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

The dataset used in our analysis initially consisted of 41,188 observations with a total of 21 different variables.

# Exploratory Data Analysis

Walkthrough of key findings in EDA

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

# Objective 1

## Restatement of Problem

Restatement of Problem and the overall approach to solve

Modeling Strategy

Upon doing EDA and creating a few derived variables, we started with creating a simple logistic regression 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. We kept poutcome and prevly\_cntctd in the simple model because even though these had comparatively higher VIFs, these are practically independent columns. After every column removal, we reran the model and repeated the exercise till we found the model with statistically significant as well moderate VIFs variables.

We then ran feature selection methods– Step and LASSO, doing the same exercise as above. Using ROC curves to find a good cut off range, we predicted the response variable with few cut off values in the range 0.1 and 0.5. We then compared the Accuracy, Sensitivity and Specificity of all the different models – simple, step, lasso on various cut offs. Considering the dataset and the goal, we chose better specificity over better accuracy and sensitivity.

Model

For simple interpretable model, the following model gave us the best confusion matrix statistics at cut off=0.15.

Variables considered significant for the model to predict the odds of a client subscribing to a term deposit as part of the marketing campaign - education, default, month, duration ,campaign, poutcome, cons\_price\_idx, euribor3m and Age\_Grp.

Log(odds of a client subscribing to a term deposit) = -31.2267 + (-0.3241)\*education\_ basic.4y + (-0.1680)\*education\_ basic.6y + (-0.3459)\*education\_ basic.9y + (-0.2058)\*education\_ high.school + (1.1845)\*education\_ illiterate + (1)education\_university.degree

+ (2.4913)\*default\_no + (2.1216)\* default\_unknown + (1)\* default\_yes

+ (1.1850)\* month\_mar (-0.6265)\* month\_apr + (-1.1457)\* month\_may + (-0.00265)\* month\_jun + (0.1381)\*month\_jul + (0.2490)\* month\_aug + (0.3729)\* month\_oct + (-0.2051)\* month\_nov + (0.0619)\* month\_dec + (1)\* month\_sep

+ (-0.7414)\* poutcome\_failure + (-0.3325)\* poutcome\_nonexistent + (1)\* poutcome\_success

+ (0.1205)\* Age\_Grp\_17\_31 + (-0.1167)\* Age\_Grp\_32\_37 + (-0.1933)\* Age\_Grp\_38\_47 + (-0.0187)\* Age\_Grp\_47\_55 + (1)\* Age\_Grp\_>55 + (0.00473)\* duration + (-0.0534)\* Campaign + (0.3002)\* cons\_price\_idx + (-0.7190)\* euribor3m

## Assumptions

Data points are independent.

Lack of fit test:

* We performed Global test for beta=0, with Likelihood ratio, Score and Wald test all agreeing at p-value<0.001, we reject the null hypothesis and conclude to say that the model is valid.
* Confusion matrix for accuracy, sensitivity and specificity: Add tables. Include tables for both train and test.

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. Please see Appendix sectionXYZ for details on outliers, model output and plots after removing the outliers from the training data set.

