6372: Project 2

Simerpreet Reddy, Rinku Lichti, Megan Ball

# Introduction

The goal of this analysis was to develop an optimal classification model to predict the success rates of a Portuguese bank telemarketing campaign. As part of this, a total of four models were developed for use in comparison for best overall combined accuracy, sensitivity, and specificity:

* Simple, interpretable linear regression model
* Complex linear regression model
* Linear discriminant analysis (LDA) model
* Random Forest model

The subsequent discussion will review the analysis steps, model metrics, and ultimate best model for classification.

# Data Description

We used bank-additional-full.csv as our base dataset. This dataset consisted of 41,188 observations with a total of 21 different variables that consisted of various factors pertaining to the following:

* bank client data: age, job, marital status, education, default (credit in default?), housing and loan.
* previous campaign attributes related to the last time the customer was contacted:  contact(method), month, day\_of\_week and duration (of the call in seconds).
* Other campaign attributes: campaign (times contacted in the current campaign), pdays (days since last call), previous (times contacted in last campaign), Poutcome (outcome of the previous campaign).
* Social and economic context attributes:  emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m, nr.employed.

Out of these the following were categorical variables: job, marital status, education, default, housing, loan, contact, month, day\_of\_week, poutcome and the response variable ‘y’.

Following were the continuous variables: duration, pdays, previous, emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m and nr.employed.

Data is a mix of categorical and continuous variables and was collected with a classification goal to predict if the client will subscribe (yes/no) to a term deposit (variable y). After initial EDA, we did the following:

1. Derived variables: We derived the following variables using the provided variables:

**Age\_Group from age**: We used the variable ‘age’ to create variable ‘Age\_Grp’(age groups) with values "17-31","32-37" ,"38-47", "47-55", ">55". We did this based in IQR for age.

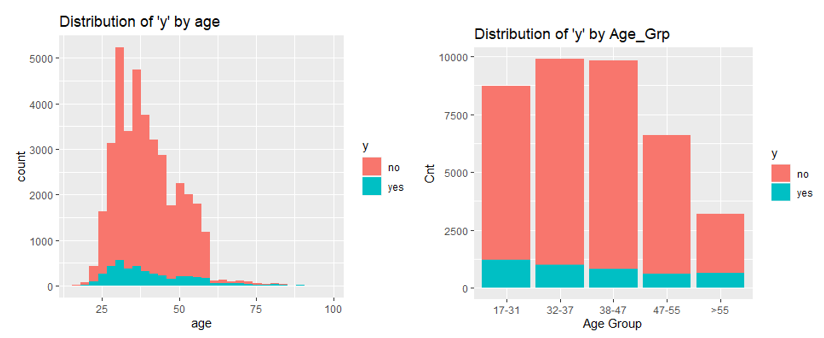


Figure 1

**prevly\_Cntctd from pdays:** We created variable prevly\_Cntctd with values yes/no to see if the client has been previously contacted ever. Pdays**=**999 meant that the client has never been contacted before.

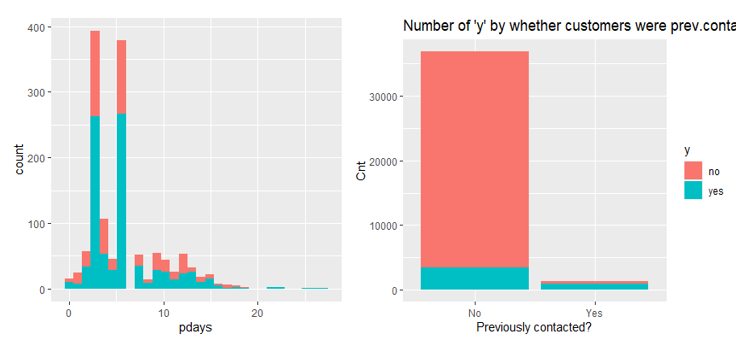


Figure 2

**duration\_group from duration:** We created variable duration group to determine what range did the last call duration fall in - "0-5min", "5-10min" or "10+ min".

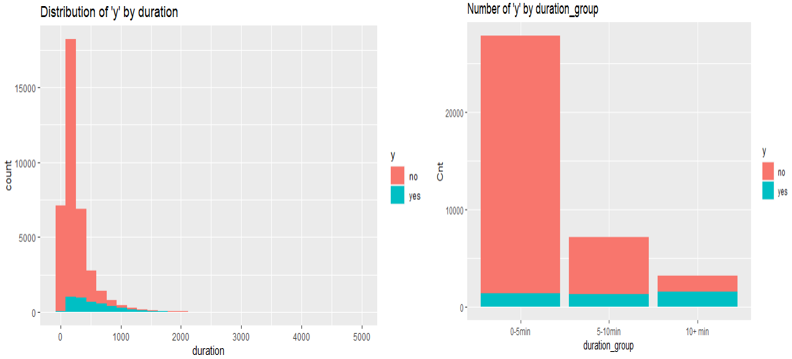


Figure 3

1. Balance/Unbalanced data set: We chose not to balance the data as it would lead to losing a lot of the information. Instead, we decided to plot ROC curves on the chosen models to predict the best cut off to be used for prediction.
2. Removing rows from the data set:
3. There were no null values in the data set but ‘unknowns’. We removed the rows where the following variable had ‘unknown’ values – marital(80), job(289), loan (990) and education (1732). Column default had ‘unknowns’ as well but we decided to keep those as they could be important predictors.
4. We removed 3 rows of data with default=’yes’ as such few values were adding error the data model.
5. Data Types: We changed the data type of following variables to factors: job", "marital", "education", "housing","loan","contact","month","day\_of\_week","default","poutcome","y". Adjust level of ‘y’ to (‘No’, ’Yes’) to make sure ‘yes’ for considered at event=1. We kept the continuous variables as numeric.
6. Train/test split: While splitting data set into train and test, we made sure these data sets received 80:20 ratio of both ‘Yes’ and ‘No’ of the response variable.

# Exploratory Data Analysis

Walkthrough of key findings in EDA

* Use graphs relevant to our interactions
* Use graphs relevant to simple model chosen vars

# 

Figure

# Objective 1

## Restatement of Problem

Our first goal was to develop a simple and interpretable model to predict and explain the relationship between our variables and the probability of customers opening an account. To do this, we chose logistic regression as our desired model type and performed variable selection and cross validation to select the best interpretable model.

Modeling Strategy

After doing EDA and creating a few derived variables, we started with creating a simple model using all the variables and then step by step removing statistically insignificant variables as well as variables with high multicollinearity based on both VIFs and scatter plots. After every column removal, we re-ran the model and repeated the exercise till we found the model with statistically significant as well moderate (generally with values less than 10) VIF variables.

We then ran both stepwise selection and LASSO as competing feature selection methods, and further limiting our variables based on VIF. Using ROC curves to find a good cut off range, we predicted the response variable with multiple cut off values in the range 0.1 and 0.5 to determine optimum value. We then compared the accuracy, sensitivity and specificity of all the different models – simple, step, and lasso – on various cut offs. Since the primary goal is to predict when a customer will subscribe to a term deposit (positive response variable = ‘yes’) we chose a cut-off value based on balancing our overall accuracy and the sensitivity.

Model

For the simple interpretable model, the following model gave us the best combined accuracy, sensitivity, and specificity at a cut off probability value of 0.15.

## Assumptions

* Data points are independent.

Lack of fit

* We performed a global test for the coefficients (beta) being equal to zero with Likelihood ratio, Score and Wald test all agreeing at p-value <0.001 that we reject the null hypothesis and conclude that the logistic regression model is valid.
* Cross validation was done in the form of a test/train split which demonstrated good metrics on our test data set.

Table

|  |  |
| --- | --- |
| Metric | Result |
| Accuracy | 86.6% |
| Sensitivity | 83.7% |
| Specificity | 87.2% |

Influential points analysis

There were several points that stood out when we looked at Leverage, Cook’s D, Standard residuals, and deviance plots. We removed those and reran the code, even though the plots looked better, there were no significant changes in the model or the variable coefficients. So, we decided to keep those data points in the training data set.

## Model Interpretation

Table . Odds ratios and 95% confidence intervals for logistic regression model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variable | Odds Ratio | 95% CI: Lower Limit | 95% CI: Upper Limit | P-value |
| Intercept | 1.486 x 10-21 | 5.619 x 10-26 | 3.934 x 10-17 | < 0.001 |
| Job: blue-collar | 0.707 | 0.610 | 0.820 | < 0.001 |
| Job: entrepreneur | 0.871 | 0.600 | 1.151 | 0.332 |
| Job: housemaid | 0.818 | 0.589 | 1.136 | 0.230 |
| Job: management | 0.950 | 0.784 | 1.150 | 0.600 |
| Job: retired | 1.085 | 0.843 | 1.397 | 0.528 |
| Job: self-employed | 0.893 | 0.690 | 1.155 | 0.388 |
| Job: services | 0.741 | 0.615 | 0.892 | 0.002 |
| Job: student | 1.192 | 0.917 | 1.550 | 0.190 |
| Job: technician | 0.994 | 0.863 | 1.146 | 0.938 |
| Job: unemployed | 0.884 | 0.655 | 1.195 | 0.423 |
| Default: unknown | 0.722 | 0.620 | 0.841 | < 0.001 |
| Contact: telephone | 0.691 | 0.588 | 0.811 | < 0.001 |
| Month: August | 1.264 | 1.000 | 1.600 | 0.050 |
| Month: December | 1.052 | 0.643 | 1.720 | 0.841 |
| Month: July | 1.396 | 1.123 | 1.736 | 0.003 |
| Month: June | 1.485 | 1.202 | 1.835 | < 0.001 |
| Month: March | 5.741 | 4.332 | 7.610 | < 0.001 |
| Month: May | 0.530 | 0.446 | 0.631 | < 0.001 |
| Month: November | 1.072 | 0.854 | 1.346 | 0.545 |
| Month: October | 1.514 | 1.135 | 2.021 | 0.005 |
| Month: September | 0.992 | 0.725 | 1.359 | 0.961 |
| Duration | 1.002 | 1.002 | 1.003 | < 0.001 |
| Campaign | 0.956 | 0.931 | 0.981 | < 0.001 |
| Previous number of contacts | 0.762 | 0.694 | 0.836 | < 0.001 |
| Cons\_price\_idx | 1.692 | 1.513 | 1.893 | < 0.001 |
| Cons\_conf\_idx | 1.060 | 1.047 | 1.072 | < 0.001 |
| Euribor3m | 0.491 | 0.470 | 0.513 | < 0.001 |
| Age Group (32-37) | 0.829 | 0.727 | 0.946 | 0.005 |
| Age Group (38-47) | 0.743 | 0.646 | 0.855 | < 0.001 |
| Age Group (47 – 55) | 0.859 | 0.734 | 1.004 | 0.057 |
| Age Group (55+) | 0.930 | 0.753 | 1.149 | 0.502 |
| Previously contacted: Yes | 6.190 | 5.048 | 7.590 | < 0.001 |
| Duration Group: 5-10min | 2.983 | 2.600 | 3.424 | < 0.001 |
| Duration Group: 10+ min | 7.200 | 5.563 | 9.319 | < 0.001 |

The variables with the largest odds ratios were categorical variables and included the month of March and each of the duration groups.

