Project2

Rinku Lichti, Simerpreet Reddy, Megan Ball

12/5/2020

# Load Data

#full <- read\_delim(here::here("data", "bank-additional-full.csv"),';')  
full <- read.csv(file.choose(), sep=';')  
str(full)

## 'data.frame': 41188 obs. of 21 variables:  
## $ age : int 56 57 37 40 56 45 59 41 24 25 ...  
## $ job : chr "housemaid" "services" "services" "admin." ...  
## $ marital : chr "married" "married" "married" "married" ...  
## $ education : chr "basic.4y" "high.school" "high.school" "basic.6y" ...  
## $ default : chr "no" "unknown" "no" "no" ...  
## $ housing : chr "no" "no" "yes" "no" ...  
## $ loan : chr "no" "no" "no" "no" ...  
## $ contact : chr "telephone" "telephone" "telephone" "telephone" ...  
## $ month : chr "may" "may" "may" "may" ...  
## $ day\_of\_week : chr "mon" "mon" "mon" "mon" ...  
## $ duration : int 261 149 226 151 307 198 139 217 380 50 ...  
## $ campaign : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ pdays : int 999 999 999 999 999 999 999 999 999 999 ...  
## $ previous : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ poutcome : chr "nonexistent" "nonexistent" "nonexistent" "nonexistent" ...  
## $ emp.var.rate : num 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...  
## $ cons.price.idx: num 94 94 94 94 94 ...  
## $ cons.conf.idx : num -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...  
## $ euribor3m : num 4.86 4.86 4.86 4.86 4.86 ...  
## $ nr.employed : num 5191 5191 5191 5191 5191 ...  
## $ y : chr "no" "no" "no" "no" ...

head(full)

## age job marital education default housing loan contact month  
## 1 56 housemaid married basic.4y no no no telephone may  
## 2 57 services married high.school unknown no no telephone may  
## 3 37 services married high.school no yes no telephone may  
## 4 40 admin. married basic.6y no no no telephone may  
## 5 56 services married high.school no no yes telephone may  
## 6 45 services married basic.9y unknown no no telephone may  
## day\_of\_week duration campaign pdays previous poutcome emp.var.rate  
## 1 mon 261 1 999 0 nonexistent 1.1  
## 2 mon 149 1 999 0 nonexistent 1.1  
## 3 mon 226 1 999 0 nonexistent 1.1  
## 4 mon 151 1 999 0 nonexistent 1.1  
## 5 mon 307 1 999 0 nonexistent 1.1  
## 6 mon 198 1 999 0 nonexistent 1.1  
## cons.price.idx cons.conf.idx euribor3m nr.employed y  
## 1 93.994 -36.4 4.857 5191 no  
## 2 93.994 -36.4 4.857 5191 no  
## 3 93.994 -36.4 4.857 5191 no  
## 4 93.994 -36.4 4.857 5191 no  
## 5 93.994 -36.4 4.857 5191 no  
## 6 93.994 -36.4 4.857 5191 no

nrow(full)

## [1] 41188

ncol(full)

## [1] 21

# Clean up column names  
full <- janitor::clean\_names(full)  
summary(full)

## age job marital education   
## Min. :17.00 Length:41188 Length:41188 Length:41188   
## 1st Qu.:32.00 Class :character Class :character Class :character   
## Median :38.00 Mode :character Mode :character Mode :character   
## Mean :40.02   
## 3rd Qu.:47.00   
## Max. :98.00   
## default housing loan contact   
## Length:41188 Length:41188 Length:41188 Length:41188   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## month day\_of\_week duration campaign   
## Length:41188 Length:41188 Min. : 0.0 Min. : 1.000   
## Class :character Class :character 1st Qu.: 102.0 1st Qu.: 1.000   
## Mode :character Mode :character Median : 180.0 Median : 2.000   
## Mean : 258.3 Mean : 2.568   
## 3rd Qu.: 319.0 3rd Qu.: 3.000   
## Max. :4918.0 Max. :56.000   
## pdays previous poutcome emp\_var\_rate   
## Min. : 0.0 Min. :0.000 Length:41188 Min. :-3.40000   
## 1st Qu.:999.0 1st Qu.:0.000 Class :character 1st Qu.:-1.80000   
## Median :999.0 Median :0.000 Mode :character Median : 1.10000   
## Mean :962.5 Mean :0.173 Mean : 0.08189   
## 3rd Qu.:999.0 3rd Qu.:0.000 3rd Qu.: 1.40000   
## Max. :999.0 Max. :7.000 Max. : 1.40000   
## cons\_price\_idx cons\_conf\_idx euribor3m nr\_employed   
## Min. :92.20 Min. :-50.8 Min. :0.634 Min. :4964   
## 1st Qu.:93.08 1st Qu.:-42.7 1st Qu.:1.344 1st Qu.:5099   
## Median :93.75 Median :-41.8 Median :4.857 Median :5191   
## Mean :93.58 Mean :-40.5 Mean :3.621 Mean :5167   
## 3rd Qu.:93.99 3rd Qu.:-36.4 3rd Qu.:4.961 3rd Qu.:5228   
## Max. :94.77 Max. :-26.9 Max. :5.045 Max. :5228   
## y   
## Length:41188   
## Class :character   
## Mode :character   
##   
##   
##

#print(dfSummary(full, graph.magnif = 0.75), method = 'browser')  
str(full)

## 'data.frame': 41188 obs. of 21 variables:  
## $ age : int 56 57 37 40 56 45 59 41 24 25 ...  
## $ job : chr "housemaid" "services" "services" "admin." ...  
## $ marital : chr "married" "married" "married" "married" ...  
## $ education : chr "basic.4y" "high.school" "high.school" "basic.6y" ...  
## $ default : chr "no" "unknown" "no" "no" ...  
## $ housing : chr "no" "no" "yes" "no" ...  
## $ loan : chr "no" "no" "no" "no" ...  
## $ contact : chr "telephone" "telephone" "telephone" "telephone" ...  
## $ month : chr "may" "may" "may" "may" ...  
## $ day\_of\_week : chr "mon" "mon" "mon" "mon" ...  
## $ duration : int 261 149 226 151 307 198 139 217 380 50 ...  
## $ campaign : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ pdays : int 999 999 999 999 999 999 999 999 999 999 ...  
## $ previous : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ poutcome : chr "nonexistent" "nonexistent" "nonexistent" "nonexistent" ...  
## $ emp\_var\_rate : num 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...  
## $ cons\_price\_idx: num 94 94 94 94 94 ...  
## $ cons\_conf\_idx : num -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...  
## $ euribor3m : num 4.86 4.86 4.86 4.86 4.86 ...  
## $ nr\_employed : num 5191 5191 5191 5191 5191 ...  
## $ y : chr "no" "no" "no" "no" ...

# Check for missing values  
tibble(variable = names(colSums(is.na(full))),  
 missing = colSums(is.na(full))) %>%   
 gt() %>%   
 tab\_header(title = "Missing Values in Data")

Missing Values in Data

variable

missing

age

0

job

0

marital

0

education

0

default

0

housing

0

loan

0

contact

0

month

0

day\_of\_week

0

duration

0

campaign

0

pdays

0

previous

0

poutcome

0

emp\_var\_rate

0

cons\_price\_idx

0

cons\_conf\_idx

0

euribor3m

0

nr\_employed

0

y

0

Looking at the dfsummary, there doesn’t seem to be missing data in terms of just not having values. However, there are some fields that have explicit unknown or non-existent classes that could be considered as ‘missing’. For example, loan and housing have 990 “unknown” values. And ‘default’ has 8597 “unknown” values representing 20.9%

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

# Remove missing values

#remove "unknowns" based on small sample sizes compared to full data set  
df <- full %>% filter(loan != "unknown")  
nrow(df)

## [1] 40198

#down to 40,198 obs  
df <- df %>% filter(marital != "unknown")  
nrow(df)

## [1] 40119

#down to 40,119 obs  
df <- df %>% filter(education != "unknown")  
nrow(df)

## [1] 38437

#down to 38,437 obs  
#remove unknowns from job  
df <- df %>% filter(job != "unknown")  
nrow(df)

## [1] 38245

#down to 38,245 obs  
#remove yes from default - only 3, and all 3 are "no"  
df <- df %>% filter(default != "yes")  
nrow(df)

## [1] 38242

#down to 38,242 obs  
str(df)

## 'data.frame': 38242 obs. of 21 variables:  
## $ age : int 56 57 37 40 56 45 59 24 25 25 ...  
## $ job : chr "housemaid" "services" "services" "admin." ...  
## $ marital : chr "married" "married" "married" "married" ...  
## $ education : chr "basic.4y" "high.school" "high.school" "basic.6y" ...  
## $ default : chr "no" "unknown" "no" "no" ...  
## $ housing : chr "no" "no" "yes" "no" ...  
## $ loan : chr "no" "no" "no" "no" ...  
## $ contact : chr "telephone" "telephone" "telephone" "telephone" ...  
## $ month : chr "may" "may" "may" "may" ...  
## $ day\_of\_week : chr "mon" "mon" "mon" "mon" ...  
## $ duration : int 261 149 226 151 307 198 139 380 50 222 ...  
## $ campaign : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ pdays : int 999 999 999 999 999 999 999 999 999 999 ...  
## $ previous : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ poutcome : chr "nonexistent" "nonexistent" "nonexistent" "nonexistent" ...  
## $ emp\_var\_rate : num 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...  
## $ cons\_price\_idx: num 94 94 94 94 94 ...  
## $ cons\_conf\_idx : num -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...  
## $ euribor3m : num 4.86 4.86 4.86 4.86 4.86 ...  
## $ nr\_employed : num 5191 5191 5191 5191 5191 ...  
## $ y : chr "no" "no" "no" "no" ...

#recheck summary  
summary(df)

## age job marital education   
## Min. :17.00 Length:38242 Length:38242 Length:38242   
## 1st Qu.:32.00 Class :character Class :character Class :character   
## Median :38.00 Mode :character Mode :character Mode :character   
## Mean :39.86   
## 3rd Qu.:47.00   
## Max. :98.00   
## default housing loan contact   
## Length:38242 Length:38242 Length:38242 Length:38242   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## month day\_of\_week duration campaign   
## Length:38242 Length:38242 Min. : 0.0 Min. : 1.000   
## Class :character Class :character 1st Qu.: 102.0 1st Qu.: 1.000   
## Mode :character Mode :character Median : 180.0 Median : 2.000   
## Mean : 258.2 Mean : 2.567   
## 3rd Qu.: 319.0 3rd Qu.: 3.000   
## Max. :4918.0 Max. :43.000   
## pdays previous poutcome emp\_var\_rate   
## Min. : 0.0 Min. :0.00 Length:38242 Min. :-3.4000   
## 1st Qu.:999.0 1st Qu.:0.00 Class :character 1st Qu.:-1.8000   
## Median :999.0 Median :0.00 Mode :character Median : 1.1000   
## Mean :963.5 Mean :0.17 Mean : 0.0828   
## 3rd Qu.:999.0 3rd Qu.:0.00 3rd Qu.: 1.4000   
## Max. :999.0 Max. :7.00 Max. : 1.4000   
## cons\_price\_idx cons\_conf\_idx euribor3m nr\_employed   
## Min. :92.20 Min. :-50.80 Min. :0.634 Min. :4964   
## 1st Qu.:93.08 1st Qu.:-42.70 1st Qu.:1.344 1st Qu.:5099   
## Median :93.44 Median :-41.80 Median :4.857 Median :5191   
## Mean :93.57 Mean :-40.54 Mean :3.623 Mean :5167   
## 3rd Qu.:93.99 3rd Qu.:-36.40 3rd Qu.:4.961 3rd Qu.:5228   
## Max. :94.77 Max. :-26.90 Max. :5.045 Max. :5228   
## y   
## Length:38242   
## Class :character   
## Mode :character   
##   
##   
##

summary(df)

## age job marital education   
## Min. :17.00 Length:38242 Length:38242 Length:38242   
## 1st Qu.:32.00 Class :character Class :character Class :character   
## Median :38.00 Mode :character Mode :character Mode :character   
## Mean :39.86   
## 3rd Qu.:47.00   
## Max. :98.00   
## default housing loan contact   
## Length:38242 Length:38242 Length:38242 Length:38242   
## Class :character Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character Mode :character   
##   
##   
##   
## month day\_of\_week duration campaign   
## Length:38242 Length:38242 Min. : 0.0 Min. : 1.000   
## Class :character Class :character 1st Qu.: 102.0 1st Qu.: 1.000   
## Mode :character Mode :character Median : 180.0 Median : 2.000   
## Mean : 258.2 Mean : 2.567   
## 3rd Qu.: 319.0 3rd Qu.: 3.000   
## Max. :4918.0 Max. :43.000   
## pdays previous poutcome emp\_var\_rate   
## Min. : 0.0 Min. :0.00 Length:38242 Min. :-3.4000   
## 1st Qu.:999.0 1st Qu.:0.00 Class :character 1st Qu.:-1.8000   
## Median :999.0 Median :0.00 Mode :character Median : 1.1000   
## Mean :963.5 Mean :0.17 Mean : 0.0828   
## 3rd Qu.:999.0 3rd Qu.:0.00 3rd Qu.: 1.4000   
## Max. :999.0 Max. :7.00 Max. : 1.4000   
## cons\_price\_idx cons\_conf\_idx euribor3m nr\_employed   
## Min. :92.20 Min. :-50.80 Min. :0.634 Min. :4964   
## 1st Qu.:93.08 1st Qu.:-42.70 1st Qu.:1.344 1st Qu.:5099   
## Median :93.44 Median :-41.80 Median :4.857 Median :5191   
## Mean :93.57 Mean :-40.54 Mean :3.623 Mean :5167   
## 3rd Qu.:93.99 3rd Qu.:-36.40 3rd Qu.:4.961 3rd Qu.:5228   
## Max. :94.77 Max. :-26.90 Max. :5.045 Max. :5228   
## y   
## Length:38242   
## Class :character   
## Mode :character   
##   
##   
##

Our yes group has decreased by about ~400 to 4,258.

#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)

## 'data.frame': 38242 obs. of 21 variables:  
## $ age : int 56 57 37 40 56 45 59 24 25 25 ...  
## $ job : Factor w/ 11 levels "admin.","blue-collar",..: 4 8 8 1 8 8 1 10 8 8 ...  
## $ marital : Factor w/ 3 levels "divorced","married",..: 2 2 2 2 2 2 2 3 3 3 ...  
## $ education : Factor w/ 7 levels "basic.4y","basic.6y",..: 1 4 4 2 4 3 6 6 4 4 ...  
## $ default : Factor w/ 2 levels "no","unknown": 1 2 1 1 1 2 1 1 1 1 ...  
## $ housing : Factor w/ 2 levels "no","yes": 1 1 2 1 1 1 1 2 2 2 ...  
## $ loan : Factor w/ 2 levels "no","yes": 1 1 1 1 2 1 1 1 1 1 ...  
## $ contact : Factor w/ 2 levels "cellular","telephone": 2 2 2 2 2 2 2 2 2 2 ...  
## $ month : Factor w/ 10 levels "apr","aug","dec",..: 7 7 7 7 7 7 7 7 7 7 ...  
## $ day\_of\_week : Factor w/ 5 levels "fri","mon","thu",..: 2 2 2 2 2 2 2 2 2 2 ...  
## $ duration : int 261 149 226 151 307 198 139 380 50 222 ...  
## $ campaign : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ pdays : int 999 999 999 999 999 999 999 999 999 999 ...  
## $ previous : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ poutcome : Factor w/ 3 levels "failure","nonexistent",..: 2 2 2 2 2 2 2 2 2 2 ...  
## $ emp\_var\_rate : num 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 1.1 ...  
## $ cons\_price\_idx: num 94 94 94 94 94 ...  
## $ cons\_conf\_idx : num -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 -36.4 ...  
## $ euribor3m : num 4.86 4.86 4.86 4.86 4.86 ...  
## $ nr\_employed : num 5191 5191 5191 5191 5191 ...  
## $ y : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

#make sure "success" level is defined as "yes"  
str(df$y)

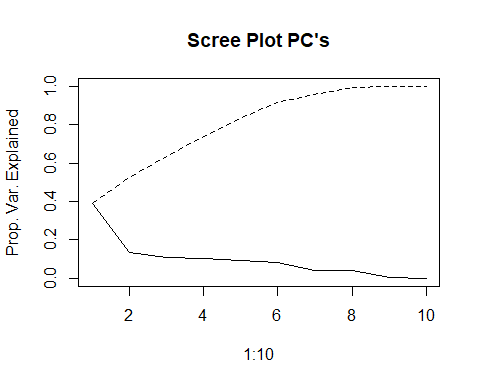
## Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...