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

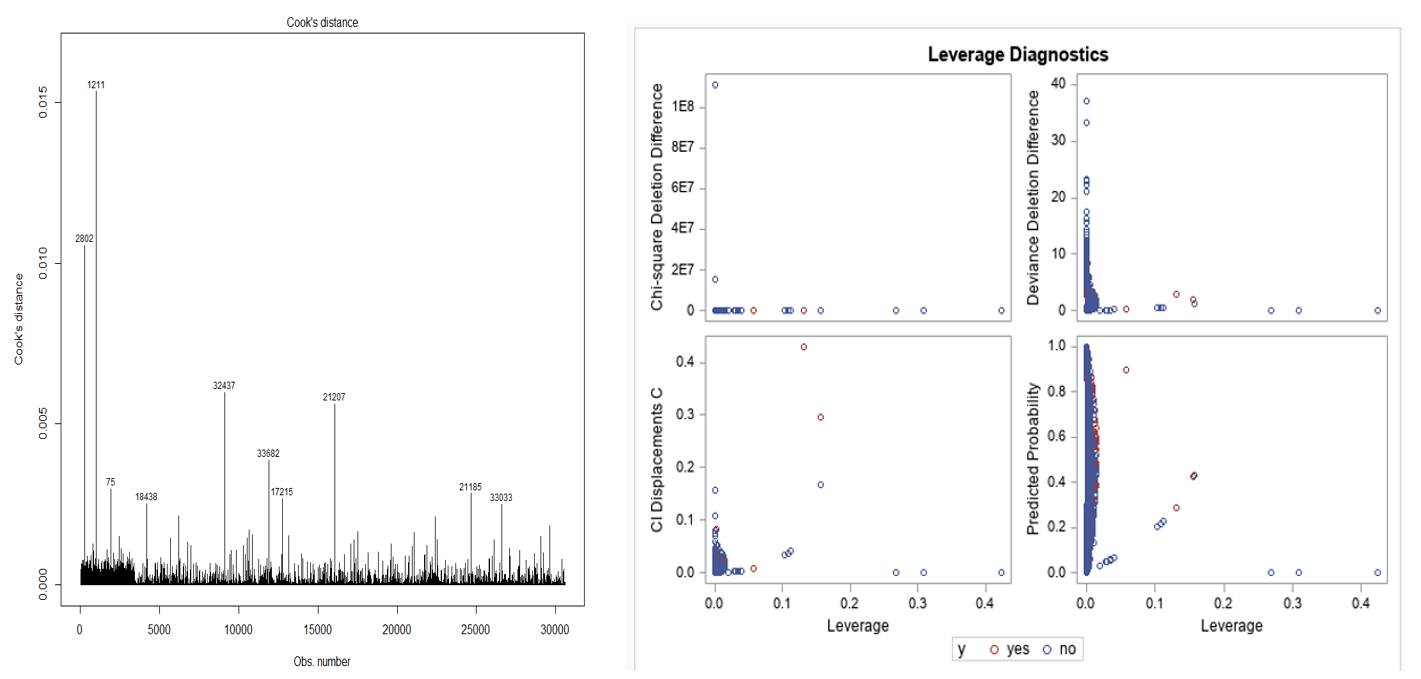


Figure 1 Residual Plots from the model with full training data set.

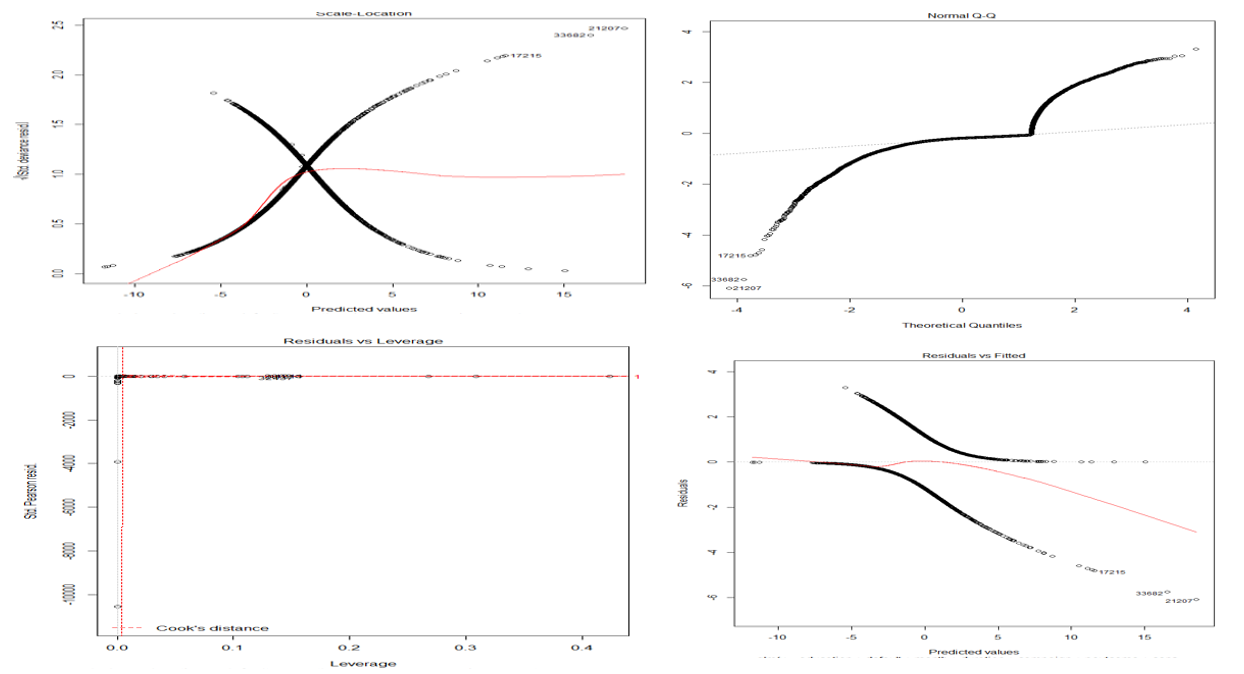


Figure 2

## Model Interpretation

Education:

Holding all other variables constant, the odds of a client (saying yes to a term deposit) with a basic.4y education are estimated to be 0.738 times lower than those of a client with a university.degree. The estimated 95% confidence interval of these odds is between 0.616 and 0.884.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) with a basic.6y education are 0.863 times lower than those of a client with a university.degree. The 95% confidence interval of these odds is between 0.687 and 1.083.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) with a basic.9y education are 0.722 times lower than those of a client with a university.degree. The 95% confidence interval of these odds is between 0.619 and 0.843.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) with a basic. high.school education are 0.831 times lower than those of a client with a university.degree. The 95% confidence interval of these odds is between 0.736 and 0.938.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who is an illiterate are 3.337 times higher than those of a client with a university.degree. The 95% confidence interval of these odds is between 0.700 and 15.903.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) with a professional.courseeducation are 0.905 times lower than those of a client with a university.degree. The 95% confidence interval of these odds is between 0.782 and 1.048.