The continuous variables with the largest odds ratios included duration, consumer price index, and consumer confidence index. For each additional minute spent on the phone, it is estimated that the odds of a client opening an account increase by 0.002%, holding all other variables constant. For each one unit increase in consumer price index, it is estimated that the odds of a client opening an account increase by a multiplicative factor of 1.7, holding all other variables constant. A 95% confidence interval associated with this odds ratio is between 1.5 and 1.9.

# Objective 2

In an attempt to further improve the accuracy and sensitivity of our classification model, we also built a more complex logistic regression model including interaction, made classifications using linear discriminant analysis, and also developed a random forest model.

#### Complex Model

A complex logistic regression model was built off of the variables deemed most significant from our simple regression model. Based on EDA, as mentioned above, the factors that appeared to have some sort of interaction included default with duration, contact with duration, default with month, and month with euribor3m.

#### Linear Discriminant Analysis

In addition to a complex logistic model, a both linear and quadratic discriminant analysis were run on the numeric variables for comparison.

#### Random Forest

To tune Random Forest, I chose mtry and the split function. Mtry was an obvious choice as the default Random Forest hyperparameter, which tells Random Forest how many variables to consider when performing each split. Each time Random Forest splits, it choose mtry variables at random, then calculates the optimal split considering a split function to select a variable for the split. For the split function, I chose giri, extratrees, and hellinger, all of which are suitable for classification, but I weren’t sure which one would perform best given our data.

Optimizing for ROC, the winning parameters are an mtry of 7 predictors considered at each split, and the Hellinger split rule (see Figure 8). It's interesting that Hellinger won. I found some papers suggesting Hellinger handles imbalanced data well because it is insensitive to skew.

Reference: <https://www3.nd.edu/~nchawla/papers/DMKD11.pdf>

Reference: <https://medium.com/@evgeni.dubov/classifying-imbalanced-data-using-hellinger-distance-f6a4330d6f9a>

To better understand how this Random Forest model is working, let’s look at the minimal depth for each predictor. The more important variables will be the ones used to split higher up the tree.

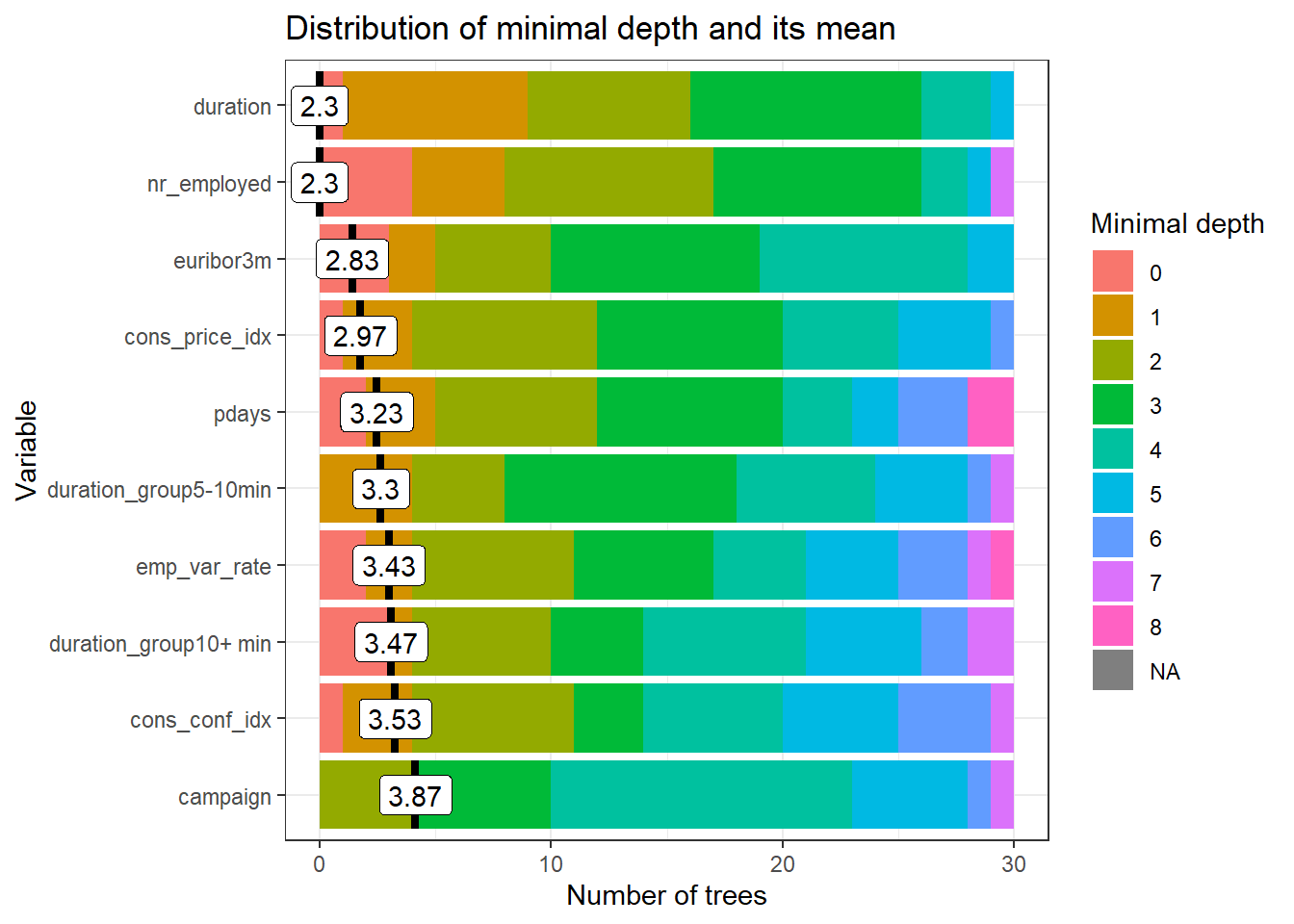


Figure – figure updated with our data set that removed the 3 yes default observations – small changes

From this I can see the most influential variable in the model by far is duration, and a derivative of duration, duration\_group10+ min is 3rd. I worry that duration of last call may be a strong “predictor” only because people who convert will of course have long duration of last call because they already decided to move forward and their last call probably is long because they are finalizing the deal. If so, then this model may be not so useful for predicting those who are likely to convert in the future. Predicting on duration feels like predicting rain based mostly on feeling raindrops starting to fall on my head.

Maybe duration should be replaced with the duration of the 2nd to last call? Or perhaps average duration of all calls? It may be worth following up with the company to refine the available variables.

Another way to interpret the model is to look at the influence of interactions:

Chart, bar chart, waterfall chart

Description automatically generated

Figure – this needs updated as the removal of default changed random forest results slightly – need to find where this is in Rinku’s code

Consumer Confidence most frequently interacted with other variables, like age, duration, and campaign, and in fact the min depth interaction is cons\_conf\_idx:duration.

## Metrics & Model Comparison

To determine optimal balance between accuracy, sensitivity, and specificity since we had a unbalanced data set, we used different cut-offs for each model. For our simple, complex, and LDA models, the cut-off used was 0.15. For the random forest model, the optimal cut-off used was xxx. Based on optimal sensitivity with good accuracy and specificity, our random forest model performed the best.

Table

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Accuracy | Sensitivity | Specificity |
| Simple | 0.87 | 0.83 | 0.87 |
| Complex | 0.87 | 0.86 | 0.87 |
| LDA | 0.89 | 0.91 | 0.71 |
| Random Forest | 0.84 | 0.94 | 0.83 |

Chart

Description automatically generated

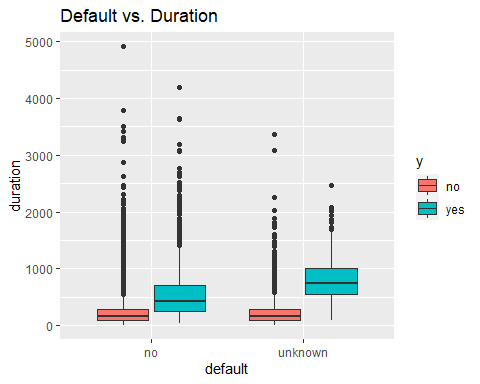
Figure . ROC curves for the top four models.

# Conclusion & Final Recommendations

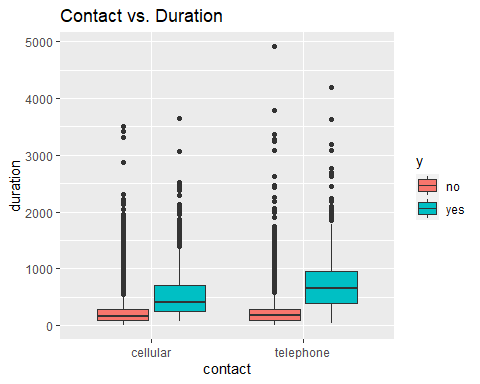
While these models performed well given the data, especially Random Forest, we recommend revisiting the problem with duration. It appears to have the most influence yet logically seems like any model based on it would only be useful to “predict” conversion after people have already converted. We would encourage rethinking about which data to include in the training set based on that data’s availability about people at the time accurate predictions would be most valuable to the business.

|  |
| --- |
| **Exploring an alternative model without duration…**  We didn’t have time to fully explore this, but as an experiment, I built a Random Forest model that excluded duration and duration\_group entirely, and here are the variables with min depth:  Chart, bar chart  Description automatically generated  If the purpose of the model is to predict which people to contact in the first place based on who is more likely to convert, then this model may be more effective since it seems to be driven mainly by predictors known in advance of any contact.  As for interactions:  Chart, bar chart  Description automatically generated  Consumer confidence index is again the most influential interaction, this time with campaign.  More work would need to be done to clear away other variables that would be unknown for people who have not yet been contacted, or perhaps were only contacted in the distant past, or for other campaigns. Then we would need to redo this analysis, but unfortunately we are out of time. Still, I wanted to call out this finding and possible next steps. |