# Exploratory Data Analysis

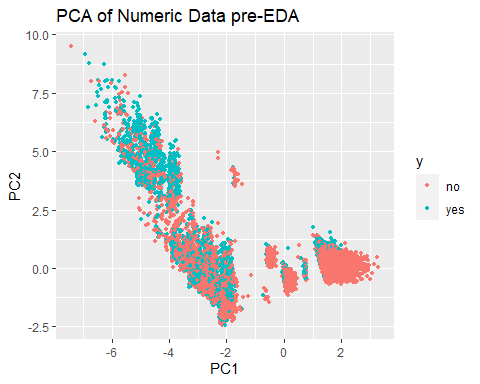
#run first pass PCA to see if we have useful numeric predictors  
df.numeric <- df[ , sapply(df, is.numeric)]  
pc.result<-prcomp(df.numeric,scale.=TRUE)  
pc.scores<-pc.result$x  
pc.scores<-data.frame(pc.scores)  
pc.scores$y<-df$y  
  
#pc.scores  
  
#Scree plot  
eigenvals<-(pc.result$sdev)^2  
eigenvals

## [1] 3.89761881 1.35967518 1.06973450 1.04945508 0.93221207 0.83728417  
## [7] 0.42764325 0.39087410 0.02482374 0.01067909

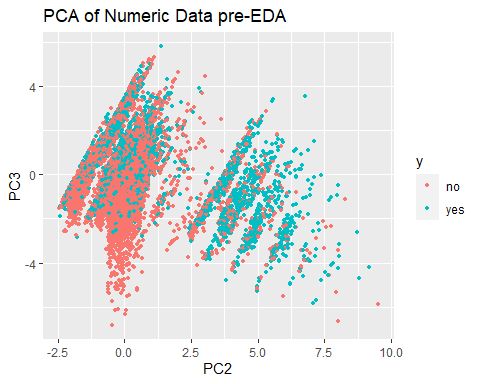
plot(1:10,eigenvals/sum(eigenvals),type="l",main="Scree Plot PC's",ylab="Prop. Var. Explained",ylim=c(0,1))  
cumulative.prop<-cumsum(eigenvals/sum(eigenvals))  
lines(1:10,cumulative.prop,lty=2)



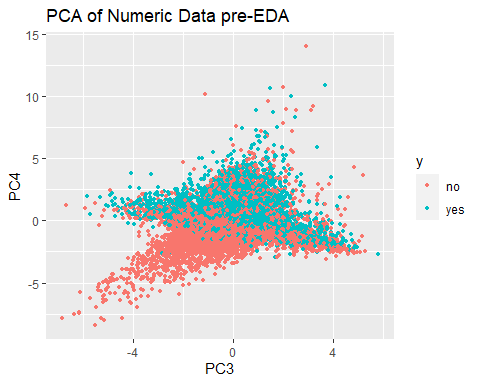
#Use ggplot2 to plot the first few pc's  
ggplot(data = pc.scores, aes(x = PC1, y = PC2)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



#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")

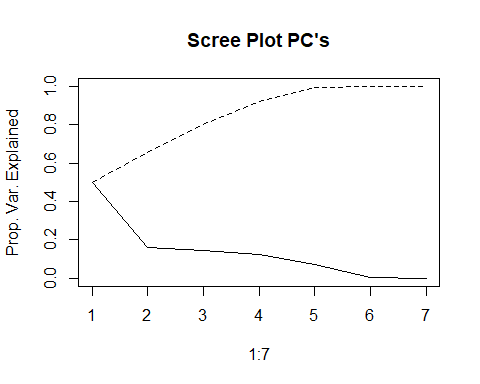


PCA without pdays. campaign, and previous as they are more like factors and not continuous

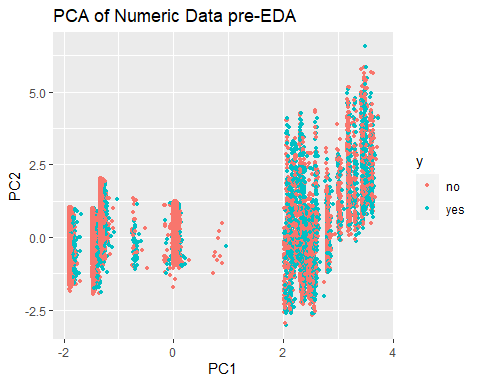
df.numeric2 <- df.numeric %>% dplyr::select(-c(pdays, campaign, previous))  
pc.result2<-prcomp(df.numeric2,scale.=TRUE)  
pc.scores2<-pc.result2$x  
pc.scores2<-data.frame(pc.scores2)  
pc.scores2$y<-df$y  
#pc.scores2  
  
#Scree plot  
eigenvals2<-(pc.result2$sdev)^2  
eigenvals2

## [1] 3.48209924 1.11577551 1.00157280 0.86043468 0.50429117 0.02505300 0.01077359

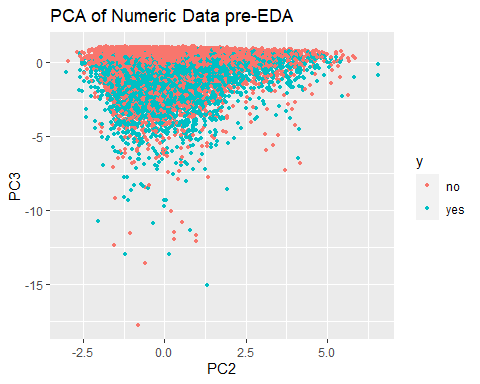
plot(1:7,eigenvals2/sum(eigenvals2),type="l",main="Scree Plot PC's",ylab="Prop. Var. Explained",ylim=c(0,1))  
cumulative.prop2<-cumsum(eigenvals2/sum(eigenvals2))  
lines(1:7,cumulative.prop2,lty=2)



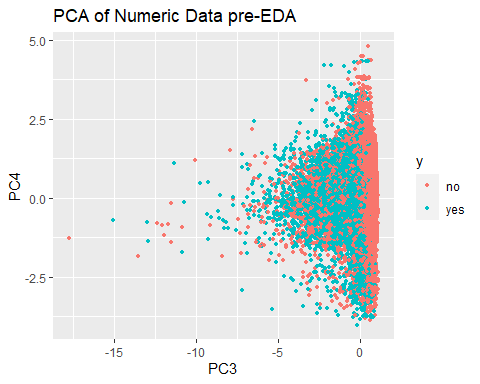
#Use ggplot2 to plot the first few pc's  
ggplot(data = pc.scores2, aes(x = PC1, y = PC2)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



#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")

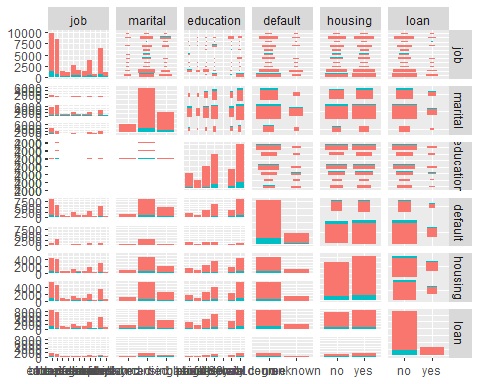


ggplot(data = pc.scores2, aes(x = PC3, y = PC4)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Numeric Data pre-EDA")



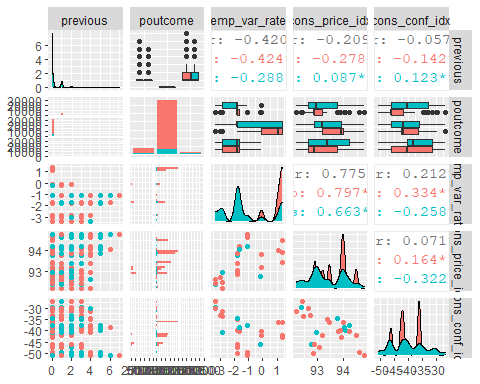
Still does not seem of much value.

#ggpairs(df,columns=1:18, aes(colour=y))  
ggpairs(df,columns=2:7, aes(colour=y))

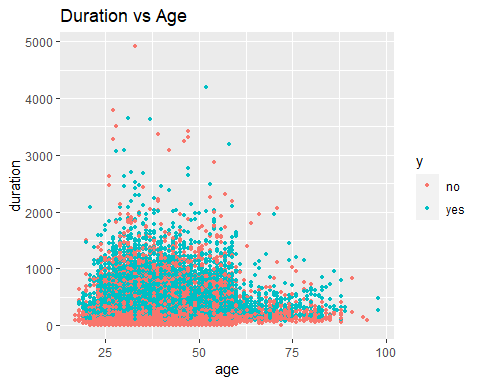


ggpairs(df, columns=14:18, aes(colour=y))

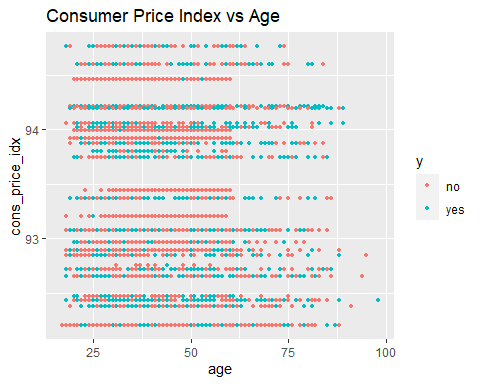
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



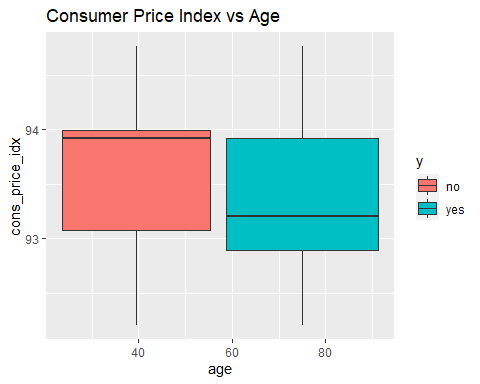
df\_yes <- df %>% filter(y=="yes")  
  
#summary(df\_yes)  
# Nothing interesting found in the below code so commenting it out  
# ggplot(bank\_additional\_full, aes(x=age, y=emp.var.rate)) +  
# geom\_point(size=1, shape="circle") +  
# ggtitle("Employment Variation Rate vs Age") +   
# facet\_wrap(~ y)  
  
ggplot(df, aes(x=age, y=duration, color = y)) + geom\_point(size=1, shape="circle") + ggtitle("Duration vs Age")

 Duration vs Age: The duration of last contact (in seconds) was longer for ages 25-50. And it was understandably longer for ‘yes’’ vs for ‘no’.

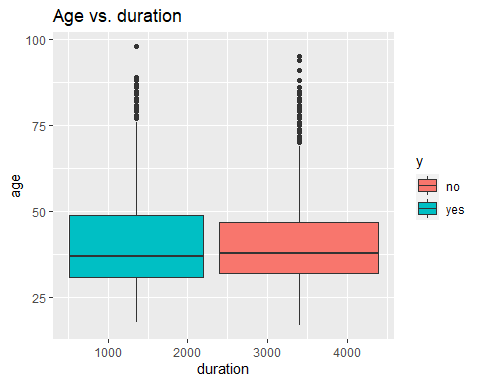
ggplot(df, aes(x = age, y = cons\_price\_idx, color = y)) +   
 geom\_point(size = 1, shape = "circle") +   
 ggtitle("Consumer Price Index vs Age")



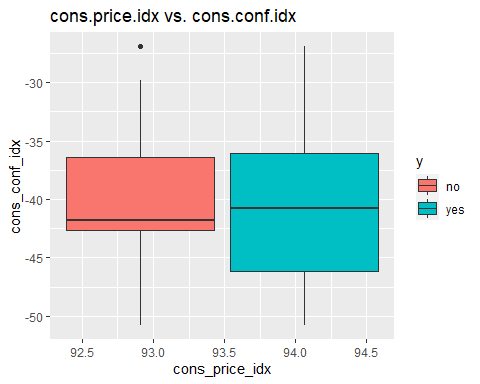
#Checking collinearlity using box plots  
ggplot(df, aes(x = age, y = cons\_price\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Consumer Price Index vs Age")



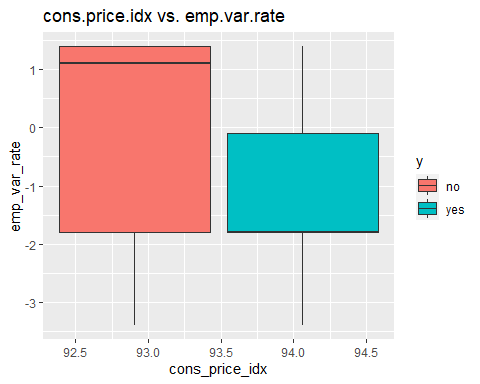
ggplot(df, aes(x = duration , y = age, fill = y)) +   
 geom\_boxplot() + ggtitle("Age vs. duration")



ggplot(df, aes(x = cons\_price\_idx , y = cons\_conf\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("cons.price.idx vs. cons.conf.idx")

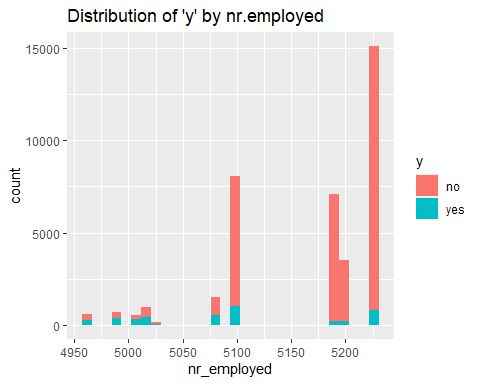


ggplot(df, aes(x = cons\_price\_idx , y = emp\_var\_rate, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("cons.price.idx vs. emp.var.rate")



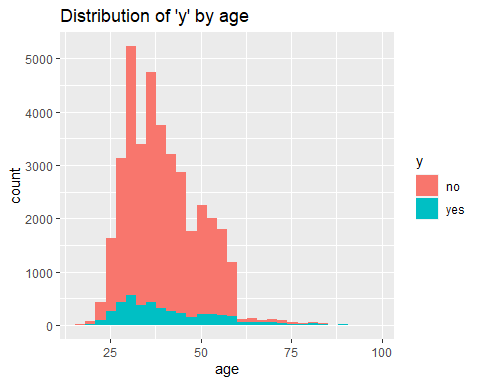
ggplot(df) + geom\_histogram(mapping = aes(x = nr\_employed, fill = y)) +  
 ggtitle("Distribution of 'y' by nr.employed")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



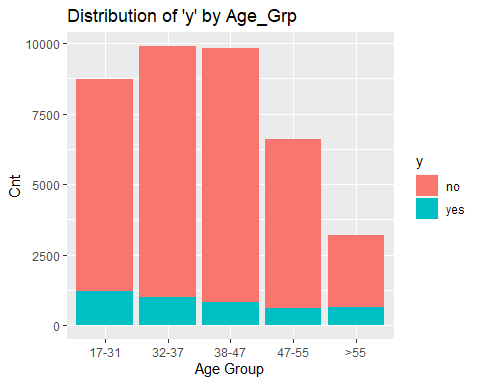
# ggplot(bank\_additional\_full, aes(x=age, y=education)) +  
# geom\_point(size=1, shape="circle") +  
# ggtitle("Education vs Age") +  
# facet\_wrap(~ y)  
ggplot(df)+   
 geom\_histogram(mapping = aes(x = age, fill = y)) +   
 ggtitle("Distribution of 'y' by age")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

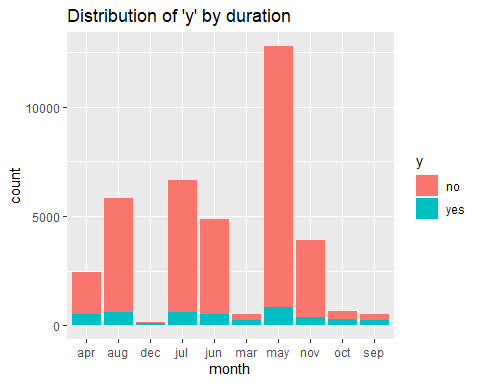


## Creating new variables

#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\_bar(mapping = aes(x = month, fill = y)) +  
 ggtitle("Distribution of 'y' by duration")

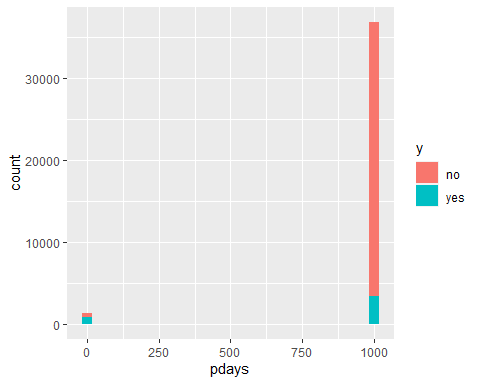


We will keep both age and age group in our model to see if one is selected over the other. We need to make sure to not use both in our model building.

