, accuracy = accuracy)

## $confusion.matrix  
##   
## 0 1  
## FALSE 27934 3714  
##   
## $accuracy  
## [1] 0.8826466

The accuracy score we see is 88.1%. This indicates that there is no under-fitting. THis result is similar to what we saw in the logistic regression model.

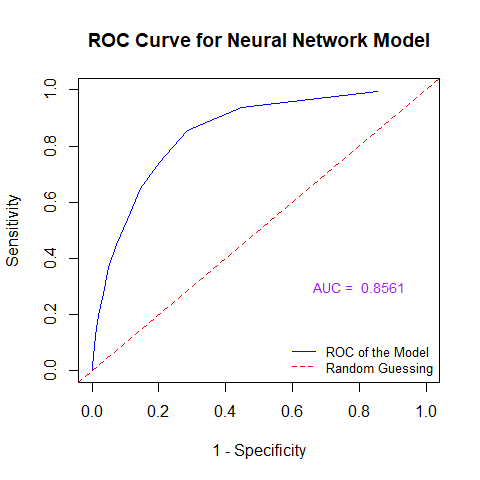
## ROC Analysis

Now a ROC will be constructed for the neural net model based on the training data set we split above.

nn.results = predict(NetworkModel, trainData) # Keep in mind that trainDat is a matrix!  
cut0 = seq(0,1, length = 20)  
SenSpe = matrix(0, ncol = length(cut0), nrow = 2, byrow = FALSE)  
for (i in 1:length(cut0)){  
 a = sum(trainData[,"yYes"] == 1 & (nn.results > cut0[i]))  
 d = sum(trainData[,"yYes"] == 0 & (nn.results < cut0[i]))  
 b = sum(trainData[,"yYes"] == 0 & (nn.results > cut0[i]))   
 c = sum(trainData[,"yYes"] == 1 & (nn.results < cut0[i]))   
 sen = a/(a + c)  
 spe = d/(b + d)  
 SenSpe[,i] = c(sen, spe)  
}  
  
# plotting ROC  
plot(1-SenSpe[2,], SenSpe[1,], type ="l", xlim=c(0,1), ylim=c(0,1),  
 xlab = "1 - Specificity", ylab = "Sensitivity", lty = 1,  
 main = "ROC Curve for Neural Network Model", col = "blue")  
abline(0,1, lty = 2, col = "red")  
  
## Calculate AUC  
xx = 1-SenSpe[2,]  
yy = SenSpe[1,]  
width = xx[-length(xx)] - xx[-1]  
height = yy[-1]  
  
## A better approx of ROC, need library {pROC}  
 prediction = nn.results  
 category = trainData[,"yYes"] == 1  
 ROCobj <- roc(category, prediction)

## Warning in roc.default(category, prediction): Deprecated use a matrix as  
## predictor. Unexpected results may be produced, please pass a numeric vector.

AUC.neural = auc(ROCobj)[1]  
 ##  
###  
  
#AUC =mean(sum(width\*height), sum(width\*yy[-length(yy)]))  
text(0.8, 0.3, paste("AUC = ", round(AUC.neural,4)), col = "purple", cex = 0.9)  
legend("bottomright", c("ROC of the Model", "Random Guessing"), lty=c(1,2),  
 col = c("blue", "red"), bty = "n", cex = 0.8)



The above ROC curve show that the neural network model is than guessing since the area under the curve (AUC) is significantly greater than 0.5. Since the AUC = 0.8538 and is also significantly greater than 0.65, this means the predictive power of the model is adequate to use.

# Decsion Trees

Now we will build a third predictive model using decision trees. At least 4 to 6 different decision trees will be constructed based on impurity measures (Gini index and information entropy), penalty coefficients, and costs of false negatives and positives.

An ROC curve will be used in order to find the best decision tree model out of the candidate models. An optimal cut-off score will be found through cross-validation for reporting predictive performance of the final model.

Tree Algorithims are is based on conditional probabilities statements, in order to identify certain sets of records to be used to make a prediction of the response. Predictive performance of a decision tree is dependent on the trained tree size. The Gini index and information entropy are two impurity measures used to control size of a decision for obtaining best performance. They also choose feature variables for defining root and decision nodes, as well as how to split the variables. Gini Index considers a split for each attribute and measures the impurity of subgroups split by a feature variable.