# Appendix



Figure



Figure

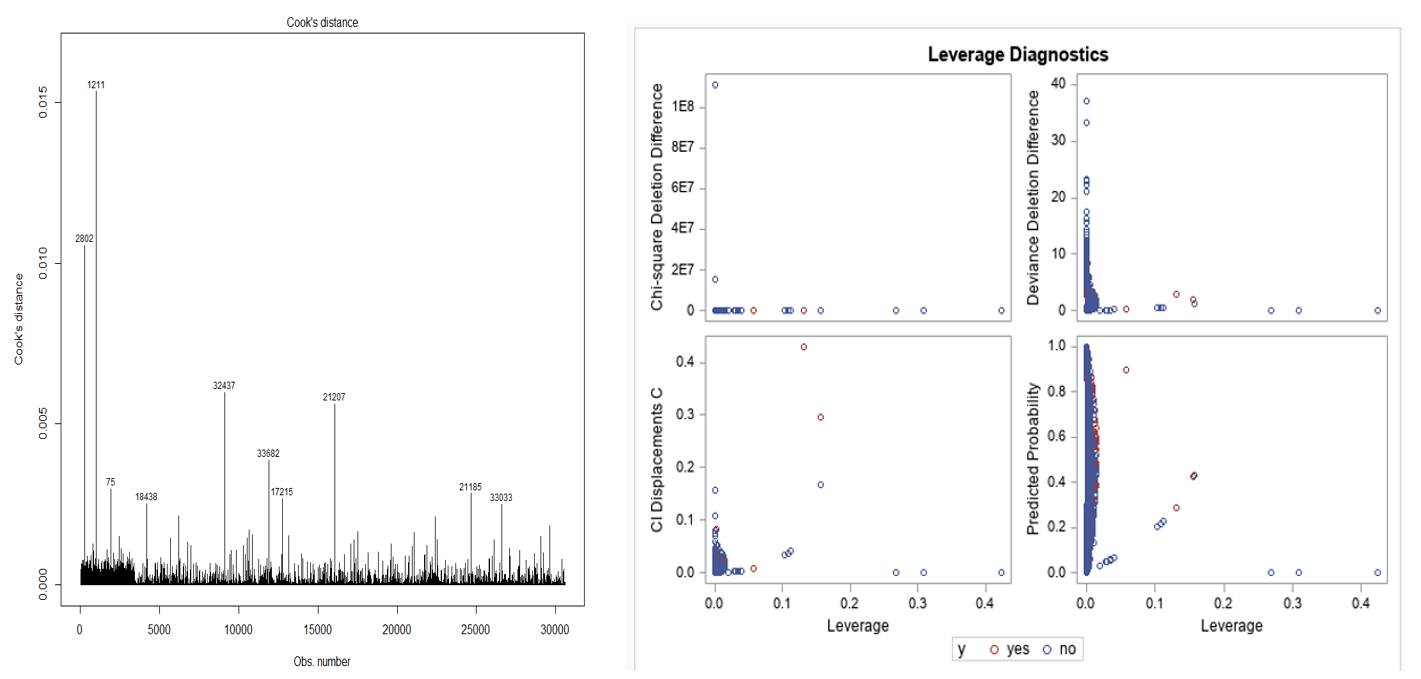


Figure 10 Cook’s D and Leverage plot from the model with full training data set:

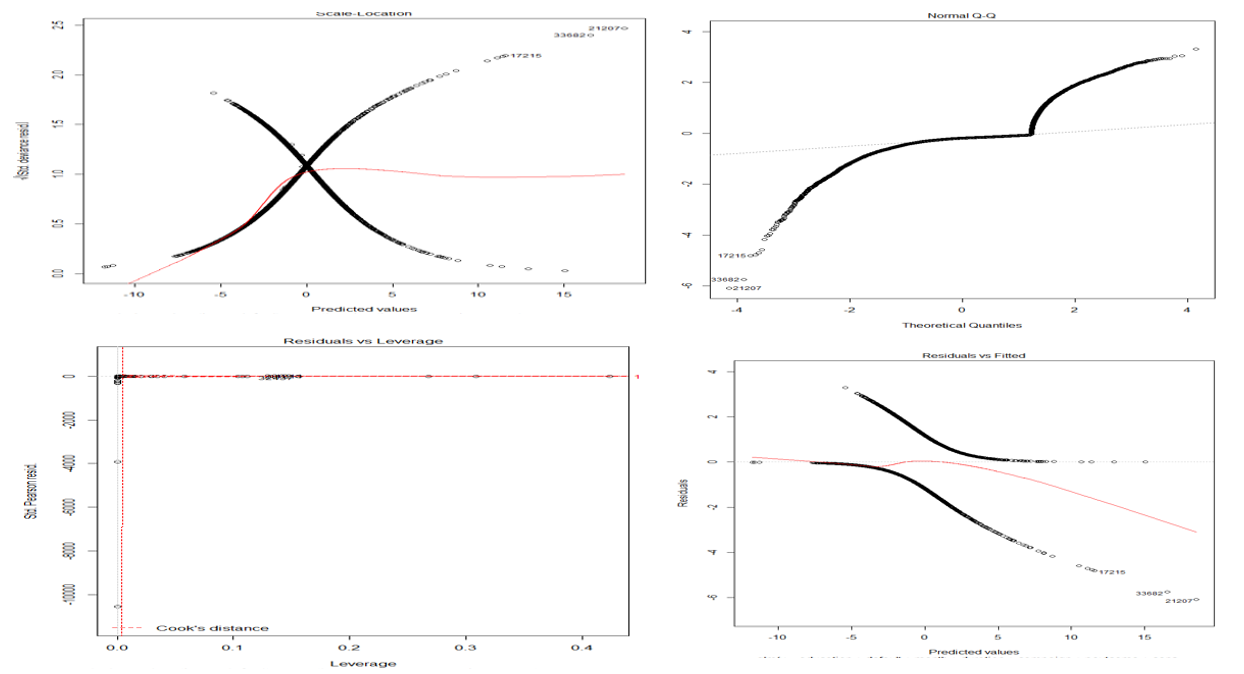


Figure 11 Residual Plots from the model with full training data set.

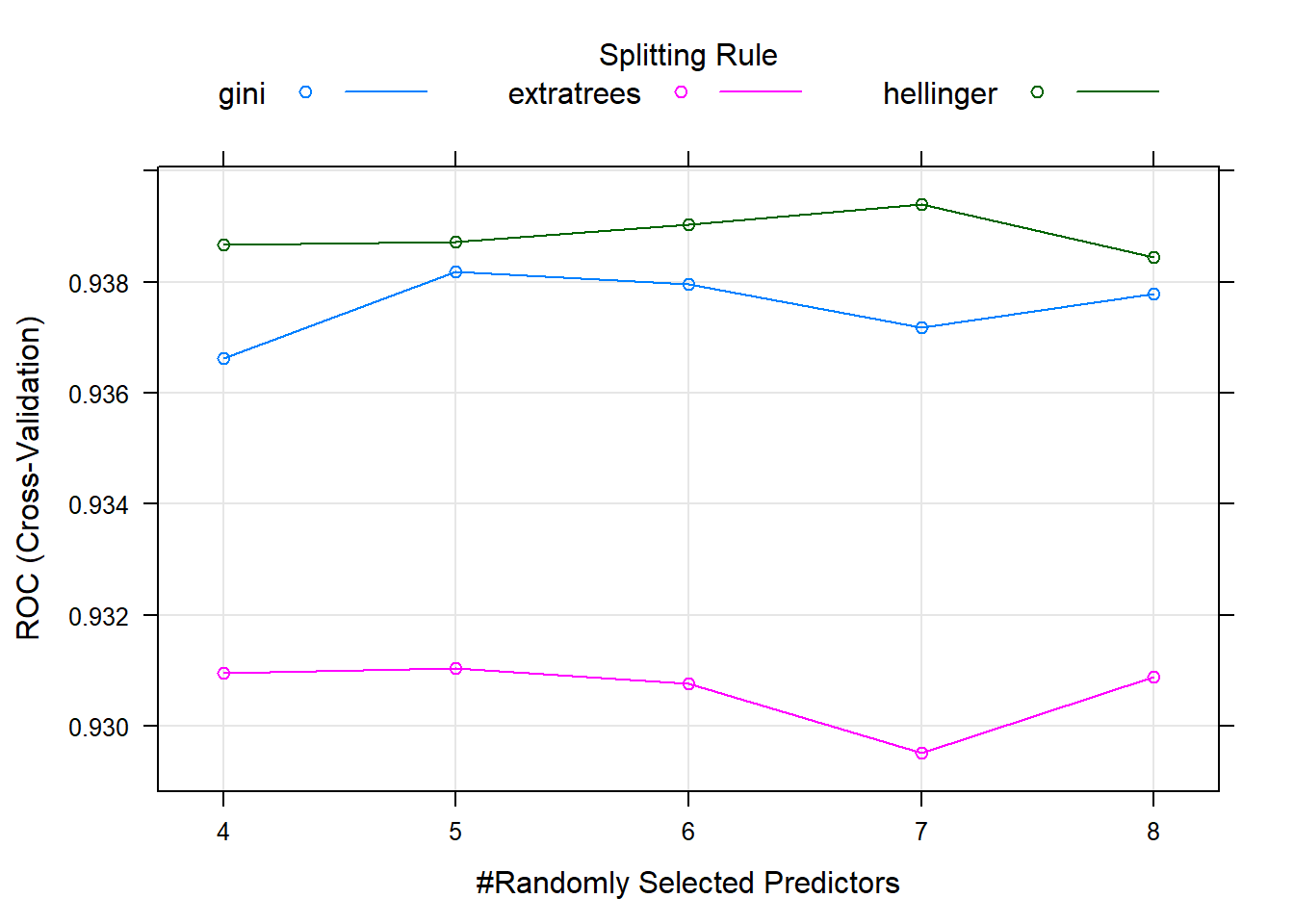


Figure 12

Link to Rmarkdown knit file (full notes/comments with graphics): <https://github.com/megball/6372-project-2/blob/main/Rmarkdown/Project2.html>

## R Code

#load libraries  
library(dplyr)  
library(tidyverse)  
library(ggplot2)  
library(caret)  
library(e1071)  
library(class)  
library(gridExtra)  
library(summarytools)  
library(gt)  
library(corrplot)  
library(janitor)  
library(tidyselect)  
library(GGally)  
library(randomForest)  
library(car)  
library(ROCR)  
library(MASS)  
library(glmnet)  
library(pROC)  
library(pacman)  
library(ranger)  
library(randomForestExplainer)  
library(broom)  
#full <- read\_delim(here::here("data", "bank-additional-full.csv"),';')  
full <- read.csv(file.choose(), sep=';')  
str(full)  
head(full)  
nrow(full)   
ncol(full)  
  
# Clean up column names  
full <- janitor::clean\_names(full)  
summary(full)  
  
#print(dfSummary(full, graph.magnif = 0.75), method = 'browser')  
str(full)  
  