### Analyzing pdays

ggplot(df) +   
 geom\_histogram(mapping = aes(x=pdays, fill=y))

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

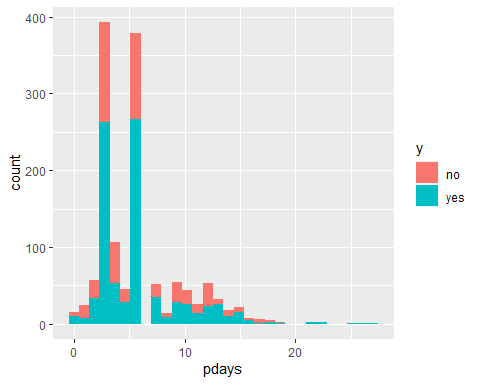


Analyzing ‘pdays’ ie., number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

Most folks had no previous campaign but if they did, it looks like most who had a previous campaign decided to subscribe

#zoom in for ones that were previously contacted  
df %>%   
 filter(pdays < 999) %>%   
 ggplot() +   
 geom\_histogram(mapping = aes(x=pdays, fill=y))

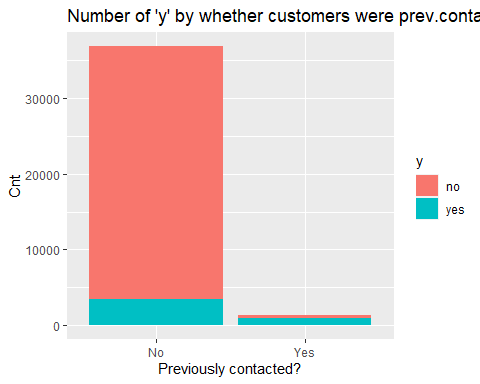
## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Highest frequency appears to be less than 10 days since last contact. Let’‘s make this into a y/n variable instead due to the large gap between days contacted and the ’999’ variable.

prevly\_Cntctd Yes/No. TO see the distribution or ‘Y’ on first time contact vs. a follow up

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?")

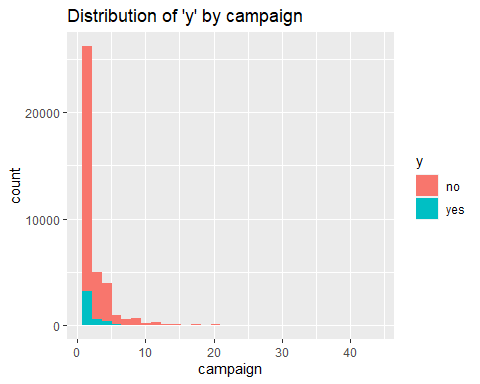


Same observation here as above: Most folks had no previous campaign but if they did, it looks like most who had a previous campaign decided to subscribe / likely to say ‘Yes’.

### Analysing campaign

ggplot(df) +   
 geom\_histogram(mapping = aes(x=campaign, fill=y)) +   
 ggtitle("Distribution of 'y' by campaign")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

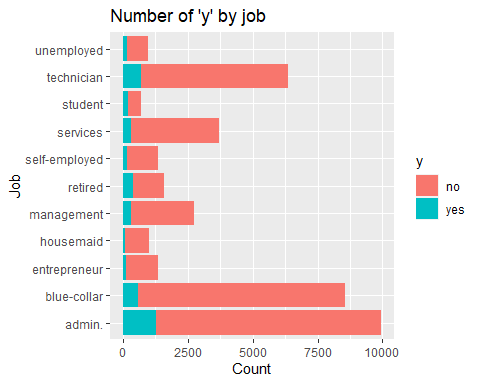


Just visually, when we decided to stop contacting a person it didn’t affect our closing ratio which still dropped off precipitously

Ideally, the campaign would stop contacting people who are less likely to subscribe, and keep contacting people if they are more likely to subscribe. Then we should see the ratio of Yes to No go up as the number of no contacts goes up. Instead, it looks like the ratio stays the same and the number of Yes’’s drops proportionately with the number of No’s.

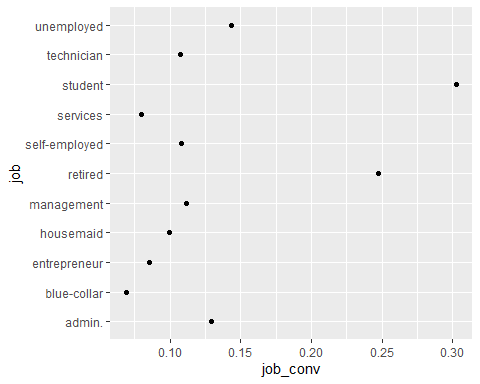
### Analyzing job

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")



“y” - has a client subscribed a term deposit? : admin, technician and blue collar jobs are the top 3 subscribers by volume

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()

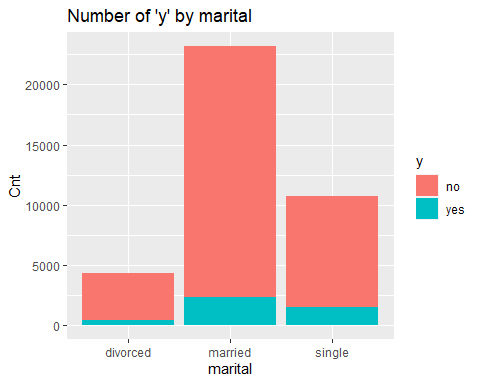


Above, I looked at the ratio of “yes” vs “no” and see that students and retired persons convert at much higher rates than those of other professions. And ‘blue collar’ has the lowest conversion rate

So, if they were to want to improve the cost effectiveness of their campaigns they might want to target more ‘students’ and ‘retirees’

### Analyzing marital

ggplot(data = df) +   
 geom\_bar(mapping = aes(x = marital, fill = y)) +   
 ggtitle("Number of 'y' by marital") +   
 ylab("Cnt") +   
 xlab("marital")



More ‘married’ people are represented in the campaign Visually looking, conversion rate seems to be higher for ‘single’ people

### Analyzing duration and creating duration group variable

summary(df$duration)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 102.0 180.0 258.2 319.0 4918.0

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")

Missing Values in Data

variable

missing

age

0

job

0

marital

0

education

0

default

0

housing

0

loan

0

contact

0

month

0

day\_of\_week

0

duration

0

campaign

0

pdays

0

previous

0

poutcome

0

emp\_var\_rate

0

cons\_price\_idx

0

cons\_conf\_idx

0

euribor3m

0

nr\_employed

0

y

0

Age\_Grp

0

prevly\_Cntctd

0

duration\_group

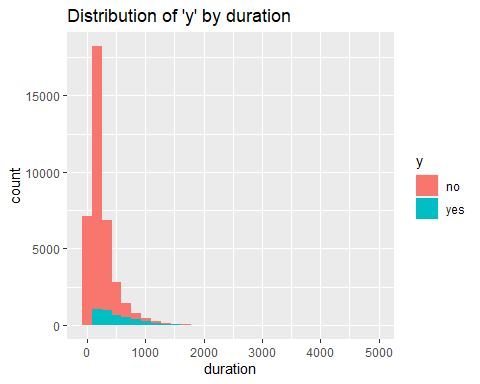
0

df3 <- df %>%   
 group\_by(duration\_group) %>%   
 count(y) %>%   
 mutate(duration\_group\_conv = n/sum(n)) %>%   
 filter(y == "yes")  
  
df3

## # A tibble: 3 x 4  
## # Groups: duration\_group [3]  
## duration\_group y n duration\_group\_conv  
## <fct> <fct> <int> <dbl>  
## 1 0-5min yes 1375 0.0494  
## 2 5-10min yes 1323 0.184   
## 3 10+ min yes 1560 0.486

ggplot(df) + geom\_histogram(mapping = aes(x = duration, fill = y)) +  
 ggtitle("Distribution of 'y' by duration")

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

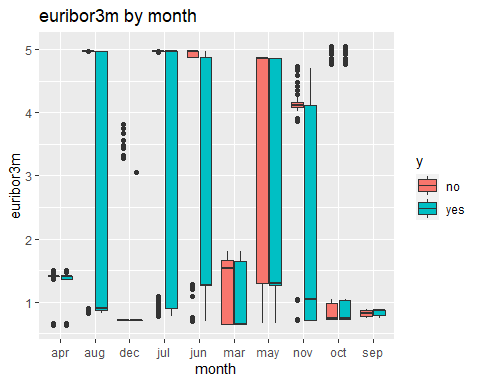


#ggplot(df3, aes(x=duration\_group, y=duration\_group\_conv)) + geom\_point() + facet\_wrap(~ y)

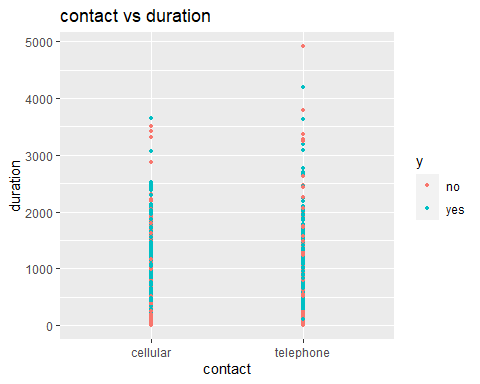
Looking above, clearly conversion rate goes up the longer the most recent call

Additional visualizations:

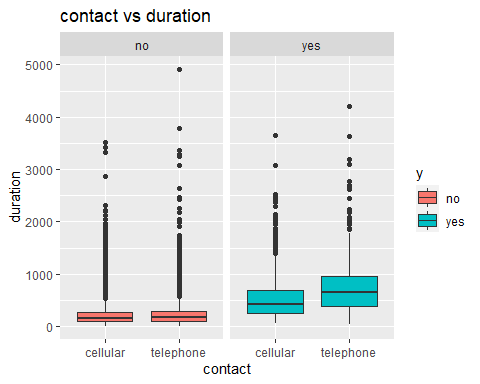
ggplot(df, aes(x = month , y = euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("euribor3m by month")



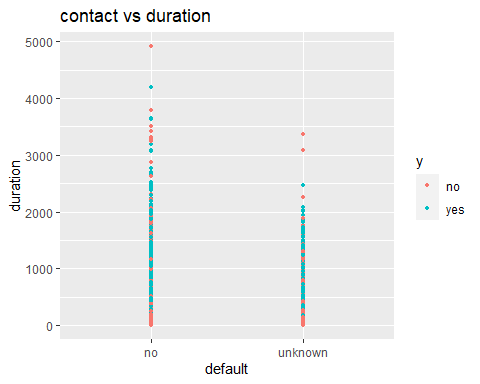
ggplot(df, aes(x=contact, y=duration, color = y)) +   
 geom\_point(size=1, shape="circle") +   
 ggtitle("contact vs duration")



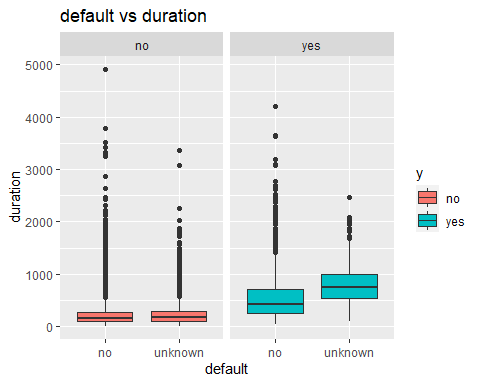
ggplot(df, aes(x=contact, y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("contact vs duration") +  
 facet\_grid(~y)



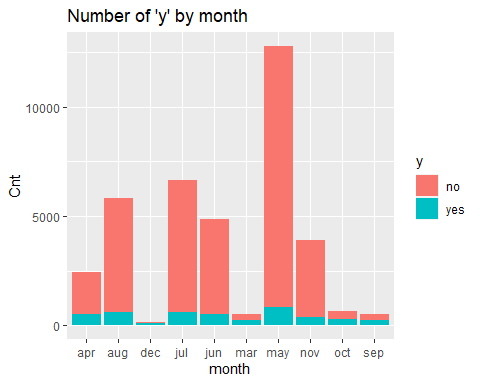
ggplot(df, aes(x=default, y=duration, color = y)) +   
 geom\_point(size=1, shape="circle") +   
 ggtitle("contact vs duration")



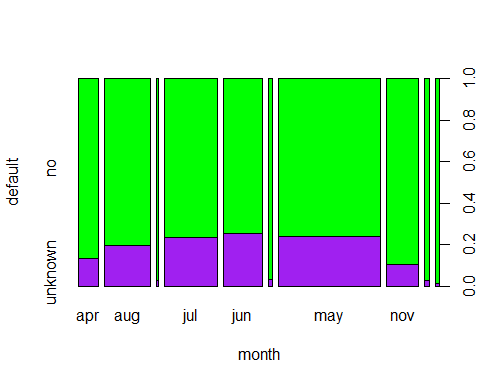
ggplot(df, aes(x=default, y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("default vs duration") +  
 facet\_grid(~y)



ggplot(df) +   
 geom\_bar(mapping = aes(x=month, fill = y)) +   
 ggtitle("Number of 'y' by month") +  
 ylab("Cnt") + xlab("month")



plot(default~month,data=df,col=c("purple","green"))

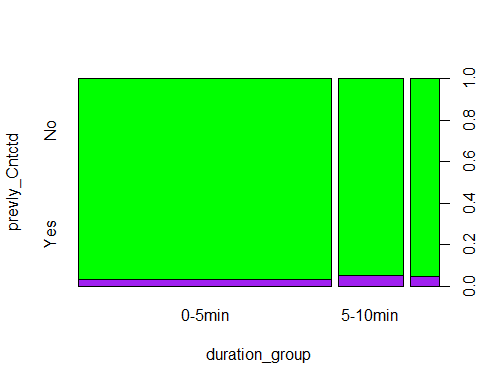


### Visualizing categorical variables

prop.table(table(df$prevly\_Cntctd,df$duration\_group),2)

##   
## 0-5min 5-10min 10+ min  
## No 0.96993858 0.94741967 0.95295950  
## Yes 0.03006142 0.05258033 0.04704050

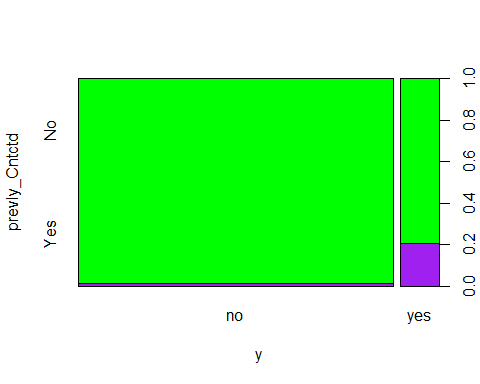
plot(prevly\_Cntctd~duration\_group,data=df,col=c("purple","green"))



prop.table(table(df$prevly\_Cntctd,df$y),2)