The original data set, with the discretized variables, will be used. The data will be split into a 70% data set for training the model and a 30% data set for testing the model.

# Read in the data set again without the NN changes  
var.names = c("age", "day", "grp.job", "marital", "education", "housing", "loan", "contact", "grp.month", "grp.duration", "grp.campaign", "grp.pdays", "grp.previous", "y")   
BankMarketingTrees = BankMarketing[, var.names]  
  
# Random split approach for new training and testing data sets  
n = dim(BankMarketingTrees)[1] # sample size  
# caution: using without replacement  
id.train = sample(1:n, round(0.7\*n), replace = FALSE)   
train = BankMarketingTrees[id.train, ] # training data  
test = BankMarketingTrees[-id.train, ] # testing data

## Building Trees

Using a wrapper we can pass in the arguments of impurity and penalty measures, as well as costs of false positives and negatives, to construct different decision trees.

we aim to create 6 trees:

Model 1: gini.tree.11 is based on the Gini index without penalizing false positives and false negatives.

Model 2: info.tree.11 is based on entropy without penalizing false positives and false negatives.

Model 3: gini.tree.110 is based on the Gini index: cost of false negatives is 10 times the positives.

Model 4: info.tree.110 is based on entropy: cost of false negatives is 10 times the positives.

Model 5: gini.tree.101 is based on the Gini index: cost of false positive is 10 times the negatives.

Model 6: info.tree.101 is based on entropy: cost of false positive is 10 times the negatives.

The code for building the different candidate trees is below:

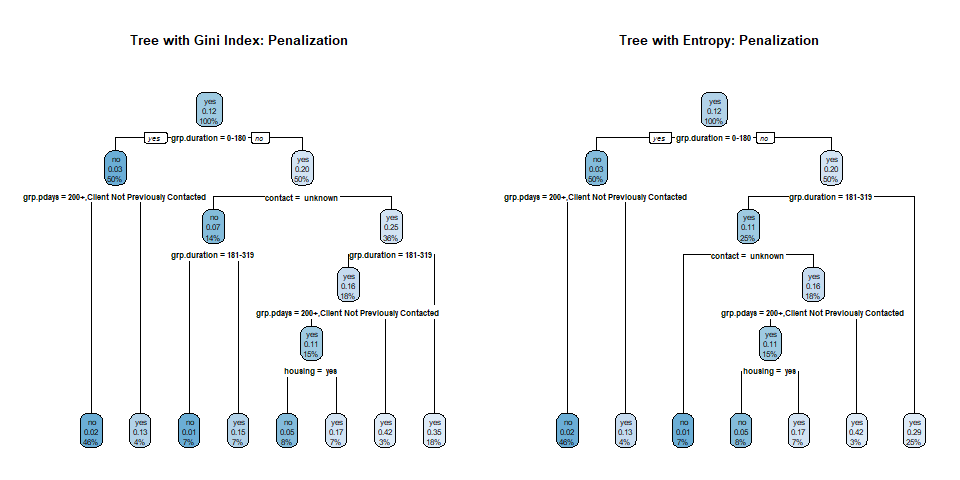
# Defining different tree models using the rpart() function  
tree.builder = function(in.data, fp, fn, purity){  
 tree = rpart(y ~ ., # including all feature variables  
 data = in.data,   
 na.action = na.rpart, # By default, deleted if the outcome is missing,   
 method = "class", # Classification form factor  
 model = TRUE,  
 x = FALSE,  
 y = TRUE,  
 parms = list( # Penalizes false positives or negatives more heavily  
 loss = matrix(c(0, fp, fn, 0), ncol = 2, byrow = TRUE),   
 split = purity), # Gini index or information entropy  
 ## rpart algorithm options start here  
 control = rpart.control(  
 minsplit = 10, # minimum number of observations required before split  
 minbucket= 10, # minimum number of observations in any terminal node  
 cp = 0.01, # complexity parameter for stopping rule  
 maxcompete = 5, # number of competitor splits retained in the output  
 maxsurrogate = 6, # number of surrogate splits retained in the output  
 maxdepth = 5,  
 xval = 10 # number of cross-validation )  
 )  
 )  
 }