# Check for missing values  
tibble(variable = names(colSums(is.na(full))),  
 missing = colSums(is.na(full))) %>%   
 gt() %>%   
 tab\_header(title = "Missing Values in Data")   
#remove "unknowns" based on small sample sizes compared to full data set  
df <- full %>% filter(loan != "unknown")  
nrow(df)  
#down to 40,198 obs  
df <- df %>% filter(marital != "unknown")  
nrow(df)  
#down to 40,119 obs  
df <- df %>% filter(education != "unknown")  
nrow(df)  
#down to 38,437 obs  
#remove unknowns from job  
df <- df %>% filter(job != "unknown")  
nrow(df)  
#down to 38,245 obs  
#remove yes from default - only 3, and all 3 are "no"  
df <- df %>% filter(default != "yes")  
nrow(df)  
#down to 38,242 obs  
str(df)  
#recheck summary  
summary(df)  
summary(df)  
#change some variables to factor  
cols <- c("job", "marital", "education", "housing","loan","contact","month","day\_of\_week","default","poutcome","y")  
df[cols] <- lapply(df[cols], factor)   
str(df)  
  
#make sure "success" level is defined as "yes"  
str(df$y)  
#run first pass PCA to see if we have useful numeric predictors  
df.numeric <- df[ , sapply(df, is.numeric)]  
pc.result<-prcomp(df.numeric,scale.=TRUE)  
pc.scores<-pc.result$x  
pc.scores<-data.frame(pc.scores)  
pc.scores$y<-df$y  
  
#pc.scores  
  
#Scree plot  
eigenvals<-(pc.result$sdev)^2  
eigenvals  
plot(1:10,eigenvals/sum(eigenvals),type="l",main="Scree Plot PC's",ylab="Prop. Var. Explained",ylim=c(0,1))  
cumulative.prop<-cumsum(eigenvals/sum(eigenvals))  
lines(1:10,cumulative.prop,lty=2)  
  
#Use ggplot2 to plot the first few pc's  
ggplot(data = pc.scores, aes(x = PC1, y = PC2)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")  
  
#There is some separation, but it is not in a way we would hope for our response variable  
ggplot(data = pc.scores, aes(x = PC2, y = PC3)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")  
ggplot(data = pc.scores, aes(x = PC3, y = PC4)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")  
df.numeric2 <- df.numeric %>% dplyr::select(-c(pdays, campaign, previous))  
pc.result2<-prcomp(df.numeric2,scale.=TRUE)  
pc.scores2<-pc.result2$x  
pc.scores2<-data.frame(pc.scores2)  
pc.scores2$y<-df$y  
#pc.scores2  
  
#Scree plot  
eigenvals2<-(pc.result2$sdev)^2  
eigenvals2  
plot(1:7,eigenvals2/sum(eigenvals2),type="l",main="Scree Plot PC's",ylab="Prop. Var. Explained",ylim=c(0,1))  
cumulative.prop2<-cumsum(eigenvals2/sum(eigenvals2))  
lines(1:7,cumulative.prop2,lty=2)  
  
#Use ggplot2 to plot the first few pc's  
ggplot(data = pc.scores2, aes(x = PC1, y = PC2)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")  
  
#There is some separation, but it is not in a way we would hope for our response variable  
ggplot(data = pc.scores2, aes(x = PC2, y = PC3)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")  
  
ggplot(data = pc.scores2, aes(x = PC3, y = PC4)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")  
#ggpairs(df,columns=1:18, aes(colour=y))  
ggpairs(df,columns=2:7, aes(colour=y))  
ggpairs(df, columns=14:18, aes(colour=y))  
df\_yes <- df %>% filter(y=="yes")  
  
#summary(df\_yes)  
# Nothing interesting found in the below code so commenting it out  
# ggplot(bank\_additional\_full, aes(x=age, y=emp.var.rate)) +  
# geom\_point(size=1, shape="circle") +  
# ggtitle("Employment Variation Rate vs Age") +   
# facet\_wrap(~ y)  
  
ggplot(df, aes(x=age, y=duration, color = y)) + geom\_point(size=1, shape="circle") + ggtitle("Duration vs Age")  
ggplot(df, aes(x = age, y = cons\_price\_idx, color = y)) +   
 geom\_point(size = 1, shape = "circle") +   
 ggtitle("Consumer Price Index vs Age")  
  
#Checking collinearlity using box plots  
ggplot(df, aes(x = age, y = cons\_price\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Consumer Price Index vs Age")  
  
ggplot(df, aes(x = duration , y = age, fill = y)) +   
 geom\_boxplot() + ggtitle("Age vs. duration")  
  
ggplot(df, aes(x = cons\_price\_idx , y = cons\_conf\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("cons.price.idx vs. cons.conf.idx")  
  
ggplot(df, aes(x = cons\_price\_idx , y = emp\_var\_rate, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("cons.price.idx vs. emp.var.rate")  
  
ggplot(df) + geom\_histogram(mapping = aes(x = nr\_employed, fill = y)) +  
 ggtitle("Distribution of 'y' by nr.employed")  
  
# ggplot(bank\_additional\_full, aes(x=age, y=education)) +  
# geom\_point(size=1, shape="circle") +  
# ggtitle("Education vs Age") +  
# facet\_wrap(~ y)  
ggplot(df)+   
 geom\_histogram(mapping = aes(x = age, fill = y)) +   
 ggtitle("Distribution of 'y' by age")  
#Age\_Grp - split the data into age groups "17-31","32-37" ,"38-47", "47-55", ">55" (based in IQR)  
df$Age\_Grp <- cut(df$age, breaks = c(16,31,37,46,55,98), labels = c("17-31","32-37" ,"38-47", "47-55", ">55"))  
  
#validate the cut command  
#df %>% filter(!$Age\_Grp %in% c("17-31","32-37" ,"38-47", "47-55", ">55"))  
#df %>% filter(df$age==55)  
  
ggplot(df) +   
 geom\_bar(mapping = aes(x=Age\_Grp, fill = y)) +   
 ggtitle("Distribution of 'y' by Age\_Grp") +   
 ylab("Cnt") +   
 xlab("Age Group")  
ggplot(df) +   
 geom\_histogram(mapping = aes(x=pdays, fill=y))  
#zoom in for ones that were previously contacted  
df %>%   
 filter(pdays < 999) %>%   
 ggplot() +   
 geom\_histogram(mapping = aes(x=pdays, fill=y))  
df$prevly\_Cntctd <- as.factor(case\_when(df$pdays==999 ~ "No", !df$pdays==999 ~ "Yes"))  
  
#Validate previously contacted variable  
  
#df %>% filter(!df$pdays==999)  
  
ggplot(df) +   
 geom\_bar(mapping = aes(x=prevly\_Cntctd, fill = y)) +   
 ggtitle("Number of 'y' by whether customers were prev.contacted or not") +  
 ylab("Cnt") +   
 xlab("Previously contacted?")  
ggplot(df) +   
 geom\_histogram(mapping = aes(x=campaign, fill=y)) +   
 ggtitle("Distribution of 'y' by campaign")  
ggplot(df) +   
 geom\_bar(mapping = aes(x=job, fill = y)) +   
 coord\_flip() + #Added coord flip here to make it more readable  
 ggtitle("Number of 'y' by job") +   
 ylab("Count") +   
 xlab("Job")  
df2 <- df %>%   
 group\_by(job) %>%   
 count(y) %>%   
 mutate(job\_conv = n/sum(n)) %>%   
 filter(y == "yes")  
  
ggplot(df2, aes(x=job, y=job\_conv)) +   
 geom\_point() +   
 coord\_flip()   
ggplot(data = df) +   
 geom\_bar(mapping = aes(x = marital, fill = y)) +   
 ggtitle("Number of 'y' by marital") +   
 ylab("Cnt") +   
 xlab("marital")  
summary(df$duration)  
df$duration\_group <-cut(df$duration,breaks = c(-Inf,300,600,Inf),labels = c("0-5min", "5-10min","10+ min"))  
  
# Check for missing values  
tibble(variable = names(colSums(is.na(df))),  
 missing = colSums(is.na(df))) %>%   
 gt() %>%   
 tab\_header(title = "Missing Values in Data")  
  
df3 <- df %>%   
 group\_by(duration\_group) %>%   
 count(y) %>%   
 mutate(duration\_group\_conv = n/sum(n)) %>%   
 filter(y == "yes")  
  
df3  
#ggplot(df3, aes(x=duration\_group, y=duration\_group\_conv)) + geom\_point() + facet\_wrap(~ y)  
prop.table(table(df$prevly\_Cntctd,df$duration\_group),2)  
plot(prevly\_Cntctd~duration\_group,data=df,col=c("purple","green"))  
prop.table(table(df$prevly\_Cntctd,df$y),2)  
plot(prevly\_Cntctd~y,data=df,col=c("purple","green"))  
prop.table(table(df$education,df$marital),2)  
plot(education~marital,data=df,col=c("purple","green","blue","yellow","orange","red","black"))  
  
prop.table(table(df$duration\_group,df$y),2)  
plot(duration\_group~y,data=df,col=c("purple","green","blue"))  
prop.table(table(df$y,df$duration\_group),2)  
df %>%   
 group\_by(education) %>%   
 count(y) %>%   
 mutate(education\_conv = n/sum(n)) %>%   
 filter(y == "yes")  
df %>%   
 group\_by(education) %>%   
 count(y) %>%   
 mutate(education\_conv = n/sum(n)) %>%   
 filter(y == "yes")  
# Convert data to numeric  
corrs <- data.frame(lapply(df, as.integer))  
  
# Plot the graph  
ggcorr(corrs,  
 method = c("pairwise", "spearman"),  
 nbreaks = 6,  
 hjust = 0.8,  
 label = TRUE,  
 label\_size = 3,  
 color = "grey50")  
#move response variable to end of data set  
df <- df %>% relocate(y, .after = last\_col())  
  
#randomly sample 10k obs  
sample10k <- sample\_n(df, 10000)  
  