##   
## no yes  
## No 0.98525777 0.79685298  
## Yes 0.01474223 0.20314702

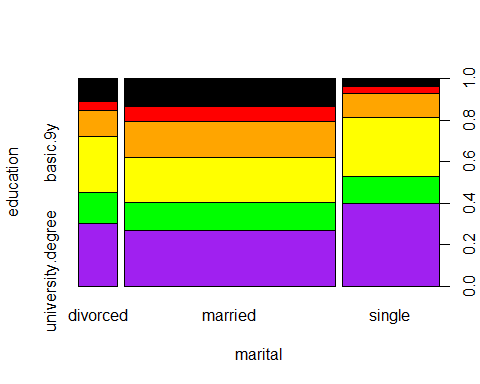
plot(prevly\_Cntctd~y,data=df,col=c("purple","green"))



prop.table(table(df$education,df$marital),2)

##   
## divorced married single  
## basic.4y 0.1113435611 0.1334771355 0.0398698885  
## basic.6y 0.0404463040 0.0738567731 0.0295539033  
## basic.9y 0.1276150628 0.1742450388 0.1178438662  
## high.school 0.2701069270 0.2164365833 0.2847583643  
## illiterate 0.0004649000 0.0006471096 0.0000929368  
## professional.course 0.1480706648 0.1327868852 0.1285315985  
## university.degree 0.3019525802 0.2685504745 0.3993494424

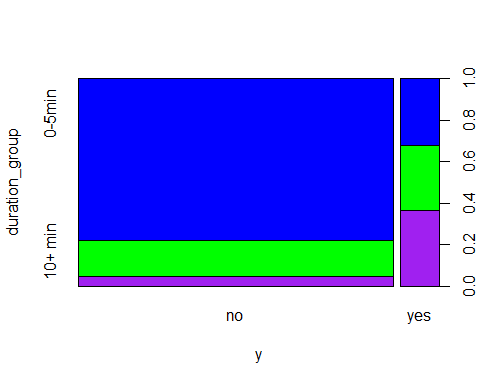
plot(education~marital,data=df,col=c("purple","green","blue","yellow","orange","red","black"))



prop.table(table(df$duration\_group,df$y),2)

##   
## no yes  
## 0-5min 0.77883710 0.32292156  
## 5-10min 0.17261064 0.31070925  
## 10+ min 0.04855226 0.36636919

plot(duration\_group~y,data=df,col=c("purple","green","blue"))



Checking proportions for duration groups and response var

prop.table(table(df$y,df$duration\_group),2)

##   
## 0-5min 5-10min 10+ min  
## no 0.95061595 0.81596884 0.51401869  
## yes 0.04938405 0.18403116 0.48598131

Check conversion rates by education levels

df %>%   
 group\_by(education) %>%   
 count(y) %>%   
 mutate(education\_conv = n/sum(n)) %>%   
 filter(y == "yes")

## # A tibble: 7 x 4  
## # Groups: education [7]  
## education y n education\_conv  
## <fct> <fct> <int> <dbl>  
## 1 basic.4y yes 412 0.103   
## 2 basic.6y yes 180 0.0817  
## 3 basic.9y yes 457 0.0780  
## 4 high.school yes 1007 0.109   
## 5 illiterate yes 4 0.222   
## 6 professional.course yes 578 0.113   
## 7 university.degree yes 1620 0.137

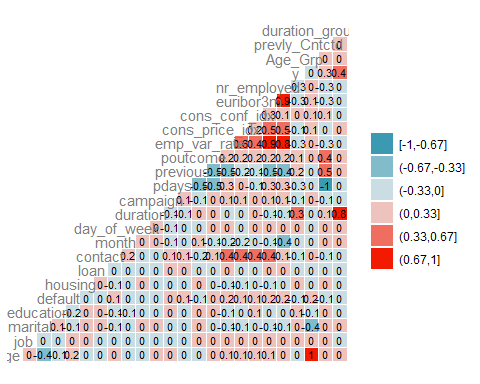
Check by marital status

df %>%   
 group\_by(education) %>%   
 count(y) %>%   
 mutate(education\_conv = n/sum(n)) %>%   
 filter(y == "yes")

## # A tibble: 7 x 4  
## # Groups: education [7]  
## education y n education\_conv  
## <fct> <fct> <int> <dbl>  
## 1 basic.4y yes 412 0.103   
## 2 basic.6y yes 180 0.0817  
## 3 basic.9y yes 457 0.0780  
## 4 high.school yes 1007 0.109   
## 5 illiterate yes 4 0.222   
## 6 professional.course yes 578 0.113   
## 7 university.degree yes 1620 0.137

# Checking for correlation

# 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")



Based on the correlation plot above, we see high correlation between ‘euribor3m’ and ‘emp\_var\_rate’ and to a lesser degree with ‘nr\_employed.’ We also see ‘nr\_employed’ and ‘emp\_var\_rate’ also highly correlated, which makes sense since you would expect the number of employees to vary at the same time as the employment variation rate. We will use VIF and feature selection tools in our model building to determine which (if any) to remove.

### Run random forest on down-sampled data set to check for variable importance

I am running RF on a subset of the data to do a gross check for important variables and to determine if the new variables duration group and age group are deemed more important than the continuous variables of just raw duration and raw age.

#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)

##   
## no yes   
## 1118 1118

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

##   
## Recursive feature selection  
##   
## Outer resampling method: Cross-Validated (5 fold)   
##   
## Resampling performance over subset size:  
##   
## Variables Accuracy Kappa AccuracySD KappaSD Selected  
## 1 0.6757 0.3515 0.029445 0.05889   
## 2 0.7272 0.4544 0.015716 0.03143   
## 3 0.8318 0.6636 0.034271 0.06854   
## 4 0.8591 0.7182 0.012716 0.02543   
## 5 0.8618 0.7236 0.009745 0.01949   
## 10 0.8622 0.7245 0.010100 0.02020   
## 15 0.8690 0.7379 0.005141 0.01028   
## 20 0.8716 0.7433 0.008989 0.01798   
## 23 0.8734 0.7469 0.006815 0.01363 \*  
##   
## The top 5 variables (out of 23):  
## duration, duration\_group, euribor3m, month, nr\_employed

varImp(RFresults)

## Overall  
## duration 42.8009884  
## duration\_group 24.6607671  
## euribor3m 20.9225935  
## month 18.9505712  
## nr\_employed 17.8288501  
## emp\_var\_rate 17.7271099  
## cons\_conf\_idx 12.1012329  
## pdays 10.7461341  
## cons\_price\_idx 10.0699347  
## prevly\_Cntctd 9.2007580  
## poutcome 8.5853345  
## contact 8.4388360  
## age 6.9608564  
## previous 5.3678243  
## job 5.0407489  
## default 4.8468608  
## Age\_Grp 4.6298576  
## day\_of\_week 3.9637829  
## campaign 3.8635042  
## education 2.6970050  
## marital 1.5615360  
## loan -0.5820975  
## housing -0.5824641

#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)

# Train/Test Split

summary(df)

## age job marital   
## Min. :17.00 admin. :9937 divorced: 4302   
## 1st Qu.:32.00 blue-collar:8560 married :23180   
## Median :38.00 technician :6378 single :10760   
## Mean :39.86 services :3716   
## 3rd Qu.:47.00 management :2728   
## Max. :98.00 retired :1577   
## (Other) :5346   
## education default housing loan   
## basic.4y : 4002 no :30485 no :17665 no :32283   
## basic.6y : 2204 unknown: 7757 yes:20577 yes: 5959   
## basic.9y : 5856   
## high.school : 9243   
## illiterate : 18   
## professional.course: 5098   
## university.degree :11821   
## contact month day\_of\_week duration   
## cellular :24438 may :12794 fri:7224 Min. : 0.0   
## telephone:13804 jul : 6630 mon:7927 1st Qu.: 102.0   
## aug : 5820 thu:8011 Median : 180.0   
## jun : 4846 tue:7478 Mean : 258.2   
## nov : 3897 wed:7602 3rd Qu.: 319.0   
## apr : 2436 Max. :4918.0   
## (Other): 1819   
## campaign pdays previous poutcome   
## Min. : 1.000 Min. : 0.0 Min. :0.00 failure : 3935   
## 1st Qu.: 1.000 1st Qu.:999.0 1st Qu.:0.00 nonexistent:33064   
## Median : 2.000 Median :999.0 Median :0.00 success : 1243   
## Mean : 2.567 Mean :963.5 Mean :0.17   
## 3rd Qu.: 3.000 3rd Qu.:999.0 3rd Qu.:0.00   
## Max. :43.000 Max. :999.0 Max. :7.00   
##   
## emp\_var\_rate cons\_price\_idx cons\_conf\_idx euribor3m   
## Min. :-3.4000 Min. :92.20 Min. :-50.80 Min. :0.634   
## 1st Qu.:-1.8000 1st Qu.:93.08 1st Qu.:-42.70 1st Qu.:1.344   
## Median : 1.1000 Median :93.44 Median :-41.80 Median :4.857   
## Mean : 0.0828 Mean :93.57 Mean :-40.54 Mean :3.623   
## 3rd Qu.: 1.4000 3rd Qu.:93.99 3rd Qu.:-36.40 3rd Qu.:4.961   
## Max. : 1.4000 Max. :94.77 Max. :-26.90 Max. :5.045   
##   
## nr\_employed Age\_Grp prevly\_Cntctd duration\_group y   
## Min. :4964 17-31:8740 No :36876 0-5min :27843 no :33984   
## 1st Qu.:5099 32-37:9893 Yes: 1366 5-10min: 7189 yes: 4258   
## Median :5191 38-47:9834 10+ min: 3210   
## Mean :5167 47-55:6598   
## 3rd Qu.:5228 >55 :3177   
## Max. :5228   
##

#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

## [1] 3406

nrow(test\_yes) #852

## [1] 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

## [1] 27187

nrow(test\_no) #6797

## [1] 6797

train <- rbind(train\_yes, train\_no)  
test <- rbind(test\_yes, test\_no)  
nrow(train) #30,593

## [1] 30593

nrow(test) #7,649

## [1] 7649

nrow(train %>% filter(y=='yes')) #3,406

## [1] 3406

nrow(test %>% filter(y=='yes')) #852

## [1] 852

summary(train)

## age job marital education   
## Min. :17.00 admin. :7949 divorced: 3461 basic.4y :3123   
## 1st Qu.:32.00 blue-collar:6846 married :18542 basic.6y :1758   
## Median :38.00 technician :5157 single : 8590 basic.9y :4694   
## Mean :39.83 services :2956 high.school :7444   
## 3rd Qu.:47.00 management :2191 illiterate : 12   
## Max. :98.00 retired :1252 professional.course:4092   
## (Other) :4242 university.degree :9470   
## default housing loan contact month   
## no :24427 no :14200 no :25850 cellular :19613 may :10156   
## unknown: 6166 yes:16393 yes: 4743 telephone:10980 jul : 5340   
## aug : 4713   
## jun : 3884   
## nov : 3119   
## apr : 1943   
## (Other): 1438   
## day\_of\_week duration campaign pdays previous   
## fri:5807 Min. : 0.0 Min. : 1.000 Min. : 0.0 Min. :0.00   
## mon:6307 1st Qu.: 103.0 1st Qu.: 1.000 1st Qu.:999.0 1st Qu.:0.00   
## thu:6392 Median : 179.0 Median : 2.000 Median :999.0 Median :0.00   
## tue:5993 Mean : 257.3 Mean : 2.571 Mean :963.6 Mean :0.17   
## wed:6094 3rd Qu.: 318.0 3rd Qu.: 3.000 3rd Qu.:999.0 3rd Qu.:0.00   
## Max. :4199.0 Max. :43.000 Max. :999.0 Max. :7.00   
##   
## poutcome emp\_var\_rate cons\_price\_idx cons\_conf\_idx   
## failure : 3159 Min. :-3.40000 Min. :92.20 Min. :-50.80   
## nonexistent:26452 1st Qu.:-1.80000 1st Qu.:93.08 1st Qu.:-42.70   
## success : 982 Median : 1.10000 Median :93.44 Median :-41.80   
## Mean : 0.08542 Mean :93.57 Mean :-40.53   
## 3rd Qu.: 1.40000 3rd Qu.:93.99 3rd Qu.:-36.40   
## Max. : 1.40000 Max. :94.77 Max. :-26.90   
##   
## euribor3m nr\_employed Age\_Grp prevly\_Cntctd duration\_group   
## Min. :0.634 Min. :4964 17-31:6981 No :29503 0-5min :22312   
## 1st Qu.:1.344 1st Qu.:5099 32-37:7969 Yes: 1090 5-10min: 5756   
## Median :4.857 Median :5191 38-47:7853 10+ min: 2525   
## Mean :3.625 Mean :5168 47-55:5273   
## 3rd Qu.:4.961 3rd Qu.:5228 >55 :2517   
## Max. :5.045 Max. :5228   
##   
## y   
## no :27187   
## yes: 3406   
##   
##   
##   
##   
##

#30593 obs. of 24 variables  
#write.csv(train, "../data/train.csv", row.names = FALSE)  
#write.csv(test, "../data/test.csv", row.names = FALSE)

# Simple Logistic Model

# Run Initial Logistic Regression  
#Simple regression model  
simple.log<-glm(y~.,family="binomial",data=train)  
summary(simple.log)

##   
## Call:  
## glm(formula = y ~ ., family = "binomial", data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.5608 -0.2992 -0.1624 -0.1226 3.2477   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.223e+02 5.036e+01 -4.415 1.01e-05 \*\*\*  
## age 5.815e-03 7.219e-03 0.806 0.420486   
## jobblue-collar -2.436e-01 9.381e-02 -2.596 0.009422 \*\*   
## jobentrepreneur -1.341e-01 1.446e-01 -0.928 0.353604   
## jobhousemaid -1.386e-01 1.755e-01 -0.790 0.429541   
## jobmanagement -8.405e-02 1.002e-01 -0.839 0.401492   
## jobretired 5.598e-02 1.424e-01 0.393 0.694245   
## jobself-employed -1.335e-01 1.334e-01 -1.000 0.317109   
## jobservices -2.259e-01 1.010e-01 -2.237 0.025293 \*   
## jobstudent 2.154e-01 1.413e-01 1.525 0.127372   
## jobtechnician 3.799e-02 8.197e-02 0.463 0.643065   
## jobunemployed -7.248e-02 1.574e-01 -0.460 0.645255   
## maritalmarried -1.636e-02 8.011e-02 -0.204 0.838184   
## maritalsingle 1.179e-02 9.190e-02 0.128 0.897935   
## educationbasic.6y 2.053e-01 1.371e-01 1.497 0.134281   
## educationbasic.9y 2.475e-02 1.099e-01 0.225 0.821782   
## educationhigh.school 6.886e-02 1.079e-01 0.638 0.523458   
## educationilliterate 1.864e+00 8.450e-01 2.206 0.027403 \*   
## educationprofessional.course 1.060e-01 1.186e-01 0.894 0.371490   
## educationuniversity.degree 1.883e-01 1.088e-01 1.732 0.083355 .   
## defaultunknown -2.945e-01 7.873e-02 -3.740 0.000184 \*\*\*  
## housingyes -5.811e-03 4.794e-02 -0.121 0.903512   
## loanyes -2.902e-02 6.697e-02 -0.433 0.664705   
## contacttelephone -6.742e-01 9.203e-02 -7.326 2.37e-13 \*\*\*  
## monthaug 8.414e-01 1.446e-01 5.818 5.96e-09 \*\*\*  
## monthdec -4.116e-02 2.669e-01 -0.154 0.877428   
## monthjul 1.327e-01 1.138e-01 1.166 0.243486   
## monthjun -5.261e-01 1.525e-01 -3.449 0.000562 \*\*\*  
## monthmar 2.260e+00 1.730e-01 13.062 < 2e-16 \*\*\*  
## monthmay -4.213e-01 9.954e-02 -4.233 2.31e-05 \*\*\*  
## monthnov -4.946e-01 1.420e-01 -3.483 0.000496 \*\*\*  
## monthoct 1.552e-01 1.834e-01 0.846 0.397479   
## monthsep 2.513e-01 2.161e-01 1.163 0.244886   
## day\_of\_weekmon -9.602e-02 7.685e-02 -1.249 0.211494   
## day\_of\_weekthu 4.673e-02 7.480e-02 0.625 0.532111   
## day\_of\_weektue 6.786e-02 7.674e-02 0.884 0.376506   
## day\_of\_weekwed 9.268e-02 7.684e-02 1.206 0.227758   
## duration 2.479e-03 1.579e-04 15.703 < 2e-16 \*\*\*  
## campaign -3.410e-02 1.306e-02 -2.610 0.009054 \*\*   
## pdays -1.477e-02 2.098e-02 -0.704 0.481417   
## previous -3.861e-02 7.365e-02 -0.524 0.600111   
## poutcomenonexistent 4.198e-01 1.152e-01 3.645 0.000268 \*\*\*  
## poutcomesuccess 1.138e+00 2.728e-01 4.173 3.01e-05 \*\*\*  
## emp\_var\_rate -1.806e+00 1.717e-01 -10.513 < 2e-16 \*\*\*  
## cons\_price\_idx 2.243e+00 3.014e-01 7.441 1.00e-13 \*\*\*  
## cons\_conf\_idx 3.103e-02 9.273e-03 3.346 0.000820 \*\*\*  
## euribor3m 3.972e-01 1.515e-01 2.622 0.008748 \*\*   
## nr\_employed 4.434e-03 3.672e-03 1.207 0.227292   
## Age\_Grp32-37 -2.062e-01 8.138e-02 -2.534 0.011270 \*   
## Age\_Grp38-47 -3.701e-01 1.204e-01 -3.074 0.002113 \*\*   
## Age\_Grp47-55 -2.579e-01 1.774e-01 -1.454 0.145967   
## Age\_Grp>55 -2.183e-01 2.491e-01 -0.877 0.380654   
## prevly\_CntctdYes -1.387e+01 2.074e+01 -0.669 0.503489   
## duration\_group5-10min 1.135e+00 7.127e-02 15.930 < 2e-16 \*\*\*  
## duration\_group10+ min 2.075e+00 1.323e-01 15.689 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12329 on 30538 degrees of freedom  
## AIC: 12439  
##   
## Number of Fisher Scoring iterations: 6

exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))

## Odds ratio 2.5 % 97.5 %  
## (Intercept) 2.755572e-97 3.757741e-140 2.020676e-54  
## age 1.005832e+00 9.917014e-01 1.020164e+00  
## jobblue-collar 7.838373e-01 6.521962e-01 9.420492e-01  
## jobentrepreneur 8.744637e-01 6.586404e-01 1.161008e+00  
## jobhousemaid 8.705519e-01 6.171960e-01 1.227909e+00  
## jobmanagement 9.193843e-01 7.554739e-01 1.118857e+00  
## jobretired 1.057577e+00 8.000080e-01 1.398073e+00  
## jobself-employed 8.750683e-01 6.737419e-01 1.136555e+00  
## jobservices 7.977859e-01 6.545150e-01 9.724181e-01  
## jobstudent 1.240405e+00 9.403251e-01 1.636249e+00  
## jobtechnician 1.038717e+00 8.845530e-01 1.219750e+00  
## jobunemployed 9.300874e-01 6.831499e-01 1.266285e+00  
## maritalmarried 9.837732e-01 8.408233e-01 1.151026e+00  
## maritalsingle 1.011858e+00 8.450698e-01 1.211565e+00  
## educationbasic.6y 1.227928e+00 9.385468e-01 1.606533e+00  
## educationbasic.9y 1.025059e+00 8.264621e-01 1.271377e+00  
## educationhigh.school 1.071284e+00 8.670414e-01 1.323638e+00  
## educationilliterate 6.449192e+00 1.230831e+00 3.379188e+01  
## educationprofessional.course 1.111788e+00 8.812320e-01 1.402663e+00  
## educationuniversity.degree 1.207250e+00 9.754599e-01 1.494117e+00  
## defaultunknown 7.449007e-01 6.383800e-01 8.691954e-01  
## housingyes 9.942054e-01 9.050452e-01 1.092149e+00  
## loanyes 9.713926e-01 8.519102e-01 1.107633e+00  
## contacttelephone 5.095418e-01 4.254416e-01 6.102666e-01  
## monthaug 2.319545e+00 1.747032e+00 3.079673e+00  
## monthdec 9.596751e-01 5.687990e-01 1.619160e+00  
## monthjul 1.141908e+00 9.136581e-01 1.427179e+00  
## monthjun 5.909148e-01 4.382230e-01 7.968096e-01  
## monthmar 9.582710e+00 6.826786e+00 1.345118e+01  
## monthmay 6.561732e-01 5.398673e-01 7.975355e-01  
## monthnov 6.098405e-01 4.616810e-01 8.055463e-01  
## monthoct 1.167882e+00 8.152245e-01 1.673096e+00  
## monthsep 1.285756e+00 8.417352e-01 1.963999e+00  
## day\_of\_weekmon 9.084426e-01 7.814141e-01 1.056121e+00  
## day\_of\_weekthu 1.047842e+00 9.049554e-01 1.213290e+00  
## day\_of\_weektue 1.070218e+00 9.207758e-01 1.243914e+00  
## day\_of\_weekwed 1.097114e+00 9.437215e-01 1.275439e+00  
## duration 1.002482e+00 1.002172e+00 1.002792e+00  
## campaign 9.664782e-01 9.420461e-01 9.915439e-01  
## pdays 9.853384e-01 9.456437e-01 1.026699e+00  
## previous 9.621262e-01 8.328027e-01 1.111532e+00  
## poutcomenonexistent 1.521587e+00 1.214123e+00 1.906913e+00  
## poutcomesuccess 3.121867e+00 1.828929e+00 5.328830e+00  
## emp\_var\_rate 1.643902e-01 1.174062e-01 2.301765e-01  
## cons\_price\_idx 9.419212e+00 5.217318e+00 1.700520e+01  
## cons\_conf\_idx 1.031513e+00 1.012934e+00 1.050432e+00  
## euribor3m 1.487657e+00 1.105461e+00 2.001992e+00  
## nr\_employed 1.004443e+00 9.972402e-01 1.011699e+00  
## Age\_Grp32-37 8.136361e-01 6.936764e-01 9.543409e-01  
## Age\_Grp38-47 6.906583e-01 5.454750e-01 8.744835e-01  
## Age\_Grp47-55 7.726490e-01 5.457288e-01 1.093925e+00  
## Age\_Grp>55 8.038517e-01 4.933828e-01 1.309688e+00  
## prevly\_CntctdYes 9.425133e-07 2.094494e-24 4.241268e+11  
## duration\_group5-10min 3.112525e+00 2.706723e+00 3.579168e+00  
## duration\_group10+ min 7.968439e+00 6.148455e+00 1.032715e+01

vif(simple.log)

## GVIF Df GVIF^(1/(2\*Df))  
## age 13.821928 1 3.717785  
## job 7.636632 10 1.106994  
## marital 1.484682 2 1.103846  
## education 3.306893 6 1.104803  
## default 1.145764 1 1.070403  
## housing 1.017146 1 1.008537  
## loan 1.008311 1 1.004147  
## contact 2.542053 1 1.594382  
## month 70.173060 9 1.266383  
## day\_of\_week 1.069342 4 1.008416  
## duration 4.838922 1 2.199755  
## campaign 1.057760 1 1.028475  
## pdays 71073.146963 1 266.595474  
## previous 4.756874 1 2.181026  
## poutcome 28.653541 2 2.313634  
## emp\_var\_rate 156.519117 1 12.510760  
## cons\_price\_idx 69.869415 1 8.358793  
## cons\_conf\_idx 5.321632 1 2.306866  
## euribor3m 138.853417 1 11.783608  
## nr\_employed 175.816806 1 13.259593  
## Age\_Grp 17.119294 4 1.426217  
## prevly\_Cntctd 70425.497503 1 265.378028  
## duration\_group 5.251183 2 1.513785

Remove either pdays or prevly\_Cntctd - think prevly\_Cntcted will be more useful, remove pdays and re-run.

train\_simple <- train %>% dplyr::select(-pdays)  
#Check vifs again  
simple.log<-glm(y~.,family="binomial",data=train\_simple)  
summary(simple.log)

##   
## Call:  
## glm(formula = y ~ ., family = "binomial", data = train\_simple)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.5593 -0.2993 -0.1624 -0.1226 3.2471   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.375e+02 4.553e+01 -5.216 1.83e-07 \*\*\*  
## age 5.858e-03 7.218e-03 0.812 0.416988   
## jobblue-collar -2.422e-01 9.378e-02 -2.583 0.009799 \*\*   
## jobentrepreneur -1.336e-01 1.446e-01 -0.924 0.355656   
## jobhousemaid -1.379e-01 1.755e-01 -0.786 0.431987   
## jobmanagement -8.348e-02 1.002e-01 -0.833 0.404646   
## jobretired 5.796e-02 1.423e-01 0.407 0.683836   
## jobself-employed -1.323e-01 1.334e-01 -0.992 0.321208   
## jobservices -2.240e-01 1.009e-01 -2.219 0.026458 \*   
## jobstudent 2.167e-01 1.413e-01 1.534 0.125148   
## jobtechnician 3.829e-02 8.197e-02 0.467 0.640433   
## jobunemployed -7.072e-02 1.574e-01 -0.449 0.653190   
## maritalmarried -1.668e-02 8.010e-02 -0.208 0.835065   
## maritalsingle 1.152e-02 9.189e-02 0.125 0.900241   
## educationbasic.6y 2.053e-01 1.371e-01 1.498 0.134262   
## educationbasic.9y 2.398e-02 1.099e-01 0.218 0.827248   
## educationhigh.school 6.940e-02 1.079e-01 0.643 0.520145   
## educationilliterate 1.862e+00 8.452e-01 2.203 0.027573 \*   
## educationprofessional.course 1.064e-01 1.186e-01 0.897 0.369580   
## educationuniversity.degree 1.899e-01 1.087e-01 1.747 0.080704 .   
## defaultunknown -2.940e-01 7.872e-02 -3.735 0.000188 \*\*\*  
## housingyes -5.333e-03 4.793e-02 -0.111 0.911406   
## loanyes -2.815e-02 6.694e-02 -0.421 0.674081   
## contacttelephone -6.737e-01 9.203e-02 -7.321 2.47e-13 \*\*\*  
## monthaug 8.414e-01 1.446e-01 5.817 5.98e-09 \*\*\*  
## monthdec -4.464e-02 2.669e-01 -0.167 0.867171   
## monthjul 1.309e-01 1.138e-01 1.151 0.249938   
## monthjun -5.278e-01 1.525e-01 -3.461 0.000539 \*\*\*  
## monthmar 2.260e+00 1.730e-01 13.063 < 2e-16 \*\*\*  
## monthmay -4.219e-01 9.954e-02 -4.238 2.25e-05 \*\*\*  
## monthnov -4.953e-01 1.420e-01 -3.489 0.000485 \*\*\*  
## monthoct 1.543e-01 1.834e-01 0.841 0.400094   
## monthsep 2.523e-01 2.161e-01 1.168 0.243004   
## day\_of\_weekmon -9.580e-02 7.685e-02 -1.247 0.212544   
## day\_of\_weekthu 4.696e-02 7.480e-02 0.628 0.530075   
## day\_of\_weektue 6.759e-02 7.673e-02 0.881 0.378427   
## day\_of\_weekwed 9.298e-02 7.683e-02 1.210 0.226226   
## duration 2.479e-03 1.579e-04 15.702 < 2e-16 \*\*\*  
## campaign -3.409e-02 1.306e-02 -2.609 0.009071 \*\*   
## previous -2.943e-02 7.242e-02 -0.406 0.684513   
## poutcomenonexistent 4.299e-01 1.142e-01 3.764 0.000167 \*\*\*  
## poutcomesuccess 1.211e+00 2.525e-01 4.797 1.61e-06 \*\*\*  
## emp\_var\_rate -1.811e+00 1.716e-01 -10.551 < 2e-16 \*\*\*  
## cons\_price\_idx 2.244e+00 3.014e-01 7.445 9.68e-14 \*\*\*  
## cons\_conf\_idx 3.106e-02 9.273e-03 3.349 0.000810 \*\*\*  
## euribor3m 3.995e-01 1.514e-01 2.638 0.008339 \*\*   
## nr\_employed 4.484e-03 3.671e-03 1.221 0.221960   
## Age\_Grp32-37 -2.064e-01 8.138e-02 -2.536 0.011204 \*   
## Age\_Grp38-47 -3.705e-01 1.204e-01 -3.078 0.002086 \*\*   
## Age\_Grp47-55 -2.582e-01 1.774e-01 -1.455 0.145572   
## Age\_Grp>55 -2.192e-01 2.490e-01 -0.880 0.378641   
## prevly\_CntctdYes 7.248e-01 2.574e-01 2.816 0.004867 \*\*   
## duration\_group5-10min 1.135e+00 7.128e-02 15.927 < 2e-16 \*\*\*  
## duration\_group10+ min 2.076e+00 1.323e-01 15.690 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12329 on 30539 degrees of freedom  
## AIC: 12437  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)

## GVIF Df GVIF^(1/(2\*Df))  
## age 13.821213 1 3.717689  
## job 7.623054 10 1.106895  
## marital 1.484784 2 1.103865  
## education 3.301390 6 1.104650  
## default 1.145646 1 1.070348  
## housing 1.016929 1 1.008429  
## loan 1.007988 1 1.003986  
## contact 2.541775 1 1.594294  
## month 70.151837 9 1.266362  
## day\_of\_week 1.069201 4 1.008399  
## duration 4.839426 1 2.199870  
## campaign 1.057798 1 1.028493  
## previous 4.604360 1 2.145777  
## poutcome 24.503123 2 2.224874  
## emp\_var\_rate 156.298337 1 12.501933  
## cons\_price\_idx 69.874867 1 8.359119  
## cons\_conf\_idx 5.323534 1 2.307279  
## euribor3m 138.744098 1 11.778968  
## nr\_employed 175.716197 1 13.255799  
## Age\_Grp 17.116671 4 1.426190  
## prevly\_Cntctd 10.849640 1 3.293879  
## duration\_group 5.251773 2 1.513828

VIFs are still high for euribor3m and nr\_employed, and model shows nr\_employed as significant. Remove nr\_employed and emp\_var\_rate and see if things change

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)