Now the models can be fit below:

## Statements to recall the tree model wrapper  
gini.tree.1.1 = tree.builder(in.data = train, fp = 1, fn = 1, purity = "gini")  
info.tree.1.1 = tree.builder(in.data = train, fp = 1, fn = 1, purity = "information")  
gini.tree.1.10 = tree.builder(in.data = train, fp = 1, fn = 10, purity = "gini")  
info.tree.1.10 = tree.builder(in.data = train, fp = 1, fn = 10, purity = "information")  
gini.tree.10.1 = tree.builder(in.data = train, fp = 10, fn = 1, purity = "gini")  
info.tree.10.1 = tree.builder(in.data = train, fp = 10, fn = 1, purity = "information")

The two plots below show the two penalized decision models where cost of false negative is 10 times the positives.

## Plotting the tree plots  
par(mfrow=c(1,2))  
rpart.plot(gini.tree.1.10, main = "Tree with Gini Index: Penalization")  
rpart.plot(info.tree.1.10, main = "Tree with Entropy: Penalization")



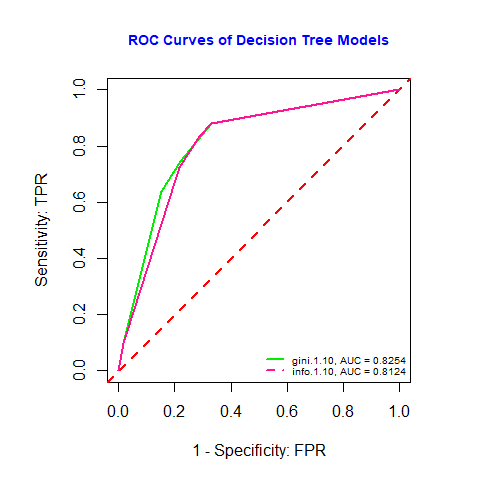
Penalized decision tree models when cost of false negatives is 10 times the positives using Gini index (left) and entropy (right).

The last two penalized decision models where cost of false positive is 10 times the negatives will not run properly for the analysis or produce any full models. This could be due to the built-in stopping rule. It may indicate, based on penalizing weights, no significant information gain for any plotting beyond the root node. Therefore, it would be best to focus on just the two models of false negatives for our predictive analysis analysis.

## ROC Anaylsis

Looking at our two canidate models, we can look at the ROCs of both these curves, as well as their AUCs in order to selet the optimal model. A new function will be used to build 2 different trees and plot their corresponding ROC curves, and calculate the AUCs for each, so we can see the global performance of these tree algorithms. The code can be seen below:

## Creating a function for returning a sensitivity and specificity matrix  
SensSpec = function(in.data, fp, fn, purity){  
 cutoff = seq(0,1, length = 20) # 20 cut-offs including 0 and 1.   
 model = tree.builder(in.data, fp, fn, purity)   
   
 ## Decision trees return both "success" and "failure" probabilities, but we need only "success" probability to define sensitivity and specificity   
 pred = predict(model, newdata = in.data, type = "prob") # two-column matrix.  
 senspe.mtx = matrix(0, ncol = length(cutoff), nrow= 2, byrow = FALSE)  
 for (i in 1:length(cutoff)){  
   
 # The following line uses only " yes" probability: pred[, " yes"]  
 pred.out = ifelse(pred[," yes"] >= cutoff[i], " yes", " no")   
 TP = sum(pred.out ==" yes" & in.data$y == " yes")  
 TN = sum(pred.out ==" no" & in.data$y == " no")  
 FP = sum(pred.out ==" yes" & in.data$y == " no")  
 FN = sum(pred.out ==" no" & in.data$y == " yes")  
 senspe.mtx[1,i] = TP/(TP + FN)  
 senspe.mtx[2,i] = TN/(TN + FP)  
 }  
 ## A better approx of ROC, need library {pROC}  
 prediction = pred[, " yes"]  
 category = in.data$y == " yes"  
 ROCobj <- roc(category, prediction)  
 AUC.tree = auc(ROCobj)  
 ##  
 list(senspe.mtx= senspe.mtx, AUC.tree = round(AUC.tree,4))  
 }  
  