#down sample to balance response  
set.seed(1)  
downsample <- downSample(x = sample10k[, -24],  
y = sample10k$y)  
table(downsample$Class)  
RFcontrol <- rfeControl(functions=rfFuncs, method="cv", number=5, verbose = FALSE)  
set.seed(123)  
subsets <- c(1:5, 10, 15, 20)  
RFresults <- rfe(downsample[,1:23], downsample[[24]], sizes=subsets, rfeControl=RFcontrol)  
RFresults  
varImp(RFresults)  
#save dataset to this point  
#df\_clean <- write.csv(df, "df\_clean.csv", row.names = FALSE)  
#open saved dataframe  
#df <- read.csv(here::here("data", "df\_clean.csv"), stringsAsFactors = TRUE)  
#str(df)  
summary(df)  
#38242 obs. of 24 variables  
  
set.seed(1234)   
df\_yes <- df %>% filter(y=='yes')  
df\_No <- df %>% filter(y=='no')  
  
num\_rows\_yes <- nrow(df\_yes) #4,258  
num\_rows\_no <- nrow(df\_No) #33,984  
  
train\_idx\_yes <- sample(1:num\_rows\_yes, 0.8 \* num\_rows\_yes)  
train\_yes <- df\_yes[train\_idx\_yes, ]  
test\_yes <- df\_yes[-train\_idx\_yes, ]  
  
nrow(train\_yes) #3,406  
nrow(test\_yes) #852  
  
train\_idx\_no <- sample(1:num\_rows\_no, 0.8 \* num\_rows\_no)  
train\_no <- df\_No[train\_idx\_no, ]  
test\_no <- df\_No[-train\_idx\_no, ]  
nrow(train\_no) #27,187  
nrow(test\_no) #6797  
  
train <- rbind(train\_yes, train\_no)  
test <- rbind(test\_yes, test\_no)  
nrow(train) #30,593  
nrow(test) #7,649  
nrow(train %>% filter(y=='yes')) #3,406  
nrow(test %>% filter(y=='yes')) #852  
summary(train)  
#30593 obs. of 24 variables  
#write.csv(train, "data/train.csv", row.names = FALSE)  
#write.csv(test, "data/test.csv", row.names = FALSE)  
# Run Initial Logistic Regression  
#Simple regression model  
simple.log<-glm(y~.,family="binomial",data=train)  
summary(simple.log)  
exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)  
train\_simple <- train %>% dplyr::select(-pdays)  
#Check vifs again  
simple.log<-glm(y~.,family="binomial",data=train\_simple)  
summary(simple.log)  
#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)  
train\_simple\_2 <- train\_simple %>% dplyr::select(-nr\_employed, -emp\_var\_rate )  
simple.log<-glm(y~.,family="binomial",data=train\_simple\_2)  
summary(simple.log)  
#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)  
train\_simple\_3 <- train\_simple\_2 %>% dplyr::select(-age)  
#Check model again  
simple.log<-glm(y~.,family="binomial",data=train\_simple\_3)  
summary(simple.log)  
#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)  
train\_simple\_4 <- train\_simple\_3 %>% dplyr::select(-marital, -housing, -loan, -day\_of\_week, -previous)  
#Check model again  
simple.log<-glm(y~.,family="binomial",data=train\_simple\_4)  
summary(simple.log)  
#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)  
#simple model -1   
simple.log <-  
 glm(  
 y ~ job + education + default + contact + month + duration + campaign +  
 poutcome + cons\_price\_idx + cons\_conf\_idx + euribor3m + Age\_Grp + prevly\_Cntctd +  
 duration\_group,  
 family = "binomial",  
 data = train  
 )  
#simple.log<-glm(y~.,family="binomial",data=train\_simple\_3)  
summary(simple.log)  
exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)  
  
#Prediction using simple model  
fit.pred.simple<-predict(simple.log,newdata=test, type="response")  
class.simple<-factor(ifelse(fit.pred.simple>0.5,"yes","no"),levels=c("no","yes"))  
  
# use caret and compute a confusion matrix  
confusionMatrix(class.simple,test$y, positive = "yes")  
# Feature selection using step  
full.log<-glm(y~.,family="binomial",data=train)  
step.log<-full.log %>% stepAIC(trace=FALSE)  
summary(step.log)  
#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)  
#Remove variables with high vifs and run the model again  
train\_step <- train %>% dplyr::select(-emp\_var\_rate, euribor3m)  
#Check vifs again  
full.log<-glm(y~.,family="binomial",data=train\_step)  
step.log<-full.log %>% stepAIC(trace=FALSE)  
summary(step.log)  
#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)   
train\_step\_2 <- train\_step %>% dplyr::select(-nr\_employed)  
full.log<-glm(y~.,family="binomial",data=train\_step\_2)  
step.log<-full.log %>% stepAIC(trace=FALSE)  
summary(step.log)  
#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)  
train\_step\_3 <- train\_step\_2 %>% dplyr::select(-poutcome )  
#Check vifs again  
full.log<-glm(y~.,family="binomial",data=train\_step\_3)  
step.log<-full.log %>% stepAIC(trace=FALSE)  
summary(step.log)  
#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)  
#Run step model again  
full.log <-  
 glm(  
 y ~ job + default + contact + month + duration + campaign + previous + cons\_price\_idx +  
 cons\_conf\_idx + euribor3m + Age\_Grp + prevly\_Cntctd + duration\_group,  
 family = "binomial",  
 data = train  
 )  
  
#full.log<-glm(y~.,family="binomial",data=train\_step\_3)  
step.log<-full.log %>% stepAIC(trace=FALSE)  
summary(step.log)  
exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)  
  
#Predicting using step   
fit.pred.step<-predict(step.log,newdata=test,type="response")  
test$y[1:15]  
fit.pred.step[1:15]  
class.step1<-factor(ifelse(fit.pred.step>0.5,"yes","no"),levels=c("no","yes"))  
  
# use caret and compute a confusion matrix  
confusionMatrix(class.step1,test$y, positive = "yes")  
 #Acc 91%, Sens. 44%, Spec. 97%  
dat.train.x <- model.matrix(y~.,train)  
dat.train.y<-as.matrix(train[,24])  
cvfit <- cv.glmnet(dat.train.x, dat.train.y, family = "binomial", type.measure = "class", nlambda = 1000)  
plot(cvfit)  
coef(cvfit, s = "lambda.min")  
#CV misclassification error rate is little below .1  
  
print("CV Error Rate:")  
cvfit$cvm[which(cvfit$lambda==cvfit$lambda.min)]  
#"CV Error Rate:"  
#0.09021672  
  
#Optimal penalty  
print("Penalty Value:")  
cvfit$lambda.min  
#"Penalty Value:"  
#0.0008648178  
  
finalmodel<-glmnet(dat.train.x, dat.train.y, family = "binomial",lambda=cvfit$lambda.min)  
finalmodel$call  
finalmodel  
dat.test.x<-model.matrix(y~.,test)  
fit.pred.lasso <- predict(finalmodel, newx = dat.test.x, type = "response")  
test$y[1:15]  
fit.pred.lasso[1:15]  
#confusion matrix at 0.5 cutoff  
class.lasso1<-factor(ifelse(fit.pred.lasso>0.5,"yes","no"),levels=c("no","yes"))  
# use caret and compute a confusion matrix  
confusionMatrix(class.lasso1,test$y, positive = "yes")  
#Acc 91.5%, Sens. 45%, Spec. 97%  
  
#ROCR  
results.lasso <-  
 prediction(fit.pred.lasso, test$y, label.ordering = c("no", "yes"))  
roc.lasso = performance(results.lasso, measure = "tpr", x.measure = "fpr")  
plot(roc.lasso, colorize = TRUE)  
abline(a = 0, b = 1)  
  
#step model prediction  
results.step <-  
 prediction(fit.pred.step, test$y, label.ordering = c("no", "yes"))  
roc.step = performance(results.step, measure = "tpr", x.measure = "fpr")  
  
simple.log <- glm(y ~ ., family = "binomial", data = train)  
fit.pred.origin <- predict(simple.log, newdata = test, type = "response")  
results.origin <-  
 prediction(fit.pred.origin, test$y, label.ordering = c("no", "yes"))  
roc.origin = performance(results.origin, measure = "tpr", x.measure = "fpr")  
  
#plot ROC curves  
plot(roc.lasso)  
plot(roc.step, col = "orange", add = TRUE)  
plot(roc.origin, col = "blue", add = TRUE)  
legend(  
 "bottomright",  
 legend = c("Lasso", "Stepwise", "Simple"),  
 col = c("black", "orange", "blue"),  
 lty = 1,  
 lwd = 1  
)  
abline(a = 0, b = 1)  
#Playing with different cut offs  
cutoff <- 0.5  
class.lasso <-  
 factor(ifelse(fit.pred.lasso > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
class.step <-  
 factor(ifelse(fit.pred.step > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
class.simple <-  
 factor(ifelse(fit.pred.simple > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
  
#Confusion Matrix for Lasso  
conf.lasso <- table(class.lasso, test$y)  
print("Confusion matrix for LASSO")  
conf.lasso  
  
#Confusion Matrix for step  
conf.step <- table(class.step, test$y)  
print("Confusion matrix for Stepwise")  
conf.step  
  
#Confusion Matrix for simple  
conf.simple <- table(class.simple, test$y)  
print("Confusion matrix for Stepwise")  
conf.simple  
  
#Accuracy of LASSO and Stepwise  
print("Overall accuracy for LASSO and Stepwise respectively")  
sum(diag(conf.lasso)) / sum(conf.lasso)  
sum(diag(conf.step)) / sum(conf.step)  
print("Alternative calculations of accuracy")  
Acc\_LASSO\_0.5 <- mean(class.lasso == test$y)  
Acc\_STEP\_0.5 <- mean(class.step == test$y)  
Acc\_SIMPLE\_0.5 <- mean(class.simple == test$y)  
  