##   
## Call:  
## glm(formula = y ~ ., family = "binomial", data = train\_simple\_2)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4487 -0.3011 -0.1734 -0.1205 3.3117   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.602e+01 5.270e+00 -8.733 < 2e-16 \*\*\*  
## age 7.291e-03 7.223e-03 1.009 0.312808   
## jobblue-collar -2.783e-01 9.360e-02 -2.974 0.002943 \*\*   
## jobentrepreneur -1.376e-01 1.438e-01 -0.957 0.338654   
## jobhousemaid -1.609e-01 1.757e-01 -0.916 0.359648   
## jobmanagement -8.610e-02 9.972e-02 -0.863 0.387911   
## jobretired 6.410e-02 1.419e-01 0.452 0.651439   
## jobself-employed -1.256e-01 1.329e-01 -0.945 0.344854   
## jobservices -2.295e-01 1.004e-01 -2.286 0.022234 \*   
## jobstudent 2.494e-01 1.408e-01 1.771 0.076584 .   
## jobtechnician 2.409e-02 8.135e-02 0.296 0.767103   
## jobunemployed -7.861e-02 1.560e-01 -0.504 0.614275   
## maritalmarried -4.442e-03 7.974e-02 -0.056 0.955580   
## maritalsingle 2.461e-02 9.143e-02 0.269 0.787803   
## educationbasic.6y 1.961e-01 1.368e-01 1.433 0.151799   
## educationbasic.9y 2.684e-02 1.097e-01 0.245 0.806752   
## educationhigh.school 4.574e-02 1.078e-01 0.424 0.671243   
## educationilliterate 1.779e+00 8.447e-01 2.106 0.035171 \*   
## educationprofessional.course 7.647e-02 1.186e-01 0.645 0.519135   
## educationuniversity.degree 1.859e-01 1.086e-01 1.711 0.087002 .   
## defaultunknown -3.222e-01 7.862e-02 -4.099 4.15e-05 \*\*\*  
## housingyes -4.285e-03 4.767e-02 -0.090 0.928376   
## loanyes -4.179e-02 6.660e-02 -0.627 0.530343   
## contacttelephone -3.683e-01 8.232e-02 -4.473 7.70e-06 \*\*\*  
## monthaug 1.854e-01 1.211e-01 1.531 0.125811   
## monthdec 1.522e-02 2.520e-01 0.060 0.951840   
## monthjul 3.201e-01 1.119e-01 2.860 0.004233 \*\*   
## monthjun 3.806e-01 1.089e-01 3.495 0.000475 \*\*\*  
## monthmar 1.723e+00 1.440e-01 11.969 < 2e-16 \*\*\*  
## monthmay -6.451e-01 8.961e-02 -7.198 6.09e-13 \*\*\*  
## monthnov 4.417e-02 1.168e-01 0.378 0.705376   
## monthoct 3.988e-01 1.482e-01 2.692 0.007106 \*\*   
## monthsep -6.102e-02 1.617e-01 -0.377 0.705902   
## day\_of\_weekmon -1.039e-01 7.637e-02 -1.360 0.173843   
## day\_of\_weekthu 1.238e-02 7.434e-02 0.167 0.867719   
## day\_of\_weektue 6.751e-02 7.624e-02 0.885 0.375899   
## day\_of\_weekwed 7.427e-02 7.637e-02 0.972 0.330850   
## duration 2.516e-03 1.582e-04 15.902 < 2e-16 \*\*\*  
## campaign -4.371e-02 1.320e-02 -3.311 0.000930 \*\*\*  
## previous -1.026e-02 7.184e-02 -0.143 0.886435   
## poutcomenonexistent 4.882e-01 1.134e-01 4.305 1.67e-05 \*\*\*  
## poutcomesuccess 1.109e+00 2.486e-01 4.463 8.10e-06 \*\*\*  
## cons\_price\_idx 4.965e-01 5.798e-02 8.563 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.700e-02 6.168e-03 9.241 < 2e-16 \*\*\*  
## euribor3m -7.135e-01 2.253e-02 -31.668 < 2e-16 \*\*\*  
## Age\_Grp32-37 -2.258e-01 8.108e-02 -2.785 0.005355 \*\*   
## Age\_Grp38-47 -3.699e-01 1.202e-01 -3.078 0.002086 \*\*   
## Age\_Grp47-55 -2.822e-01 1.772e-01 -1.593 0.111141   
## Age\_Grp>55 -2.453e-01 2.485e-01 -0.987 0.323533   
## prevly\_CntctdYes 8.528e-01 2.536e-01 3.363 0.000772 \*\*\*  
## duration\_group5-10min 1.102e+00 7.053e-02 15.624 < 2e-16 \*\*\*  
## duration\_group10+ min 1.996e+00 1.318e-01 15.145 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12483 on 30541 degrees of freedom  
## AIC: 12587  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)

## GVIF Df GVIF^(1/(2\*Df))  
## age 13.872114 1 3.724529  
## job 7.614780 10 1.106835  
## marital 1.480832 2 1.103129  
## education 3.301974 6 1.104666  
## default 1.139451 1 1.067451  
## housing 1.015359 1 1.007650  
## loan 1.007564 1 1.003775  
## contact 2.011504 1 1.418275  
## month 5.563261 9 1.100037  
## day\_of\_week 1.056779 4 1.006927  
## duration 4.826047 1 2.196827  
## campaign 1.053851 1 1.026573  
## previous 4.601227 1 2.145047  
## poutcome 23.920157 2 2.211521  
## cons\_price\_idx 2.619965 1 1.618630  
## cons\_conf\_idx 2.367099 1 1.538538  
## euribor3m 3.096350 1 1.759645  
## Age\_Grp 17.174524 4 1.426792  
## prevly\_Cntctd 10.681927 1 3.268322  
## duration\_group 5.170012 2 1.507901

poutcome has higher VIF, but seems like it is practically significant. Remove age because it is significant and age\_grp is not - should lower VIF

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)

##   
## Call:  
## glm(formula = y ~ ., family = "binomial", data = train\_simple\_3)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4520 -0.3014 -0.1734 -0.1205 3.3154   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.603e+01 5.271e+00 -8.733 < 2e-16 \*\*\*  
## jobblue-collar -2.818e-01 9.354e-02 -3.012 0.002593 \*\*   
## jobentrepreneur -1.384e-01 1.439e-01 -0.962 0.336075   
## jobhousemaid -1.518e-01 1.752e-01 -0.867 0.386197   
## jobmanagement -8.534e-02 9.973e-02 -0.856 0.392144   
## jobretired 1.121e-01 1.333e-01 0.841 0.400441   
## jobself-employed -1.243e-01 1.329e-01 -0.935 0.349623   
## jobservices -2.315e-01 1.004e-01 -2.305 0.021153 \*   
## jobstudent 2.225e-01 1.384e-01 1.608 0.107799   
## jobtechnician 2.418e-02 8.136e-02 0.297 0.766348   
## jobunemployed -8.215e-02 1.560e-01 -0.527 0.598528   
## maritalmarried -1.056e-02 7.946e-02 -0.133 0.894267   
## maritalsingle 1.297e-02 9.066e-02 0.143 0.886232   
## educationbasic.6y 1.872e-01 1.364e-01 1.373 0.169852   
## educationbasic.9y 1.737e-02 1.092e-01 0.159 0.873620   
## educationhigh.school 3.362e-02 1.070e-01 0.314 0.753263   
## educationilliterate 1.754e+00 8.479e-01 2.069 0.038571 \*   
## educationprofessional.course 6.392e-02 1.178e-01 0.542 0.587497   
## educationuniversity.degree 1.740e-01 1.078e-01 1.614 0.106513   
## defaultunknown -3.217e-01 7.859e-02 -4.094 4.24e-05 \*\*\*  
## housingyes -4.493e-03 4.767e-02 -0.094 0.924914   
## loanyes -4.107e-02 6.659e-02 -0.617 0.537383   
## contacttelephone -3.704e-01 8.230e-02 -4.500 6.78e-06 \*\*\*  
## monthaug 1.851e-01 1.211e-01 1.529 0.126353   
## monthdec 1.831e-02 2.520e-01 0.073 0.942080   
## monthjul 3.163e-01 1.118e-01 2.828 0.004682 \*\*   
## monthjun 3.772e-01 1.088e-01 3.466 0.000527 \*\*\*  
## monthmar 1.726e+00 1.440e-01 11.980 < 2e-16 \*\*\*  
## monthmay -6.468e-01 8.958e-02 -7.220 5.21e-13 \*\*\*  
## monthnov 4.476e-02 1.168e-01 0.383 0.701632   
## monthoct 4.008e-01 1.481e-01 2.705 0.006823 \*\*   
## monthsep -6.175e-02 1.617e-01 -0.382 0.702546   
## day\_of\_weekmon -1.037e-01 7.637e-02 -1.358 0.174374   
## day\_of\_weekthu 1.267e-02 7.433e-02 0.171 0.864602   
## day\_of\_weektue 6.804e-02 7.623e-02 0.893 0.372120   
## day\_of\_weekwed 7.490e-02 7.637e-02 0.981 0.326698   
## duration 2.518e-03 1.583e-04 15.907 < 2e-16 \*\*\*  
## campaign -4.366e-02 1.320e-02 -3.307 0.000943 \*\*\*  
## previous -9.443e-03 7.186e-02 -0.131 0.895450   
## poutcomenonexistent 4.897e-01 1.134e-01 4.318 1.57e-05 \*\*\*  
## poutcomesuccess 1.113e+00 2.486e-01 4.476 7.60e-06 \*\*\*  
## cons\_price\_idx 4.992e-01 5.794e-02 8.616 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.730e-02 6.163e-03 9.297 < 2e-16 \*\*\*  
## euribor3m -7.146e-01 2.250e-02 -31.766 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.836e-01 6.951e-02 -2.641 0.008261 \*\*   
## Age\_Grp38-47 -2.760e-01 7.608e-02 -3.628 0.000286 \*\*\*  
## Age\_Grp47-55 -1.260e-01 8.629e-02 -1.461 0.144069   
## Age\_Grp>55 -2.272e-02 1.142e-01 -0.199 0.842304   
## prevly\_CntctdYes 8.477e-01 2.536e-01 3.342 0.000830 \*\*\*  
## duration\_group5-10min 1.101e+00 7.053e-02 15.609 < 2e-16 \*\*\*  
## duration\_group10+ min 1.994e+00 1.318e-01 15.129 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12484 on 30542 degrees of freedom  
## AIC: 12586  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 6.364741 10 1.096956  
## marital 1.456183 2 1.098510  
## education 3.250803 6 1.103230  
## default 1.139667 1 1.067552  
## housing 1.015331 1 1.007636  
## loan 1.007453 1 1.003720  
## contact 2.010726 1 1.418001  
## month 5.537815 9 1.099757  
## day\_of\_week 1.056630 4 1.006909  
## duration 4.827719 1 2.197207  
## campaign 1.053827 1 1.026561  
## previous 4.602195 1 2.145273  
## poutcome 23.925351 2 2.211641  
## cons\_price\_idx 2.615900 1 1.617374  
## cons\_conf\_idx 2.362573 1 1.537066  
## euribor3m 3.087530 1 1.757137  
## Age\_Grp 2.717304 4 1.133097  
## prevly\_Cntctd 10.681581 1 3.268269  
## duration\_group 5.170014 2 1.507901

Remove statistically insignificant values

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)

##   
## Call:  
## glm(formula = y ~ ., family = "binomial", data = train\_simple\_4)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4536 -0.3021 -0.1733 -0.1206 3.3331   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.555e+01 5.100e+00 -8.930 < 2e-16 \*\*\*  
## jobblue-collar -2.796e-01 9.329e-02 -2.997 0.002727 \*\*   
## jobentrepreneur -1.412e-01 1.435e-01 -0.984 0.325046   
## jobhousemaid -1.437e-01 1.748e-01 -0.822 0.411089   
## jobmanagement -8.590e-02 9.919e-02 -0.866 0.386499   
## jobretired 1.136e-01 1.332e-01 0.853 0.393568   
## jobself-employed -1.260e-01 1.328e-01 -0.949 0.342866   
## jobservices -2.371e-01 1.003e-01 -2.364 0.018083 \*   
## jobstudent 2.333e-01 1.372e-01 1.701 0.089013 .   
## jobtechnician 2.235e-02 8.132e-02 0.275 0.783479   
## jobunemployed -8.566e-02 1.560e-01 -0.549 0.582921   
## educationbasic.6y 1.912e-01 1.363e-01 1.403 0.160546   
## educationbasic.9y 1.619e-02 1.091e-01 0.148 0.882051   
## educationhigh.school 3.673e-02 1.068e-01 0.344 0.730980   
## educationilliterate 1.782e+00 8.429e-01 2.114 0.034525 \*   
## educationprofessional.course 6.658e-02 1.178e-01 0.565 0.571799   
## educationuniversity.degree 1.742e-01 1.076e-01 1.620 0.105242   
## defaultunknown -3.247e-01 7.851e-02 -4.136 3.54e-05 \*\*\*  
## contacttelephone -3.687e-01 8.205e-02 -4.494 7.00e-06 \*\*\*  
## monthaug 1.944e-01 1.203e-01 1.616 0.106058   
## monthdec 1.736e-02 2.519e-01 0.069 0.945049   
## monthjul 3.289e-01 1.114e-01 2.953 0.003144 \*\*   
## monthjun 3.867e-01 1.081e-01 3.578 0.000346 \*\*\*  
## monthmar 1.734e+00 1.435e-01 12.088 < 2e-16 \*\*\*  
## monthmay -6.341e-01 8.895e-02 -7.128 1.02e-12 \*\*\*  
## monthnov 5.511e-02 1.161e-01 0.475 0.634926   
## monthoct 4.115e-01 1.478e-01 2.785 0.005355 \*\*   
## monthsep -4.864e-02 1.613e-01 -0.302 0.762975   
## duration 2.521e-03 1.582e-04 15.936 < 2e-16 \*\*\*  
## campaign -4.562e-02 1.323e-02 -3.449 0.000562 \*\*\*  
## poutcomenonexistent 4.974e-01 7.632e-02 6.518 7.12e-11 \*\*\*  
## poutcomesuccess 1.129e+00 2.381e-01 4.739 2.14e-06 \*\*\*  
## cons\_price\_idx 4.940e-01 5.590e-02 8.838 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.772e-02 6.147e-03 9.389 < 2e-16 \*\*\*  
## euribor3m -7.122e-01 2.213e-02 -32.181 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.909e-01 6.747e-02 -2.830 0.004660 \*\*   
## Age\_Grp38-47 -2.900e-01 7.178e-02 -4.040 5.35e-05 \*\*\*  
## Age\_Grp47-55 -1.409e-01 8.038e-02 -1.753 0.079682 .   
## Age\_Grp>55 -3.846e-02 1.091e-01 -0.352 0.724479   
## prevly\_CntctdYes 8.272e-01 2.361e-01 3.503 0.000460 \*\*\*  
## duration\_group5-10min 1.102e+00 7.050e-02 15.630 < 2e-16 \*\*\*  
## duration\_group10+ min 1.994e+00 1.318e-01 15.136 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12492 on 30551 degrees of freedom  
## AIC: 12576  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))  
vif(simple.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 6.090679 10 1.094544  
## education 3.201880 6 1.101836  
## default 1.137064 1 1.066332  
## contact 2.000639 1 1.414439  
## month 5.285220 9 1.096908  
## duration 4.827347 1 2.197122  
## campaign 1.048985 1 1.024200  
## poutcome 10.432950 2 1.797222  
## cons\_price\_idx 2.437940 1 1.561391  
## cons\_conf\_idx 2.352976 1 1.533941  
## euribor3m 2.990674 1 1.729356  
## Age\_Grp 2.236906 4 1.105875  
## prevly\_Cntctd 9.261856 1 3.043330  
## duration\_group 5.166910 2 1.507675