#Building the six tree models  
giniROC110 = SensSpec(in.data = train, fp=1, fn=10, purity="gini")  
infoROC110 = SensSpec(in.data = train, fp=1, fn=10, purity="information")  
  
# Creating the ROC curves for both models  
par(pty="s") # set up square plot through graphic parameter  
plot(1-giniROC110$senspe.mtx[2,], giniROC110$senspe.mtx[1,], type = "l", xlim=c(0,1), ylim=c(0,1),   
 xlab="1 - Specificity: FPR", ylab="Sensitivity: TPR", col = "green2", lwd = 2,  
 main="ROC Curves of Decision Tree Models", cex.main = 0.9, col.main = "blue")  
abline(0,1, lty = 2, col = "red", lwd = 2)  
lines(1-infoROC110$senspe.mtx[2,], infoROC110$senspe.mtx[1,], col = "deeppink", lwd = 2)  
  
legend("bottomright", c(paste("gini.1.10, AUC =",giniROC110$AUC.tree), paste("info.1.10, AUC =",infoROC110$AUC.tree)), col=c("green2","deeppink"), lty=c(1,2,rep(1,4)), lwd=rep(2,6), cex = 0.6, bty = "n")



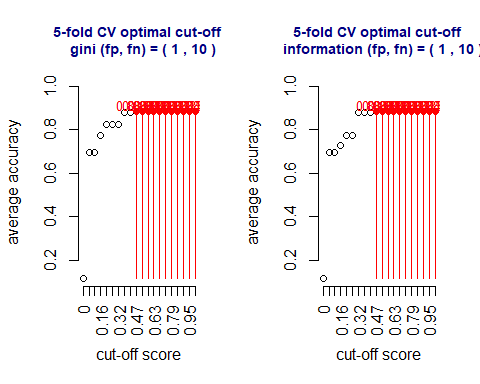
We can see above the gini model AUC. This indicates that the predictive model is both better than the random guess since the area under the curve (AUC) is significantly greater than 0.5. Since all these AUC are also significantly greater than 0.65, this means the predictive power of the model is acceptable.

The Gini index model will be the final model used for finding the optimal cut-off probability for predictive modeling since it has the largest AUC.

## Optimal Cut-Off Score Determination

With the final decision tree model established, we will find the optimal cut-off score for reporting predictive performance. This will be done using the test data and cross-validation based on the training data set. The code is below:

optm.cutoff = function(in.data, fp, fn, purity){  
 n0 = dim(in.data)[1]/5  
 cutoff = seq(0,1, length = 20) # candidate cut off prob  
 ## accuracy for each candidate cut-off  
 accuracy.mtx = matrix(0, ncol=20, nrow=5) # 20 candidate cutoffs and gini.11  
 ##  
 for (k in 1:5){  
 valid.id = ((k-1)\*n0 + 1):(k\*n0)  
 valid.dat = in.data[valid.id,]  
 train.dat = in.data[-valid.id,]   
 ## tree model  
 tree.model = tree.builder(in.data, fp, fn, purity)  
 ## prediction   
 pred = predict(tree.model, newdata = valid.dat, type = "prob")[,2]  
 ## for-loop  
 for (i in 1:20){  
 ## predicted probabilities  
 pc.1 = ifelse(pred > cutoff[i], " yes", " no")  
 ## accuracy  
 a1 = mean(pc.1 == valid.dat$y)  
 accuracy.mtx[k,i] = a1  
 }  
 }  
 avg.acc = apply(accuracy.mtx, 2, mean)  
 ## plots  
 n = length(avg.acc)  
 idx = which(avg.acc == max(avg.acc))  
 tick.label = as.character(round(cutoff,2))  
 ##  
 plot(1:n, avg.acc, xlab="cut-off score", ylab="average accuracy",   
 ylim=c(min(avg.acc), 1),   
 axes = FALSE,  
 main=paste("5-fold CV optimal cut-off \n ",purity,"(fp, fn) = (", fp, ",", fn,")" , collapse = ""),  
 cex.main = 0.9,  
 col.main = "navy")  
 axis(1, at=1:20, label = tick.label, las = 2)  
 axis(2)  
 points(idx, avg.acc[idx], pch=19, col = "red")  
 segments(idx , min(avg.acc), idx , avg.acc[idx ], col = "red")  
 text(idx, avg.acc[idx]+0.03, as.character(round(avg.acc[idx],4)), col = "red", cex = 0.8)   
 }  
  