Default:

Default no

Default unknown

Month:

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Mar are 3.360 times higher than those of a client last contacted in September. The 95% confidence interval of these odds is between 2.425 and 4.654.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Apr are 0.549 times lower than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.418 and 0.721.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of May are 0.327 times lower than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.252 and 0.424.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Jun are 1.024 times higher than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.777 and 1.350.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Jul are 1.179 times higher than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.890 and 1.563.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Aug are 1.318 times higher than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.997 and 1.742.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Oct are 1.491 times higher than those of a client last contacted in September. The 95% confidence interval of these odds is between 1.091 and 2.040.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Nov are 0.837 times lower than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.626 and 1.118.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) who was last contacted in the month of Dec are 1.093 times higher than those of a client last contacted in September. The 95% confidence interval of these odds is between 0.668 and 1.789.

Poutcome:

Holding all other variables constant, the odds of a client (saying yes to a term deposit) whose last campaign outcome was a failure are 0.163 times lower than those whose last campaign outcome was a success. The 95% confidence interval of these odds is between 0.134 and 0.197.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) whose last campaign outcome is nonexistent are 0.245 times lower than those whose last campaign outcome was a success. The 95% confidence interval of these odds is between 0.207 and 0.290.

Age\_Grp:

Holding all other variables constant, the odds of a client (saying yes to a term deposit) in the age group 17-31 are 0.916 times lower than those who are >55yrs in age. The 95% confidence interval of these odds is between 0.775 and 1.082.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) in the age group 32-37are 0.723 times lower than those who are >55yrs in age. The 95% confidence interval of these odds is between 0.610 and 0.856.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) in the age group 38-47 are 0.669 times lower than those who are >55yrs in age. The 95% confidence interval of these odds is between 0.562 and 0.796.

Holding all other variables constant, the odds of a client (saying yes to a term deposit) in the age group 47-55 are 0.797 times lower than those who are >55yrs in age. The 95% confidence interval of these odds is between 0.664 and 0.957.

Duration: The odds ratio is 1.005 which means that for every single unit increase in the call duration, the odds a client subscribing to a term deposit increase by 0.5%. The 95% confident interval for the odds is between 1.005 and 1.005.

Campaign: The odds ratio is 0.948 which means that for every single unit increase in the number of contacts, the odds a client subscribing to a term deposit decreases by 0.52%. The 95% confident interval for the odds is between 0.923 and 0.973.

cons\_price\_idx: The odds ratio is 1.350 which means that for every single unit increase in the cons\_price\_idx, the odds a client subscribing to a term deposit increase by 35%. The 95% confident interval for the odds is between 1.229 and 1.483.

euribor3m: The odds ratio is 0.487 which means that for every single unit increase in euribor3m, the odds a client subscribing to a term deposit decreases by 0.513%. The 95% confident interval for the odds is between 0.468 and 0.507.

# Objective 2:

For competing models, we chose Complex Model, LDA/QDA, and Random Forest.

Complex Model

TODO: Brief summary of analysis and tuning parameters

LDA/QDA

TODO: Brief summary of analysis and tuning parameters

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.

Chart, line chart

Description automatically generated

Optimizing for ROC, the winning parameters are an mtry of 5 predictors considered at each split, and the Hellinger split rule. 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.

Chart, bar chart

Description automatically generated

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.

|  |
| --- |
| **Aside: Modeling without duration**  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. 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, we wanted to call out this finding and possible next steps. |

Getting back to our original model, another way to interpret the model is to look at the influence of interactions:

Chart, bar chart, waterfall chart

Description automatically generated

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.

**Random Forest Performance**

The performance of our random forest model on the holdout test set with default cutoff is:

Accuracy : 0.9176

Kappa : 0.5315

Sensitivity : 0.50235

Specificity : 0.96969

Our model is 91.7% Accurate, but much more Specific than it is Sensitive.

Using an ROC curve, we can find a cutoff that gets more balanced results between sensitivity and specificity. Our ROC curve gives a “best” cutoff of 0.135, which when applied to our prediction probabilities gives more balanced, but less accurate results:

Accuracy : 0.8558

Kappa : 0.5145

Sensitivity : 0.9249

Specificity : 0.8471

## Metrics & Model Comparison

Table of accuracy, sensitivity, specificity for each model

Chart

Description automatically generated

Figure 3

# 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.

# Appendix

Figures referred to in report

## R Code

knitr::opts\_chunk$set(echo = TRUE)

*#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(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, fill = 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"))

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%*

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")) %>% 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")

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)

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.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

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

*#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.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",

trControl = cv\_control,

num.threads = 6,

num.trees = 30,

tuneGrid=rf\_grid)

fitRF

plot(fitRF)

confusionMatrix(fitRF, positive = "yes")

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")

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)

*#plot(roc.randomforest, col="purple", 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)