#Confusion Matrix for cut off =05  
lasso\_0.5 <- confusionMatrix(conf.lasso)  
step\_0.5 <- confusionMatrix(conf.step)  
simple\_0.5 <- confusionMatrix(conf.simple)  
cutoff <- 0.1  
class.lasso <-  
 factor(ifelse(fit.pred.lasso > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
class.step <-  
 factor(ifelse(fit.pred.step > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
class.simple <-  
 factor(ifelse(fit.pred.simple > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
conf.lasso<-table(class.lasso,test$y)  
print("Confusion matrix for LASSO")  
conf.lasso  
conf.step<-table(class.step,test$y)  
print("Confusion matrix for Stepwise")  
conf.step  
conf.simple<-table(class.simple,test$y)  
print("Confusion matrix for Stepwise")  
conf.simple  
print("Overall accuracy for LASSO and Stepwise respectively")  
sum(diag(conf.lasso))/sum(conf.lasso)  
sum(diag(conf.step))/sum(conf.step)  
print("Alternative calculations of accuracy")  
Acc\_LASSO\_0.1 <- mean(class.lasso==test$y)  
Acc\_STEP\_0.1 <-mean(class.step==test$y)  
Acc\_SIMPLE\_0.1<-mean(class.simple==test$y)  
lasso\_0.1<-confusionMatrix(conf.lasso, positive = "yes")  
step\_0.1<-confusionMatrix(conf.step, positive = "yes")  
simple\_0.1<-confusionMatrix(conf.simple, positive = "yes")  
cutoff<-0.15  
class.lasso<-factor(ifelse(fit.pred.lasso>cutoff,"yes","no"),levels=c("no","yes"))  
class.step<-factor(ifelse(fit.pred.step>cutoff,"yes","no"),levels=c("no","yes"))  
class.simple<-factor(ifelse(fit.pred.simple>cutoff,"yes","no"),levels=c("no","yes"))  
conf.lasso<-table(class.lasso,test$y)  
print("Confusion matrix for LASSO")  
conf.lasso  
conf.step<-table(class.step,test$y)  
print("Confusion matrix for Stepwise")  
conf.step  
  
confusionMatrix(conf.step, positive = "yes")  
conf.simple<-table(class.simple,test$y)  
print("Confusion matrix for Stepwise")  
conf.simple  
print("Overall accuracy for LASSO and Stepwise respectively")  
sum(diag(conf.lasso))/sum(conf.lasso)  
sum(diag(conf.step))/sum(conf.step)  
print("Alternative calculations of accuracy")  
Acc\_LASSO\_0.15 <- mean(class.lasso==test$y)  
Acc\_STEP\_0.15 <-mean(class.step==test$y)  
Acc\_SIMPLE\_0.15<-mean(class.simple==test$y)  
lasso\_0.15<-confusionMatrix(conf.lasso, positive = "yes")  
step\_0.15<-confusionMatrix(conf.step, positive = "yes")  
simple\_0.15<-confusionMatrix(conf.simple, positive = "yes")  
cutoff <- 0.2  
class.lasso <-  
 factor(ifelse(fit.pred.lasso > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
class.step <-  
 factor(ifelse(fit.pred.step > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
class.simple <-  
 factor(ifelse(fit.pred.simple > cutoff, "yes", "no"),  
 levels = c("no", "yes"))  
  
#Confusion Matrix for Lasso  
conf.lasso <- table(class.lasso, test$y)  
print("Confusion matrix for LASSO")  
conf.lasso  
#Confusion Matrix for step  
conf.step <- table(class.step, test$y)  
print("Confusion matrix for Stepwise")  
conf.step  
  
#Confusion Matrix for simple  
conf.simple <- table(class.simple, test$y)  
print("Confusion matrix for Stepwise")  
conf.simple  
  
#Accuracy of LASSO and Stepwise  
print("Overall accuracy for LASSO and Stepwise respectively")  
sum(diag(conf.lasso)) / sum(conf.lasso)  
sum(diag(conf.step)) / sum(conf.step)  
#print("Alternative calculations of accuracy")  
#Acc\_LASSO\_0.2 <- mean(class.lasso==test$y)  
#Acc\_STEP\_0.2 <-mean(class.step==test$y)  
#Acc\_SIMPLE\_0.2<-mean(class.simple==test$y)  
  
#Confusion Matrix for cut off =0.2  
lasso\_0.2 <- confusionMatrix(conf.lasso)  
step\_0.2 <- confusionMatrix(conf.step)  
simple\_0.2 <- confusionMatrix(conf.simple)  
Sensitivity\_simple <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "Simple\_Sensitivty" = c(  
 simple\_0.1$byClass[1],  
 simple\_0.15$byClass[1],  
 simple\_0.2$byClass[1],  
 simple\_0.5$byClass[1]  
 )  
 )  
Sensitivity\_step <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "Step\_Sensitivity" = c(  
 step\_0.1$byClass[1],  
 step\_0.15$byClass[1],  
 step\_0.2$byClass[1],  
 step\_0.5$byClass[1]  
 )  
 )  
Sensitivity\_lasso <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "LASSO\_Sensitivity" = c(  
 lasso\_0.1$byClass[1],  
 lasso\_0.15$byClass[1],  
 lasso\_0.2$byClass[1],  
 lasso\_0.5$byClass[1]  
 )  
 )  
Specificity\_simple <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "Simple\_Specificity" = c(  
 simple\_0.1$byClass[2],  
 simple\_0.15$byClass[2],  
 simple\_0.2$byClass[2],  
 simple\_0.5$byClass[2]  
 )  
 )  
Specificity\_step <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "Step\_Specificity" = c(  
 step\_0.1$byClass[2],  
 step\_0.15$byClass[2],  
 step\_0.2$byClass[2],  
 step\_0.5$byClass[2]  
 )  
 )  
Specificity\_lasso <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "LASSO\_Specificity" = c(  
 lasso\_0.1$byClass[2],  
 lasso\_0.15$byClass[2],  
 lasso\_0.2$byClass[2],  
 lasso\_0.5$byClass[2]  
 )  
 )  
Accuracy\_simple <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "Simple\_Accuracy" = c(  
 simple\_0.1$overall[1],  
 simple\_0.15$overall[1],  
 simple\_0.2$overall[1],  
 simple\_0.5$overall[1]  
 )  
 )  
Accuracy\_step <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "Step\_Accuracy" = c(  
 step\_0.1$overall[1],  
 step\_0.15$overall[1],  
 step\_0.2$overall[1],  
 step\_0.5$overall[1]  
 )  
 )  
Accuracy\_lasso <-  
 data.frame(  
 "CutOff" = c("0.1", "0.15", "0.2", "0.5"),  
 "LASSO\_Accuracy" = c(  
 lasso\_0.1$overall[1],  
 lasso\_0.15$overall[1],  
 lasso\_0.2$overall[1],  
 lasso\_0.5$overall[1]  
 )  
 )  
Sensitivity <-  
 cbind(  
 Sensitivity\_simple,  
 Sensitivity\_step$Step\_Sensitivity,  
 Sensitivity\_lasso$LASSO\_Sensitivity  
 )  
Specificity <-  
 cbind(  
 Specificity\_simple,  
 Specificity\_step$Step\_Specificity,  
 Specificity\_lasso$LASSO\_Specificity  
 )  
Accuracy <-  
 cbind(Accuracy\_simple,  
 Accuracy\_step$Step\_Accuracy,  
 Accuracy\_lasso$LASSO\_Accuracy)  
Sensitivity  
Specificity  
Accuracy  
#compare all at 0.15 cutoff  
Sensitivity <-  
 data.frame(  
 "Model" = c("Simple", "Step", "LASSO"),  
 "Sensitivity" = c(  
 simple\_0.15$byClass[1],  
 step\_0.15$byClass[1],  
 lasso\_0.15$byClass[1]  
 )  
 )  
  
Specificity <-  
 data.frame("Specificity" = c(  
 simple\_0.15$byClass[2],  
 step\_0.15$byClass[2],  
 lasso\_0.15$byClass[2]  
 ))  
  
Accuracy <-  
 data.frame("Accuracy" = c(  
 simple\_0.15$overall[1],  
 step\_0.15$overall[1],  
 lasso\_0.15$overall[1]  
 ))  
  
Overall <- cbind(Sensitivity, Specificity, Accuracy)  
Overall  
plot(step.log, which = 4, id.n = 10) #Cooks D plot  
#step.log.data  
step.log.data <- augment(step.log) %>%   
 mutate(index = 1:n())   
  
ggplot(step.log.data, aes(index, .std.resid)) +   
 geom\_point(aes(color = y)) +   
 ggtitle("Residual plot")  
#Residual diagnostics   
plot(step.log)  
  
#examine outliers 1   
nrow(train) #30593  
train2 <- train %>% dplyr::filter(!rownames(train) %in% c("17215","31370","33679"))  
nrow(train2)   
  
#Residual diagnostics   
step.log2 <- glm(  
 y ~ job + default + contact + month + duration +  
 campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m +  
 Age\_Grp + prevly\_Cntctd + duration\_group,  
 family = "binomial",  
 data = train2  
)  
  
#full.log<-glm(y~.,family="binomial",data=train)  
summary(step.log2)  
plot(step.log2)  
#examine outliers 2   
nrow(train2) #30590  
train3 <- train2 %>% dplyr::filter(!rownames(train2) %in% c("32754","18438","21183"))  
nrow(train3)   
  
#Residual diagnostics   
step.log3 <- glm(  
 y ~ job + default + contact + month + duration +  
 campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m +  
 Age\_Grp + prevly\_Cntctd + duration\_group,  
 family = "binomial",  
 data = train3  
)  
  
#full.log<-glm(y~.,family="binomial",data=train)  
summary(step.log3)  
plot(step.log3)  
train %>% dplyr::filter(rownames(train) %in% c("17215", "31370", "33679", "32754", "18438", "21183")) %>% dplyr::select(  
 y,  
 job,  
 default,  
 contact,  
 month,  
 duration,  
 campaign,  
 previous,  
 cons\_price\_idx,  
 cons\_conf\_idx,  
 euribor3m,  
 Age\_Grp,  
 prevly\_Cntctd,  
 duration\_group  
)  
fit.pred.step\_outlier<-predict(step.log3,newdata=test,type="response")  
class.step\_out<-factor(ifelse(fit.pred.step\_outlier>0.5,"yes","no"),levels=c("no","yes"))  
  
# use caret and compute a confusion matrix  
confusionMatrix(class.step\_out,test$y, positive = "yes")  
#computer memory issues - start with only one added interaction  
complex.log <-  
 glm(  
 y ~ job + default + contact + month + duration + campaign +  
 previous + cons\_price\_idx + cons\_conf\_idx + euribor3m +  
 Age\_Grp + prevly\_Cntctd + duration\_group + duration \* default,  
 family = "binomial",  
 data = train  
 )  
summary(complex.log)  
#exp(cbind("Odds ratio" = coef(complex.log), confint.default(complex.log, level = 0.95)))  
complex.log <-  
 glm(  
 y ~ job + default + contact + month + duration + campaign +  
 previous + cons\_price\_idx + cons\_conf\_idx + euribor3m +  
 Age\_Grp + prevly\_Cntctd + duration\_group + Age\_Grp \* education + campaign \*  
 duration + cons\_price\_idx \* euribor3m + month \* euribor3m,  
 family = "binomial",  
 data = train  
 )  
summary(complex.log)  
#exp(cbind("Odds ratio" = coef(complex.log), confint.default(complex.log, level = 0.95)))  
#complex.pred <- predict(complex.log, newdata = test, type="response")  
#numerical y vars  
ggplot(df, aes(x=month , y=emp\_var\_rate, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Month vs. emp.var.rate")  
  
ggplot(df, aes(x=default , y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. Duration")  
  
ggplot(df, aes(x=default , y=campaign, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. Campaign")  
  
ggplot(df, aes(x=default , y=cons\_price\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. cons.price.idx")  
  
ggplot(df, aes(x=default , y=euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. euribor3m")  
  
ggplot(df, aes(x=contact , y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Contact vs. Duration")  
  
ggplot(df, aes(x=contact , y=campaign, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Contact vs. Campaign")  
  
ggplot(df, aes(x=prevly\_Cntctd , y=cons\_price\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("prevly\_Cntctd vs. cons.price.idx")  
  
ggplot(df, aes(x=prevly\_Cntctd , y=euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("prevly\_Cntctdt vs. euribor3m")  
  
ggplot(df, aes(x=prevly\_Cntctd , y=campaign, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("prevly\_Cntctdt vs. campaign")  
  
ggplot(df, aes(x=prevly\_Cntctd , y=previous, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("prevly\_Cntctdt vs. previous")  
#tables for categoricals  
prop.table(table(df\_yes$default,df\_yes$month),2)  
prop.table(table(df\_No$y,df\_No$month),2)  
complex.log <-  
 glm(  
 y ~ job + default + contact + month + duration + campaign +  
 previous + cons\_price\_idx + cons\_conf\_idx + euribor3m +  
 Age\_Grp + prevly\_Cntctd + duration\_group + default \*  
 duration + contact \* duration + default \* month + month \* euribor3m,  
 family = "binomial",  
 data = train  
 )  
summary(complex.log)  
step.complex<-complex.log %>% stepAIC(trace=FALSE)  
summary(step.complex)  
complex.pred <- predict(step.complex, newdata = test, type="response")  
  