par(mfrow=c(1,2))  
optm.cutoff(in.data = train, fp=1, fn=10, purity="gini")  
optm.cutoff(in.data = train, fp=1, fn=10, purity="information")



The above figure indicates that the optimal cut-off probabilities for the Gini index (left) and entropy (right) models are relatively the same in range. Therefore, the optimal average cut-off probability for the final Gini index model is 0.74.

# Bagging

Now we will build a final predictive model using bagging. We will split the data into testing and training once again.

# We use a random split approach  
n = dim(BankMarketingTrees)[1] # sample size  
# caution: using without replacement  
train.id = sample(1:n, round(0.7\*n), replace = FALSE)   
train = BankMarketingTrees[train.id, ] # training data  
test = BankMarketingTrees[-train.id, ] # testing data

The code for fitting the bagging model is below:

boot\_data <- function(df){  
 return(df[sample(1:nrow(df), replace = T),])  
}  
  
# (2) Function to run an Rpart model, provided  
# `x` data and a model formula `form`  
rpart\_fit <- function(x, form){  
 rpart(form, data = x, method = "class")  
}  
  
# (3) Function to get predictions from models on new data  
get\_pred <- function(x, newdat) {  
 p <- predict(x, newdata = newdat)  
   
 return(p[, 2])  
}  
  
# (4) Function for random forest parameter selection  
sample\_p <- function(df, y, maxp = 3) {  
 x <- df[!names(df) %in% y]  
 x <- df[sample(names(x), maxp)]  
   
 return(cbind(x, df[y]))  
}  
  
# BAGGING  
#-------------------#  
# define number of bootstrap iterations  
B = 100  
  
# set up list to hold models, length B  
blist <- vector(mode = "list", length = B)  
  
# bootstrap B models  
for(i in 1:B)  
 blist[[i]] <- boot\_data(train)  
  
# run vectorized model, adding our model formula in  
bag\_fit <- lapply(blist, rpart\_fit, form = y ~ .)  
  
# get predictions on new data  
b\_avg <- do.call(rbind, (lapply(bag\_fit, get\_pred, test)))  
  
# average over predictions  
b\_final <- apply(b\_avg, 2, mean)

## ROC and Test

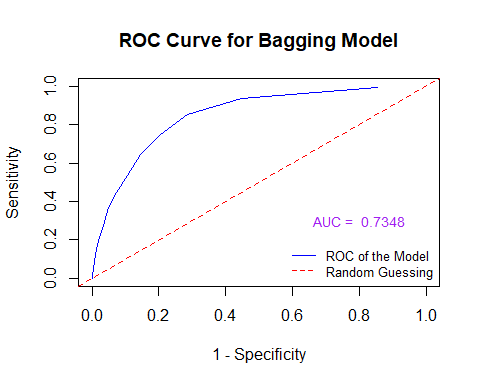
# Use pROC for AUC  
  
# bagging and random forest  
roc(test$y, b\_final)

##   
## Call:  
## roc.default(response = test$y, predictor = b\_final)  
##   
## Data: b\_final in 11960 controls (test$y no) < 1603 cases (test$y yes).  
## Area under the curve: 0.7348

# plotting ROC  
plot(1-SenSpe[2,], SenSpe[1,], type ="l", xlim=c(0,1), ylim=c(0,1),  
 xlab = "1 - Specificity", ylab = "Sensitivity", lty = 1,  
 main = "ROC Curve for Bagging Model", col = "blue")  
abline(0,1, lty = 2, col = "red")  
  
## Calculate AUC  
xx = 1-SenSpe[2,]  
yy = SenSpe[1,]  
width = xx[-length(xx)] - xx[-1]  
height = yy[-1]  
  
## A better approx of ROC, need library {pROC}  
 prediction = nn.results  
 category = trainData[,"yYes"] == 1  
 ROCobj <- roc(category, prediction)

## Warning in roc.default(category, prediction): Deprecated use a matrix as  
## predictor. Unexpected results may be produced, please pass a numeric vector.