#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)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4536 -0.3021 -0.1733 -0.1206 3.3331   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.555e+01 5.100e+00 -8.930 < 2e-16 \*\*\*  
## jobblue-collar -2.796e-01 9.329e-02 -2.997 0.002727 \*\*   
## jobentrepreneur -1.412e-01 1.435e-01 -0.984 0.325046   
## jobhousemaid -1.437e-01 1.748e-01 -0.822 0.411089   
## jobmanagement -8.590e-02 9.919e-02 -0.866 0.386499   
## jobretired 1.136e-01 1.332e-01 0.853 0.393568   
## jobself-employed -1.260e-01 1.328e-01 -0.949 0.342866   
## jobservices -2.371e-01 1.003e-01 -2.364 0.018083 \*   
## jobstudent 2.333e-01 1.372e-01 1.701 0.089013 .   
## jobtechnician 2.235e-02 8.132e-02 0.275 0.783479   
## jobunemployed -8.566e-02 1.560e-01 -0.549 0.582921   
## educationbasic.6y 1.912e-01 1.363e-01 1.403 0.160546   
## educationbasic.9y 1.619e-02 1.091e-01 0.148 0.882051   
## educationhigh.school 3.673e-02 1.068e-01 0.344 0.730980   
## educationilliterate 1.782e+00 8.429e-01 2.114 0.034525 \*   
## educationprofessional.course 6.658e-02 1.178e-01 0.565 0.571799   
## educationuniversity.degree 1.742e-01 1.076e-01 1.620 0.105242   
## defaultunknown -3.247e-01 7.851e-02 -4.136 3.54e-05 \*\*\*  
## contacttelephone -3.687e-01 8.205e-02 -4.494 7.00e-06 \*\*\*  
## monthaug 1.944e-01 1.203e-01 1.616 0.106058   
## monthdec 1.736e-02 2.519e-01 0.069 0.945049   
## monthjul 3.289e-01 1.114e-01 2.953 0.003144 \*\*   
## monthjun 3.867e-01 1.081e-01 3.578 0.000346 \*\*\*  
## monthmar 1.734e+00 1.435e-01 12.088 < 2e-16 \*\*\*  
## monthmay -6.341e-01 8.895e-02 -7.128 1.02e-12 \*\*\*  
## monthnov 5.511e-02 1.161e-01 0.475 0.634926   
## monthoct 4.115e-01 1.478e-01 2.785 0.005355 \*\*   
## monthsep -4.864e-02 1.613e-01 -0.302 0.762975   
## duration 2.521e-03 1.582e-04 15.936 < 2e-16 \*\*\*  
## campaign -4.562e-02 1.323e-02 -3.449 0.000562 \*\*\*  
## poutcomenonexistent 4.974e-01 7.632e-02 6.518 7.12e-11 \*\*\*  
## poutcomesuccess 1.129e+00 2.381e-01 4.739 2.14e-06 \*\*\*  
## cons\_price\_idx 4.940e-01 5.590e-02 8.838 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.772e-02 6.147e-03 9.389 < 2e-16 \*\*\*  
## euribor3m -7.122e-01 2.213e-02 -32.181 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.909e-01 6.747e-02 -2.830 0.004660 \*\*   
## Age\_Grp38-47 -2.900e-01 7.178e-02 -4.040 5.35e-05 \*\*\*  
## Age\_Grp47-55 -1.409e-01 8.038e-02 -1.753 0.079682 .   
## Age\_Grp>55 -3.846e-02 1.091e-01 -0.352 0.724479   
## prevly\_CntctdYes 8.272e-01 2.361e-01 3.503 0.000460 \*\*\*  
## duration\_group5-10min 1.102e+00 7.050e-02 15.630 < 2e-16 \*\*\*  
## duration\_group10+ min 1.994e+00 1.318e-01 15.136 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12492 on 30551 degrees of freedom  
## AIC: 12576  
##   
## Number of Fisher Scoring iterations: 6

exp(cbind("Odds ratio" = coef(simple.log), confint.default(simple.log, level = 0.95)))

## Odds ratio 2.5 % 97.5 %  
## (Intercept) 1.658223e-20 7.553626e-25 3.640242e-16  
## jobblue-collar 7.560969e-01 6.297496e-01 9.077934e-01  
## jobentrepreneur 8.683244e-01 6.554875e-01 1.150270e+00  
## jobhousemaid 8.661328e-01 6.148343e-01 1.220143e+00  
## jobmanagement 9.176893e-01 7.555540e-01 1.114617e+00  
## jobretired 1.120319e+00 8.629585e-01 1.454431e+00  
## jobself-employed 8.816401e-01 6.795860e-01 1.143769e+00  
## jobservices 7.888820e-01 6.480718e-01 9.602867e-01  
## jobstudent 1.262743e+00 9.650506e-01 1.652266e+00  
## jobtechnician 1.022599e+00 8.719301e-01 1.199302e+00  
## jobunemployed 9.179100e-01 6.761200e-01 1.246167e+00  
## educationbasic.6y 1.210743e+00 9.269355e-01 1.581447e+00  
## educationbasic.9y 1.016323e+00 8.206218e-01 1.258695e+00  
## educationhigh.school 1.037410e+00 8.414426e-01 1.279017e+00  
## educationilliterate 5.940524e+00 1.138530e+00 3.099594e+01  
## educationprofessional.course 1.068843e+00 8.485637e-01 1.346305e+00  
## educationuniversity.degree 1.190336e+00 9.640908e-01 1.469675e+00  
## defaultunknown 7.227403e-01 6.196633e-01 8.429634e-01  
## contacttelephone 6.916445e-01 5.889046e-01 8.123083e-01  
## monthaug 1.214560e+00 9.594936e-01 1.537432e+00  
## monthdec 1.017515e+00 6.210204e-01 1.667156e+00  
## monthjul 1.389496e+00 1.116988e+00 1.728487e+00  
## monthjun 1.472093e+00 1.191120e+00 1.819346e+00  
## monthmar 5.664909e+00 4.276279e+00 7.504466e+00  
## monthmay 5.304177e-01 4.455534e-01 6.314460e-01  
## monthnov 1.056661e+00 8.416479e-01 1.326603e+00  
## monthoct 1.509097e+00 1.129633e+00 2.016030e+00  
## monthsep 9.525283e-01 6.943908e-01 1.306628e+00  
## duration 1.002525e+00 1.002214e+00 1.002836e+00  
## campaign 9.554044e-01 9.309552e-01 9.804956e-01  
## poutcomenonexistent 1.644498e+00 1.416033e+00 1.909825e+00  
## poutcomesuccess 3.091373e+00 1.938442e+00 4.930037e+00  
## cons\_price\_idx 1.638933e+00 1.468848e+00 1.828713e+00  
## cons\_conf\_idx 1.059414e+00 1.046727e+00 1.072255e+00  
## euribor3m 4.905445e-01 4.697204e-01 5.122918e-01  
## Age\_Grp32-37 8.262114e-01 7.238754e-01 9.430148e-01  
## Age\_Grp38-47 7.482865e-01 6.500800e-01 8.613290e-01  
## Age\_Grp47-55 8.686031e-01 7.419962e-01 1.016813e+00  
## Age\_Grp>55 9.622735e-01 7.770161e-01 1.191700e+00  
## prevly\_CntctdYes 2.286866e+00 1.439602e+00 3.632779e+00  
## duration\_group5-10min 3.010208e+00 2.621703e+00 3.456285e+00  
## duration\_group10+ min 7.347602e+00 5.675279e+00 9.512706e+00

vif(simple.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 6.090679 10 1.094544  
## education 3.201880 6 1.101836  
## default 1.137064 1 1.066332  
## contact 2.000639 1 1.414439  
## month 5.285220 9 1.096908  
## duration 4.827347 1 2.197122  
## campaign 1.048985 1 1.024200  
## poutcome 10.432950 2 1.797222  
## cons\_price\_idx 2.437940 1 1.561391  
## cons\_conf\_idx 2.352976 1 1.533941  
## euribor3m 2.990674 1 1.729356  
## Age\_Grp 2.236906 4 1.105875  
## prevly\_Cntctd 9.261856 1 3.043330  
## duration\_group 5.166910 2 1.507675

#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")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 6604 464  
## yes 193 388  
##   
## Accuracy : 0.9141   
## 95% CI : (0.9076, 0.9203)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1.186e-13   
##   
## Kappa : 0.496   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.45540   
## Specificity : 0.97161   
## Pos Pred Value : 0.66781   
## Neg Pred Value : 0.93435   
## Prevalence : 0.11139   
## Detection Rate : 0.05073   
## Detection Prevalence : 0.07596   
## Balanced Accuracy : 0.71350   
##   
## 'Positive' Class : yes   
##

# STEP

## Feature Selection using stepwise

# Feature selection using step  
full.log<-glm(y~.,family="binomial",data=train)  
step.log<-full.log %>% stepAIC(trace=FALSE)  
summary(step.log)

##   
## Call:  
## glm(formula = y ~ job + default + contact + month + duration +   
## campaign + pdays + poutcome + emp\_var\_rate + cons\_price\_idx +   
## cons\_conf\_idx + euribor3m + Age\_Grp + duration\_group, family = "binomial",   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.5797 -0.2998 -0.1621 -0.1237 3.2283   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.819e+02 1.226e+01 -14.836 < 2e-16 \*\*\*  
## jobblue-collar -3.112e-01 7.604e-02 -4.092 4.28e-05 \*\*\*  
## jobentrepreneur -1.375e-01 1.430e-01 -0.962 0.336273   
## jobhousemaid -1.889e-01 1.680e-01 -1.124 0.260955   
## jobmanagement -5.819e-02 9.836e-02 -0.592 0.554102   
## jobretired 6.095e-02 1.294e-01 0.471 0.637654   
## jobself-employed -1.178e-01 1.322e-01 -0.892 0.372624   
## jobservices -2.844e-01 9.574e-02 -2.971 0.002970 \*\*   
## jobstudent 1.487e-01 1.347e-01 1.104 0.269636   
## jobtechnician 1.527e-02 7.291e-02 0.209 0.834099   
## jobunemployed -1.234e-01 1.553e-01 -0.794 0.427085   
## defaultunknown -3.014e-01 7.813e-02 -3.858 0.000114 \*\*\*  
## contacttelephone -6.424e-01 8.784e-02 -7.314 2.60e-13 \*\*\*  
## monthaug 7.749e-01 1.282e-01 6.046 1.49e-09 \*\*\*  
## monthdec -1.442e-01 2.531e-01 -0.570 0.568729   
## monthjul 1.348e-01 1.130e-01 1.193 0.232999   
## monthjun -4.174e-01 1.251e-01 -3.336 0.000848 \*\*\*  
## monthmar 2.146e+00 1.402e-01 15.305 < 2e-16 \*\*\*  
## monthmay -4.678e-01 9.094e-02 -5.144 2.69e-07 \*\*\*  
## monthnov -5.587e-01 1.267e-01 -4.411 1.03e-05 \*\*\*  
## monthoct 3.599e-02 1.501e-01 0.240 0.810493   
## monthsep 9.181e-02 1.613e-01 0.569 0.569354   
## duration 2.483e-03 1.576e-04 15.752 < 2e-16 \*\*\*  
## campaign -3.553e-02 1.308e-02 -2.716 0.006599 \*\*   
## pdays -6.807e-04 2.421e-04 -2.812 0.004926 \*\*   
## poutcomenonexistent 4.614e-01 7.670e-02 6.016 1.79e-09 \*\*\*  
## poutcomesuccess 1.250e+00 2.426e-01 5.153 2.56e-07 \*\*\*  
## emp\_var\_rate -1.677e+00 1.349e-01 -12.437 < 2e-16 \*\*\*  
## cons\_price\_idx 1.900e+00 1.271e-01 14.950 < 2e-16 \*\*\*  
## cons\_conf\_idx 2.394e-02 6.604e-03 3.624 0.000290 \*\*\*  
## euribor3m 5.332e-01 1.023e-01 5.212 1.87e-07 \*\*\*  
## Age\_Grp32-37 -1.755e-01 6.773e-02 -2.592 0.009555 \*\*   
## Age\_Grp38-47 -3.095e-01 7.190e-02 -4.304 1.68e-05 \*\*\*  
## Age\_Grp47-55 -1.560e-01 8.050e-02 -1.938 0.052650 .   
## Age\_Grp>55 -7.817e-02 1.085e-01 -0.720 0.471281   
## duration\_group5-10min 1.135e+00 7.120e-02 15.937 < 2e-16 \*\*\*  
## duration\_group10+ min 2.074e+00 1.321e-01 15.701 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12350 on 30556 degrees of freedom  
## AIC: 12424  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 2.399399 10 1.044733  
## default 1.128309 1 1.062219  
## contact 2.320168 1 1.523210  
## month 14.501164 9 1.160172  
## duration 4.835885 1 2.199065  
## campaign 1.052089 1 1.025714  
## pdays 9.463933 1 3.076351  
## poutcome 10.711233 2 1.809089  
## emp\_var\_rate 96.727663 1 9.835022  
## cons\_price\_idx 12.381847 1 3.518785  
## cons\_conf\_idx 2.705182 1 1.644744  
## euribor3m 63.436080 1 7.964677  
## Age\_Grp 2.173642 4 1.101916  
## duration\_group 5.243071 2 1.513200

#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)

##   
## Call:  
## glm(formula = y ~ job + default + contact + month + duration +   
## campaign + pdays + poutcome + cons\_price\_idx + cons\_conf\_idx +   
## euribor3m + nr\_employed + Age\_Grp + duration\_group, family = "binomial",   
## data = train\_step)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4922 -0.3000 -0.1690 -0.1231 3.2576   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.381e+02 2.859e+01 4.830 1.36e-06 \*\*\*  
## jobblue-collar -3.189e-01 7.593e-02 -4.200 2.67e-05 \*\*\*  
## jobentrepreneur -1.247e-01 1.425e-01 -0.875 0.381496   
## jobhousemaid -2.027e-01 1.679e-01 -1.207 0.227262   
## jobmanagement -5.481e-02 9.804e-02 -0.559 0.576129   
## jobretired 7.680e-02 1.289e-01 0.596 0.551462   
## jobself-employed -1.141e-01 1.319e-01 -0.866 0.386742   
## jobservices -2.906e-01 9.532e-02 -3.048 0.002300 \*\*   
## jobstudent 1.762e-01 1.345e-01 1.310 0.190339   
## jobtechnician -1.139e-02 7.254e-02 -0.157 0.875220   
## jobunemployed -1.385e-01 1.545e-01 -0.897 0.369825   
## defaultunknown -3.149e-01 7.810e-02 -4.032 5.53e-05 \*\*\*  
## contacttelephone -3.406e-01 8.175e-02 -4.166 3.09e-05 \*\*\*  
## monthaug 3.566e-02 1.222e-01 0.292 0.770420   
## monthdec -4.789e-01 2.627e-01 -1.823 0.068351 .   
## monthjul 2.493e-01 1.114e-01 2.238 0.025253 \*   
## monthjun 5.689e-01 1.102e-01 5.161 2.46e-07 \*\*\*  
## monthmar 1.350e+00 1.520e-01 8.884 < 2e-16 \*\*\*  
## monthmay -7.790e-01 9.198e-02 -8.469 < 2e-16 \*\*\*  
## monthnov -4.760e-01 1.423e-01 -3.345 0.000824 \*\*\*  
## monthoct -2.740e-01 1.813e-01 -1.511 0.130728   
## monthsep -7.500e-01 1.942e-01 -3.862 0.000112 \*\*\*  
## duration 2.511e-03 1.579e-04 15.902 < 2e-16 \*\*\*  
## campaign -4.126e-02 1.319e-02 -3.127 0.001765 \*\*   
## pdays -7.986e-04 2.392e-04 -3.338 0.000843 \*\*\*  
## poutcomenonexistent 4.740e-01 7.643e-02 6.201 5.61e-10 \*\*\*  
## poutcomesuccess 1.151e+00 2.397e-01 4.800 1.59e-06 \*\*\*  
## cons\_price\_idx -4.634e-01 1.575e-01 -2.943 0.003250 \*\*   
## cons\_conf\_idx 1.382e-02 9.102e-03 1.518 0.128939   
## euribor3m 2.707e-01 1.530e-01 1.770 0.076766 .   
## nr\_employed -1.903e-02 2.932e-03 -6.489 8.63e-11 \*\*\*  
## Age\_Grp32-37 -1.813e-01 6.751e-02 -2.686 0.007227 \*\*   
## Age\_Grp38-47 -2.887e-01 7.156e-02 -4.035 5.47e-05 \*\*\*  
## Age\_Grp47-55 -1.529e-01 8.013e-02 -1.909 0.056306 .   
## Age\_Grp>55 -5.383e-02 1.081e-01 -0.498 0.618364   
## duration\_group5-10min 1.115e+00 7.079e-02 15.754 < 2e-16 \*\*\*  
## duration\_group10+ min 2.024e+00 1.318e-01 15.353 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12460 on 30556 degrees of freedom  
## AIC: 12534  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 2.406441 10 1.044886  
## default 1.125471 1 1.060882  
## contact 1.991970 1 1.411372  
## month 14.325935 9 1.159389  
## duration 4.838433 1 2.199644  
## campaign 1.050518 1 1.024948  
## pdays 9.299780 1 3.049554  
## poutcome 10.513278 2 1.800672  
## cons\_price\_idx 18.804402 1 4.336404  
## cons\_conf\_idx 5.163621 1 2.272360  
## euribor3m 142.731932 1 11.947047  
## nr\_employed 114.114514 1 10.682440  
## Age\_Grp 2.167942 4 1.101554  
## duration\_group 5.209427 2 1.510767

euribor and nr\_employed are both statistically significant in the model but have high VIFs. Removing nr\_employed

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)