#ROCR  
results.complex<-prediction(complex.pred, test$y,label.ordering=c("no","yes"))  
roc.complex = performance(results.complex, measure = "tpr", x.measure = "fpr")  
plot(roc.complex,colorize = TRUE)  
abline(a=0, b= 1)  
cutoff<-0.5  
class.complex<-factor(ifelse(complex.pred>cutoff,"yes","no"),levels=c("no","yes"))  
  
#Confusion Matrix  
conf.complex<-table(class.complex,test$y)  
conf.complex  
complex<-confusionMatrix(conf.complex, positive = "yes")  
complex  
cutoff<-0.15  
class.complex<-factor(ifelse(complex.pred>cutoff,"yes","no"),levels=c("no","yes"))  
  
#Confusion Matrix  
conf.complex<-table(class.complex,test$y)  
conf.complex  
complex<-confusionMatrix(conf.complex, positive = "yes")  
complex  
cutoff<-0.3  
class.complex<-factor(ifelse(complex.pred>cutoff,"yes","no"),levels=c("no","yes"))  
  
#Confusion Matrix  
conf.complex<-table(class.complex,test$y)  
conf.complex  
complex<-confusionMatrix(conf.complex, positive = "yes")  
complex  
#Training Set  
train.lda.x <- train[ , sapply(train, is.numeric)]  
train.lda.y <- train$y  
fit.lda <- lda(train.lda.y ~ ., data = train.lda.x)  
pred.lda <- predict(fit.lda, newdata = train.lda.x)  
preds <- pred.lda$posterior  
preds <- as.data.frame(preds)  
pred <- prediction(preds[,2],train.lda.y)  
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  
auc.train <- performance(pred, measure = "auc")  
auc.train <- auc.train@y.values  
plot(roc.perf, colorize = TRUE)  
abline(a=0, b= 1)  
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  
#AUC = 0.922  
# Test Set  
test.lda.x <- test[ , sapply(test, is.numeric)]  
test.lda.y <- test$y  
pred.lda1 <- predict(fit.lda, newdata = test.lda.x)  
preds1 <- pred.lda1$posterior  
preds1 <- as.data.frame(preds1)  
pred1 <- prediction(preds1[,2],test.lda.y)  
roc.perf = performance(pred1, measure = "tpr", x.measure = "fpr")  
auc.train <- performance(pred1, measure = "auc")  
auc.train <- auc.train@y.values  
plot(roc.perf, colorize = TRUE)  
abline(a=0, b= 1)  
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  
#AUC = 0.919  
#running cv on train set using LDA  
nloops<-50 #number of CV loops  
ntrains<-dim(train.lda.x)[1] #No. of samples in training data set  
cv.aucs<-c()  
  
set.seed(123)  
for (i in 1:nloops){  
 index<-sample(1:ntrains,ntrains\*.8)  
 cvtrain.x<-train.lda.x[index,]  
 cvtest.x<-train.lda.x[-index,]  
 cvtrain.y<-train.lda.y[index]  
 cvtest.y<-train.lda.y[-index]  
   
 cvfit <- lda(cvtrain.y ~ ., data = cvtrain.x)  
 fit.pred <- predict(cvfit, newdata = cvtest.x)  
 preds.cv <- fit.pred$posterior  
 preds.cv <- as.data.frame(preds.cv)  
 pred.cv <- prediction(preds.cv[,2], cvtest.y)  
 roc.perf = performance(pred.cv, measure = "tpr", x.measure = "fpr")  
 auc.train <- performance(pred.cv, measure = "auc")  
 auc.train <- auc.train@y.values  
   
 cv.aucs[i]<-auc.train[[1]]  
}  
  
hist(cv.aucs)  
summary(cv.aucs)  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.9100 0.9187 0.9219 0.9217 0.9248 0.9336   
#test using just the numeric ones from our best step model  
fit.lda\_step <- lda(train.lda.y ~ duration + campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m, data = train.lda.x)  
pred.lda\_step <- predict(fit.lda\_step, newdata = train.lda.x)  
preds\_step <- pred.lda\_step$posterior  
preds\_step <- as.data.frame(preds\_step)  
pred\_step <- prediction(preds\_step[,2],train.lda.y)  
roc.perf\_step = performance(pred\_step, measure = "tpr", x.measure = "fpr")  
auc.train\_step <- performance(pred\_step, measure = "auc")  
auc.train\_step <- auc.train\_step@y.values  
plot(roc.perf\_step, colorize = TRUE)  
abline(a=0, b= 1)  
text(x = .40, y = .6,paste("AUC = ", round(auc.train\_step[[1]],3), sep = ""))  
#AUC = 0.911  
  
#running cv on train set using LDA with subset of numeric vars  
nloops<-50 #number of CV loops  
ntrains<-dim(train.lda.x)[1] #No. of samples in training data set  
cv.aucs\_2<-c()  
  
set.seed(123)  
  
for (i in 1:nloops) {  
 index <- sample(1:ntrains, ntrains \* .8)  
 cvtrain.x <- train.lda.x[index, ]  
 cvtest.x <- train.lda.x[-index, ]  
 cvtrain.y <- train.lda.y[index]  
 cvtest.y <- train.lda.y[-index]  
   
 cvfit\_2 <-  
 lda(  
 cvtrain.y ~ duration + campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m,  
 data = cvtrain.x  
 )  
 fit.pred\_2 <- predict(cvfit\_2, newdata = cvtest.x)  
 preds.cv\_2 <- fit.pred\_2$posterior  
 preds.cv\_2 <- as.data.frame(preds.cv\_2)  
 pred.cv\_2 <- prediction(preds.cv\_2[, 2], cvtest.y)  
 roc.perf\_2 = performance(pred.cv\_2, measure = "tpr", x.measure = "fpr")  
 auc.train\_2 <- performance(pred.cv\_2, measure = "auc")  
 auc.train\_2 <- auc.train\_2@y.values  
   
 cv.aucs\_2[i] <- auc.train\_2[[1]]  
}  
hist(cv.aucs\_2)  
summary(cv.aucs\_2)  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.9003 0.9074 0.9114 0.9107 0.9137 0.9233   
#run on test set  
# Test Set  
pred.lda1\_step <- predict(fit.lda\_step, newdata = test.lda.x)  
preds1\_step <- pred.lda1\_step$posterior  
preds1\_step <- as.data.frame(preds1\_step)  
pred1\_step <- prediction(preds1\_step[, 2], test.lda.y)  
roc.perf\_step2 = performance(pred1\_step, measure = "tpr", x.measure = "fpr")  
auc.train\_step2 <- performance(pred1\_step, measure = "auc")  
auc.train\_step2 <- auc.train\_step2@y.values  
plot(roc.perf\_step2, colorize = TRUE)  
abline(a = 0, b = 1)  
text(x = .40, y = .6, paste("AUC = ", round(auc.train\_step2[[1]], 3), sep = ""))  
#AUC = 0.903  
#test using just the numeric ones from our best step model  
fit.lda\_step2 <-  
 lda(train.lda.y ~ duration + cons\_price\_idx + cons\_conf\_idx + euribor3m,  
 data = train.lda.x)  
pred.lda\_step2 <- predict(fit.lda\_step2, newdata = train.lda.x)  
preds\_step2 <- pred.lda\_step2$posterior  
preds\_step2 <- as.data.frame(preds\_step2)  
pred\_step2 <- prediction(preds\_step2[, 2], train.lda.y)  
roc.perf\_step2 = performance(pred\_step2, measure = "tpr", x.measure = "fpr")  
auc.train\_step2 <- performance(pred\_step2, measure = "auc")  
auc.train\_step2 <- auc.train\_step2@y.values  
plot(roc.perf\_step2, colorize = TRUE)  
abline(a = 0, b = 1)  
text(x = .40, y = .6, paste("AUC = ", round(auc.train\_step2[[1]], 3), sep = ""))  
#AUC = 0.912  
#running cv on train set using QDA with subset of numeric vars  
nloops<-50 #number of CV loops  
ntrains<-dim(train.lda.x)[1] #No. of samples in training data set  
cv.aucs\_qda<-c()  
  
set.seed(123)  
  
for (i in 1:nloops) {  
 index <- sample(1:ntrains, ntrains \* .8)  
 cvtrain.x <- train.lda.x[index, ]  
 cvtest.x <- train.lda.x[-index, ]  
 cvtrain.y <- train.lda.y[index]  
 cvtest.y <- train.lda.y[-index]  
   
 cvfit\_qda <-  
 qda(  
 cvtrain.y ~ duration + campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m,  
 data = cvtrain.x  
 )  
 fit.pred\_qda <- predict(cvfit\_qda, newdata = cvtest.x)  
 preds.cv\_qda <- fit.pred\_qda$posterior  
 preds.cv\_qda <- as.data.frame(preds.cv\_qda)  
 pred.cv\_qda <- prediction(preds.cv\_qda[, 2], cvtest.y)  
 roc.perf\_qda = performance(pred.cv\_qda, measure = "tpr", x.measure = "fpr")  
 auc.train\_qda <- performance(pred.cv\_qda, measure = "auc")  
 auc.train\_qda <- auc.train\_qda@y.values  
   
 cv.aucs\_qda[i] <- auc.train\_qda[[1]]  
}  
  
hist(cv.aucs\_qda)  
summary(cv.aucs\_qda)  
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.8845 0.8911 0.8958 0.8955 0.8993 0.9097   
fit.qda <-  
 qda(  
 train.lda.y ~ duration + campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m,  
 data = train.lda.x  
 )  
  
pred.qda <- predict(fit.qda, newdata = train.lda.x)  
preds\_qda <- pred.qda$posterior  
preds\_qda <- as.data.frame(preds\_qda)  
pred\_qda <- prediction(preds\_qda[, 2], train.lda.y)  
roc.perf\_qda = performance(pred\_qda, measure = "tpr", x.measure = "fpr")  
auc.train\_qda <- performance(pred\_qda, measure = "auc")  
auc.train\_qda <- auc.train\_qda@y.values  
plot(roc.perf\_qda, colorize = TRUE)  
abline(a = 0, b = 1)  
text(x = .40, y = .6, paste("AUC = ", round(auc.train\_qda[[1]], 3), sep = ""))  
  