AUC.b\_final = auc(test$y,b\_final)  
 ##  
###  
  
#AUC =mean(sum(width\*height), sum(width\*yy[-length(yy)]))  
text(0.8, 0.3, paste("AUC = ", round(AUC.b\_final,4)), col = "purple", cex = 0.9)  
legend("bottomright", c("ROC of the Model", "Random Guessing"), lty=c(1,2),  
 col = c("blue", "red"), bty = "n", cex = 0.8)

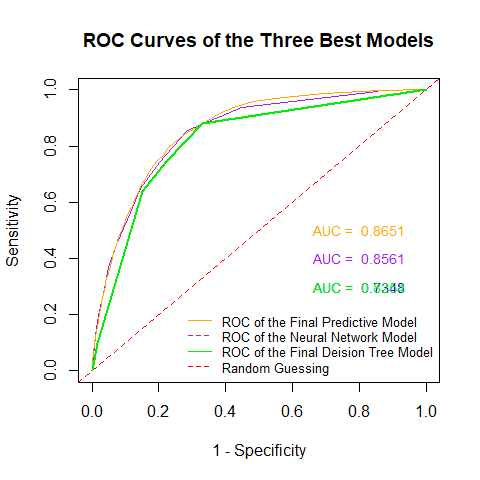
 We can see above bagging model AUC. This indicates that the predictive model is both better than the random guess since the area under the curve (AUC) is significantly greater than 0.5. Since all these AUC are also significantly greater than 0.65, this means the predictive power of the model is acceptable.

# Model Comparison

# Plotting both ROC curves  
plot(1-SenSpe[2,], SenSpe[1,], type ="l", xlim=c(0,1), ylim=c(0,1), xlab = "1 - Specificity", ylab = "Sensitivity", lty = 1, main = "ROC Curves of the Three Best Models", col = "purple")  
abline(0,1, lty = 2, col = "red")  
lines(one.minus.spec.final, sens.vec.final, col = "orange")  
lines(1-giniROC110$senspe.mtx[2,], giniROC110$senspe.mtx[1,], col = "green2", lwd = 2)  
lines(1-AUC.b\_final, AUC.b\_final ,col = "blue")  
  
  
## Calculate AUCs for both  
text(0.8, 0.4, paste("AUC = ", round(AUC.neural,4)), col = "purple", cex = 0.9)  
text(0.8, 0.3, paste("AUC = ", round(AUC.b\_final,4)), col = "blue", cex = 0.9)  
AUC.final = round(sum(sens.vec.final\*(one.minus.spec.final[-101]-one.minus.spec.final[-1])),4)

## Warning in one.minus.spec.final[-101] - one.minus.spec.final[-1]: longer object  
## length is not a multiple of shorter object length

text(0.8, 0.5, paste("AUC = ", round(AUC.final,4)), col = "orange", cex = 0.9)  
text(0.8, 0.3, paste("AUC = ", round(giniROC110$AUC,4)), col = "green2", cex = 0.9)  
  
  
# Adding a legend to the figure  
legend("bottomright", c("ROC of the Final Predictive Model", "ROC of the Neural Network Model", "ROC of the Final Deision Tree Model", "Random Guessing"), lty=c(1,2), col = c("orange", "purple", "green2", "red","black", ""), bty = "n", cex = 0.8)



# Summary and Discussion

Overall in this project we have done predictive modeling consisting of logistic regression, neural networks, decision tree analysis, and bagging in order to predict if a client will subscribe after a direct marketing campaign. After looking at all the final candidate models, the best one to used based of AUC would be the logistical model with AUC = .8541, followed by the neural network with an AUC = .8534. The logistical model is the final model and is the simplest model in the program. This proves that sometimes simple models can answer some questions better than complex models such as decision trees.