#AUC = 0.896  
  
# Test Set  
pred.qda1 <- predict(fit.qda, newdata = test.lda.x)  
preds1\_qda <- pred.qda1$posterior  
preds1\_qda <- as.data.frame(preds1\_qda)  
pred1\_qda <- prediction(preds1\_qda[, 2], test.lda.y)  
roc.perf\_qda1 = performance(pred1\_qda, measure = "tpr", x.measure = "fpr")  
auc.train\_qda1 <- performance(pred1\_qda, measure = "auc")  
auc.train\_qda1 <- auc.train\_qda1@y.values  
plot(roc.perf\_qda1)  
abline(a = 0, b = 1)  
text(x = .40, y = .6, paste("AUC = ", round(auc.train\_qda1[[1]], 3), sep = ""))  
  
#AUC = 0.892  
#Run randomly shuffled y -vars because the models are performing very similarly  
nloops<-50 #number of CV loops  
ntrains<-dim(train.lda.x)[1] #No. of samples in training data set  
cv.aucs\_shuf<-c()  
dat.train.yshuf<-train.lda.y[sample(1:length(train.lda.y))]  
  
set.seed(123)  
  
for (i in 1:nloops) {  
 index <- sample(1:ntrains, ntrains \* .8)  
 cvtrain.x <- train.lda.x[index, ]  
 cvtest.x <- train.lda.x[-index, ]  
 cvtrain.yshuf <- dat.train.yshuf[index]  
 cvtest.yshuf <- dat.train.yshuf[-index]  
   
 cvfit\_shuf <-  
 lda(  
 cvtrain.yshuf ~ duration + campaign + previous + cons\_price\_idx + cons\_conf\_idx + euribor3m,  
 data = cvtrain.x  
 )  
 fit.pred\_shuf <- predict(cvfit\_shuf, newdata = cvtest.x)  
 preds.cv\_shuf <- fit.pred\_shuf$posterior  
 preds.cv\_shuf <- as.data.frame(preds.cv\_shuf)  
 pred.cv\_shuf <- prediction(preds.cv\_shuf[, 2], cvtest.yshuf)  
 roc.perf\_shuf = performance(pred.cv\_shuf, measure = "tpr", x.measure = "fpr")  
 auc.train\_shuf <- performance(pred.cv\_shuf, measure = "auc")  
 auc.train\_shuf <- auc.train\_shuf@y.values  
   
 cv.aucs\_shuf[i] <- auc.train\_shuf[[1]]  
}  
hist(cv.aucs\_shuf)  
summary(cv.aucs\_shuf)  
 # Min. 1st Qu. Median Mean 3rd Qu. Max.   
 #0.4871 0.5081 0.5127 0.5125 0.5186 0.5299   
cutoff<-0.15  
class.lda\_all<-factor(ifelse(preds1[2]>cutoff,"yes","no"),levels=c("no","yes"))  
class.lda\_step<-factor(ifelse(preds1\_step[2]>cutoff,"yes","no"),levels=c("no","yes"))  
class.qda\_step<-factor(ifelse(preds1\_qda[2]>cutoff,"yes","no"),levels=c("no","yes"))  
#Confusion Matrix for LDA with all vars  
conf.lda\_all<-table(class.lda\_all,test.lda.y)  
print("Confusion matrix for LDA with all Vars")  
conf.lda\_all  
  
#Confusion Matrix for LDA with stepwise vars  
conf.lda\_step<-table(class.lda\_step,test.lda.y)  
print("Confusion matrix for LDA with some Vars")  
conf.lda\_step  
  
#Confusion Matrix for QDA with stepwise vars  
conf.qda\_step<-table(class.qda\_step,test.lda.y)  
print("Confusion matrix for QDA with some Vars")  
conf.qda\_step  
  
#Accuracy of LASSO and Stepwise  
print("Overall accuracy for LDA w/ all vars, LDA w/ some vars, and QDA respectively")  
sum(diag(conf.lda\_all))/sum(conf.lda\_all)  
sum(diag(conf.lda\_all))/sum(conf.lda\_all)  
sum(diag(conf.qda\_step))/sum(conf.qda\_step)  
  
#Confusion Matrix for cut off =0.15  
lda\_all\_0.15<-confusionMatrix(conf.lda\_all)  
lda\_step\_0.15<-confusionMatrix(conf.lda\_step)  
qda\_0.15<-confusionMatrix(conf.qda\_step)  
lda\_all\_0.15  
lda\_step\_0.15  
qda\_0.15  
  
Sensitivity\_LDA <-  
 data.frame(  
 "Model" = c("LDA All", "LDA Stepwise", "QDA Stepwise"),  
 "Sensitivity" = c(  
 lda\_all\_0.15$byClass[1],  
 lda\_step\_0.15$byClass[1],  
 qda\_0.15$byClass[1]  
 )  
 )  
Specificity\_LDA <-  
 data.frame("Specificity" = c(  
 lda\_all\_0.15$byClass[2],  
 lda\_step\_0.15$byClass[2],  
 qda\_0.15$byClass[2]  
 ))  
Accuracy\_LDA <-  
 data.frame("Accuracy" = c(  
 lda\_all\_0.15$overall[1],  
 lda\_step\_0.15$overall[1],  
 qda\_0.15$overall[1]  
 ))  
  
Overall <- cbind(Sensitivity\_LDA, Specificity\_LDA, Accuracy\_LDA)  
Overall  
#train <- read.csv("../data/train.csv", stringsAsFactors = TRUE)  
#test <- read.csv("../data/test.csv", stringsAsFactors = TRUE)  
# set up train2/test2 to explore modeling without duration  
#train2 <- train %>% dplyr::select(c(-duration, -duration\_group))  
#test2 <- test %>% dplyr::select(c(-duration, -duration\_group))  
#train\_orig <- train  
#test\_orig <- test  
#train <- train2  
#test <- test2  
set.seed(1234)  
cv\_control <- trainControl(method="cv",   
 classProbs = TRUE,  
 savePredictions = TRUE,  
 summaryFunction = twoClassSummary,  
 num = 5)  
rf\_grid <- expand.grid(  
 mtry = 4:8,  
 splitrule = c("gini","extratrees", "hellinger"),  
 min.node.size = c(1)  
)  
fitRF <- train(y ~ .,   
 data = train,   
 method = "ranger",   
 metric = "ROC",  
 importance = "impurity",  
 trControl = cv\_control,  
 num.threads = 6,  
 num.trees = 30,  
 tuneGrid=rf\_grid)   
fitRF  
plot(fitRF)  
confusionMatrix(fitRF, positive = "yes")  
library(randomForestExplainer)  
forest\_frame <- min\_depth\_distribution(fitRF$finalModel)  
plot\_min\_depth\_distribution(forest\_frame)  
# !!!DANGER!!! !!!SUPER SLOW!!! !!!LUNCH BREAK/WASH YOUR CAR SLOW!!  
#plot\_min\_depth\_interactions(fitRF$finalModel, k=7)  
multi\_imps = measure\_importance(fitRF$finalModel)  
plot\_importance\_ggpairs(multi\_imps)  
fitRF.predictions.raw <- predict(fitRF, newdata = test, type="raw")  
fitRF.predictions.prob <- predict(fitRF, newdata = test, type="prob")  
confusionMatrix(fitRF.predictions.raw, test$y, positive = "yes")  
prediction.probabilities <- fitRF.predictions.prob$yes  
predicted.classes <- fitRF.predictions.raw  
observed.classes <- test$y  
  
# Compute roc  
res.roc <- roc(observed.classes, prediction.probabilities)  
plot.roc(res.roc, print.auc = TRUE, print.thres = "best")  
# If we wanted cutoffs for specific specificities we specifically specify, we could do THIS:  
#roc.data <- data\_frame(  
# thresholds = res.roc$thresholds,  
# sensitivity = res.roc$sensitivities,  
# specificity = res.roc$specificities  
#)  
# Then we can get the cutoff for specificity = <something> like this  
#roc.data %>% filter(specificity >= 0.6)  
#...or similar  
  
#ROCR - trying to get in same format for overlay below  
pred.rf <- prediction(fitRF.predictions.prob[,2],test$y)  
roc.perf\_rf = performance(pred.rf, measure = "tpr", x.measure = "fpr")  
auc.rf <- performance(pred.rf, measure = "auc")  
auc.rf <- auc.rf@y.values  
plot(roc.perf\_rf)  
abline(a=0, b= 1)  
text(x = .40, y = .6,paste("AUC = ", round(auc.rf[[1]],3), sep = ""))  
library(pROC)  
prediction.probabilities <- fitRF.predictions.prob$yes  
predicted.classes <- fitRF.predictions.raw  
observed.classes <- test$y  
  
# Compute roc  
roc.randomforest <- roc(observed.classes, prediction.probabilities)  
plot.roc(roc.randomforest, print.auc = TRUE, print.thres = "best", col="purple")  
# Get the best cutoff for balancing Sensitivity and Specificity  
cutoff <- coords(roc.randomforest, "best", ret="threshold", transpose = FALSE)$threshold  
  
# Predict using the best cutoff and confirm with a Confusion Matrix  
predicted.classes.balanced <- factor(  
 ifelse( fitRF.predictions.prob$yes > cutoff, "yes", "no"), levels=c("no","yes"))  
confusionMatrix(predicted.classes.balanced, test$y, positive="yes")  
# If exploring modeling without duration, restore the original train/test for use by any code below that might rely on it  
#train <- train\_orig  
#test <- test\_orig  
#graphics.off()  
  
#add ROC curve for our top simple model, complex model, LDA, and RF  
plot(roc.step,col="orange")  
plot(roc.complex,col = "blue", add = TRUE)  
plot(roc.perf\_step2, col="red", add = TRUE)  
plot(roc.perf\_rf, col = "green", add = TRUE)  
legend("bottomright",legend=c("Stepwise Logistic Regression","Complex Model", "LDA", "Random Forest"),col=c("orange","blue","red","green"),lty=1,lwd=1)  
abline(a=0, b= 1)