##   
## Call:  
## glm(formula = y ~ job + default + contact + month + duration +   
## campaign + pdays + poutcome + cons\_price\_idx + cons\_conf\_idx +   
## euribor3m + Age\_Grp + duration\_group, family = "binomial",   
## data = train\_step\_2)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4750 -0.3025 -0.1736 -0.1211 3.3220   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.449e+01 5.118e+00 -8.694 < 2e-16 \*\*\*  
## jobblue-collar -3.438e-01 7.573e-02 -4.540 5.63e-06 \*\*\*  
## jobentrepreneur -1.441e-01 1.424e-01 -1.012 0.311339   
## jobhousemaid -2.050e-01 1.682e-01 -1.219 0.222772   
## jobmanagement -5.485e-02 9.793e-02 -0.560 0.575389   
## jobretired 7.707e-02 1.290e-01 0.597 0.550254   
## jobself-employed -1.105e-01 1.318e-01 -0.839 0.401556   
## jobservices -2.992e-01 9.517e-02 -3.143 0.001670 \*\*   
## jobstudent 1.713e-01 1.344e-01 1.275 0.202435   
## jobtechnician -3.929e-03 7.239e-02 -0.054 0.956719   
## jobunemployed -1.311e-01 1.541e-01 -0.851 0.394948   
## defaultunknown -3.280e-01 7.801e-02 -4.205 2.62e-05 \*\*\*  
## contacttelephone -3.698e-01 8.203e-02 -4.508 6.53e-06 \*\*\*  
## monthaug 2.073e-01 1.201e-01 1.726 0.084260 .   
## monthdec 2.063e-02 2.515e-01 0.082 0.934617   
## monthjul 3.271e-01 1.113e-01 2.938 0.003298 \*\*   
## monthjun 3.890e-01 1.080e-01 3.603 0.000315 \*\*\*  
## monthmar 1.740e+00 1.434e-01 12.138 < 2e-16 \*\*\*  
## monthmay -6.397e-01 8.886e-02 -7.199 6.07e-13 \*\*\*  
## monthnov 6.126e-02 1.160e-01 0.528 0.597358   
## monthoct 4.113e-01 1.477e-01 2.784 0.005369 \*\*   
## monthsep -4.368e-02 1.612e-01 -0.271 0.786403   
## duration 2.522e-03 1.581e-04 15.951 < 2e-16 \*\*\*  
## campaign -4.496e-02 1.321e-02 -3.404 0.000663 \*\*\*  
## pdays -8.301e-04 2.387e-04 -3.478 0.000506 \*\*\*  
## poutcomenonexistent 5.006e-01 7.626e-02 6.564 5.23e-11 \*\*\*  
## poutcomesuccess 1.135e+00 2.391e-01 4.749 2.04e-06 \*\*\*  
## cons\_price\_idx 4.930e-01 5.582e-02 8.833 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.793e-02 6.141e-03 9.434 < 2e-16 \*\*\*  
## euribor3m -7.137e-01 2.211e-02 -32.275 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.869e-01 6.738e-02 -2.774 0.005533 \*\*   
## Age\_Grp38-47 -2.936e-01 7.145e-02 -4.109 3.97e-05 \*\*\*  
## Age\_Grp47-55 -1.489e-01 7.996e-02 -1.862 0.062656 .   
## Age\_Grp>55 -6.490e-02 1.081e-01 -0.601 0.548131   
## duration\_group5-10min 1.099e+00 7.045e-02 15.605 < 2e-16 \*\*\*  
## duration\_group10+ min 1.990e+00 1.316e-01 15.123 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12503 on 30557 degrees of freedom  
## AIC: 12575  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 2.392047 10 1.044572  
## default 1.122391 1 1.059429  
## contact 2.001812 1 1.414854  
## month 5.216081 9 1.096106  
## duration 4.822348 1 2.195984  
## campaign 1.048289 1 1.023860  
## pdays 9.339550 1 3.056068  
## poutcome 10.518220 2 1.800883  
## cons\_price\_idx 2.433382 1 1.559930  
## cons\_conf\_idx 2.349901 1 1.532939  
## euribor3m 2.987426 1 1.728417  
## Age\_Grp 2.164950 4 1.101364  
## duration\_group 5.157382 2 1.506979

poutcome has high VIF, removing poutcome

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)

##   
## Call:  
## glm(formula = y ~ job + default + contact + month + duration +   
## campaign + pdays + previous + cons\_price\_idx + cons\_conf\_idx +   
## euribor3m + Age\_Grp + prevly\_Cntctd + duration\_group, family = "binomial",   
## data = train\_step\_3)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4894 -0.3013 -0.1741 -0.1216 3.3233   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 8.052e+00 1.896e+01 0.425 0.671058   
## jobblue-collar -3.460e-01 7.571e-02 -4.570 4.87e-06 \*\*\*  
## jobentrepreneur -1.395e-01 1.419e-01 -0.983 0.325686   
## jobhousemaid -2.020e-01 1.678e-01 -1.204 0.228707   
## jobmanagement -5.561e-02 9.779e-02 -0.569 0.569570   
## jobretired 7.486e-02 1.292e-01 0.580 0.562176   
## jobself-employed -1.182e-01 1.317e-01 -0.897 0.369490   
## jobservices -3.058e-01 9.516e-02 -3.214 0.001311 \*\*   
## jobstudent 1.747e-01 1.339e-01 1.304 0.192079   
## jobtechnician -4.223e-03 7.230e-02 -0.058 0.953422   
## jobunemployed -1.308e-01 1.538e-01 -0.851 0.394801   
## defaultunknown -3.269e-01 7.797e-02 -4.193 2.75e-05 \*\*\*  
## contacttelephone -3.735e-01 8.211e-02 -4.549 5.39e-06 \*\*\*  
## monthaug 2.439e-01 1.198e-01 2.035 0.041816 \*   
## monthdec 6.262e-02 2.510e-01 0.250 0.802960   
## monthjul 3.418e-01 1.111e-01 3.076 0.002096 \*\*   
## monthjun 3.894e-01 1.079e-01 3.609 0.000308 \*\*\*  
## monthmar 1.757e+00 1.437e-01 12.231 < 2e-16 \*\*\*  
## monthmay -6.282e-01 8.880e-02 -7.075 1.50e-12 \*\*\*  
## monthnov 7.327e-02 1.161e-01 0.631 0.527815   
## monthoct 4.253e-01 1.472e-01 2.889 0.003860 \*\*   
## monthsep -9.541e-03 1.605e-01 -0.059 0.952594   
## duration 2.533e-03 1.582e-04 16.011 < 2e-16 \*\*\*  
## campaign -4.534e-02 1.321e-02 -3.432 0.000599 \*\*\*  
## pdays -5.782e-02 1.886e-02 -3.065 0.002176 \*\*   
## previous -2.798e-01 4.760e-02 -5.878 4.16e-09 \*\*\*  
## cons\_price\_idx 5.450e-01 5.743e-02 9.490 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.764e-02 6.129e-03 9.404 < 2e-16 \*\*\*  
## euribor3m -7.156e-01 2.234e-02 -32.027 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.866e-01 6.729e-02 -2.773 0.005547 \*\*   
## Age\_Grp38-47 -2.965e-01 7.141e-02 -4.151 3.30e-05 \*\*\*  
## Age\_Grp47-55 -1.540e-01 7.990e-02 -1.928 0.053897 .   
## Age\_Grp>55 -7.320e-02 1.081e-01 -0.677 0.498446   
## prevly\_CntctdYes -5.559e+01 1.873e+01 -2.968 0.002998 \*\*   
## duration\_group5-10min 1.096e+00 7.039e-02 15.569 < 2e-16 \*\*\*  
## duration\_group10+ min 1.978e+00 1.316e-01 15.026 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12527 on 30557 degrees of freedom  
## AIC: 12599  
##   
## Number of Fisher Scoring iterations: 6

#exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))  
vif(step.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 2.397584 10 1.044693  
## default 1.122495 1 1.059479  
## contact 2.008049 1 1.417056  
## month 5.242353 9 1.096412  
## duration 4.828431 1 2.197369  
## campaign 1.048006 1 1.023722  
## pdays 58968.610289 1 242.834533  
## previous 2.043043 1 1.429350  
## cons\_price\_idx 2.585390 1 1.607915  
## cons\_conf\_idx 2.339167 1 1.529433  
## euribor3m 3.055895 1 1.748112  
## Age\_Grp 2.168712 4 1.101603  
## prevly\_Cntctd 58955.177392 1 242.806873  
## duration\_group 5.160517 2 1.507208

pdays and prevly\_Cntcted now have very large VIF. Let’s remove pdays as it is likely to be more misleading versus our categorical version

#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)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.4844 -0.3014 -0.1744 -0.1217 3.3227   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.796e+01 5.196e+00 -9.230 < 2e-16 \*\*\*  
## jobblue-collar -3.462e-01 7.569e-02 -4.574 4.79e-06 \*\*\*  
## jobentrepreneur -1.376e-01 1.418e-01 -0.970 0.331938   
## jobhousemaid -2.015e-01 1.678e-01 -1.201 0.229808   
## jobmanagement -5.121e-02 9.769e-02 -0.524 0.600094   
## jobretired 8.145e-02 1.289e-01 0.632 0.527548   
## jobself-employed -1.135e-01 1.315e-01 -0.863 0.387958   
## jobservices -3.002e-01 9.506e-02 -3.158 0.001587 \*\*   
## jobstudent 1.756e-01 1.340e-01 1.311 0.189806   
## jobtechnician -5.637e-03 7.231e-02 -0.078 0.937871   
## jobunemployed -1.231e-01 1.535e-01 -0.802 0.422637   
## defaultunknown -3.259e-01 7.793e-02 -4.182 2.88e-05 \*\*\*  
## contacttelephone -3.700e-01 8.206e-02 -4.510 6.50e-06 \*\*\*  
## monthaug 2.347e-01 1.199e-01 1.958 0.050196 .   
## monthdec 5.034e-02 2.511e-01 0.201 0.841078   
## monthjul 3.338e-01 1.111e-01 3.003 0.002671 \*\*   
## monthjun 3.956e-01 1.078e-01 3.669 0.000244 \*\*\*  
## monthmar 1.748e+00 1.437e-01 12.160 < 2e-16 \*\*\*  
## monthmay -6.341e-01 8.876e-02 -7.144 9.04e-13 \*\*\*  
## monthnov 6.965e-02 1.160e-01 0.601 0.548114   
## monthoct 4.150e-01 1.472e-01 2.819 0.004823 \*\*   
## monthsep -7.822e-03 1.604e-01 -0.049 0.961108   
## duration 2.534e-03 1.582e-04 16.019 < 2e-16 \*\*\*  
## campaign -4.551e-02 1.321e-02 -3.444 0.000574 \*\*\*  
## previous -2.720e-01 4.753e-02 -5.723 1.05e-08 \*\*\*  
## cons\_price\_idx 5.261e-01 5.706e-02 9.221 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.780e-02 6.130e-03 9.429 < 2e-16 \*\*\*  
## euribor3m -7.107e-01 2.229e-02 -31.885 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.871e-01 6.728e-02 -2.780 0.005430 \*\*   
## Age\_Grp38-47 -2.968e-01 7.141e-02 -4.156 3.24e-05 \*\*\*  
## Age\_Grp47-55 -1.522e-01 7.985e-02 -1.906 0.056700 .   
## Age\_Grp>55 -7.245e-02 1.080e-01 -0.671 0.502157   
## prevly\_CntctdYes 1.823e+00 1.040e-01 17.522 < 2e-16 \*\*\*  
## duration\_group5-10min 1.093e+00 7.035e-02 15.535 < 2e-16 \*\*\*  
## duration\_group10+ min 1.974e+00 1.316e-01 14.998 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12537 on 30558 degrees of freedom  
## AIC: 12607  
##   
## Number of Fisher Scoring iterations: 6

exp(cbind("Odds ratio" = coef(step.log), confint.default(step.log, level = 0.95)))

## Odds ratio 2.5 % 97.5 %  
## (Intercept) 1.486816e-21 5.619449e-26 3.933874e-17  
## jobblue-collar 7.073685e-01 6.098433e-01 8.204898e-01  
## jobentrepreneur 8.714654e-01 6.600057e-01 1.150675e+00  
## jobhousemaid 8.175043e-01 5.883847e-01 1.135844e+00  
## jobmanagement 9.500750e-01 7.845235e-01 1.150562e+00  
## jobretired 1.084859e+00 8.426183e-01 1.396739e+00  
## jobself-employed 8.926634e-01 6.898213e-01 1.155151e+00  
## jobservices 7.406533e-01 6.147517e-01 8.923396e-01  
## jobstudent 1.192006e+00 9.167574e-01 1.549895e+00  
## jobtechnician 9.943792e-01 8.629751e-01 1.145792e+00  
## jobunemployed 8.842043e-01 6.545034e-01 1.194520e+00  
## defaultunknown 7.218532e-01 6.196072e-01 8.409716e-01  
## contacttelephone 6.907091e-01 5.880970e-01 8.112251e-01  
## monthaug 1.264588e+00 9.997991e-01 1.599504e+00  
## monthdec 1.051629e+00 6.429310e-01 1.720127e+00  
## monthjul 1.396264e+00 1.122948e+00 1.736103e+00  
## monthjun 1.485329e+00 1.202348e+00 1.834911e+00  
## monthmar 5.741521e+00 4.332014e+00 7.609639e+00  
## monthmay 5.304126e-01 4.457234e-01 6.311931e-01  
## monthnov 1.072134e+00 8.541504e-01 1.345747e+00  
## monthoct 1.514426e+00 1.134774e+00 2.021094e+00  
## monthsep 9.922084e-01 7.245362e-01 1.358769e+00  
## duration 1.002537e+00 1.002227e+00 1.002848e+00  
## campaign 9.555141e-01 9.310837e-01 9.805856e-01  
## previous 7.618694e-01 6.941063e-01 8.362479e-01  
## cons\_price\_idx 1.692394e+00 1.513325e+00 1.892652e+00  
## cons\_conf\_idx 1.059500e+00 1.046847e+00 1.072305e+00  
## euribor3m 4.913206e-01 4.703196e-01 5.132594e-01  
## Age\_Grp32-37 8.293998e-01 7.269379e-01 9.463038e-01  
## Age\_Grp38-47 7.432155e-01 6.461396e-01 8.548760e-01  
## Age\_Grp47-55 8.588405e-01 7.344125e-01 1.004350e+00  
## Age\_Grp>55 9.301098e-01 7.527272e-01 1.149293e+00  
## prevly\_CntctdYes 6.190231e+00 5.048317e+00 7.590443e+00  
## duration\_group5-10min 2.983055e+00 2.598828e+00 3.424089e+00  
## duration\_group10+ min 7.200307e+00 5.563033e+00 9.319452e+00

vif(step.log)

## GVIF Df GVIF^(1/(2\*Df))  
## job 2.391848 10 1.044568  
## default 1.122330 1 1.059401  
## contact 2.006911 1 1.416655  
## month 5.233321 9 1.096307  
## duration 4.831485 1 2.198064  
## campaign 1.048058 1 1.023747  
## previous 2.035705 1 1.426781  
## cons\_price\_idx 2.555074 1 1.598460  
## cons\_conf\_idx 2.345479 1 1.531496  
## euribor3m 3.041457 1 1.743977  
## Age\_Grp 2.166411 4 1.101457  
## prevly\_Cntctd 1.820919 1 1.349414  
## duration\_group 5.162044 2 1.507320

Keep as final model. All are significant and VIFS look good - indicates we should be OK to keep both duration and duration group.

#Predicting using step   
fit.pred.step<-predict(step.log,newdata=test,type="response")  
test$y[1:15]

## [1] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes  
## Levels: no yes

fit.pred.step[1:15]

## 1 18 20 33 44 52 53   
## 0.47209884 0.14373633 0.70226588 0.18703178 0.02098207 0.18941824 0.04853144   
## 60 66 70 73 74 76 77   
## 0.02232532 0.16818365 0.12144409 0.15817724 0.27592340 0.11789706 0.15779946   
## 81   
## 0.31397049

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")

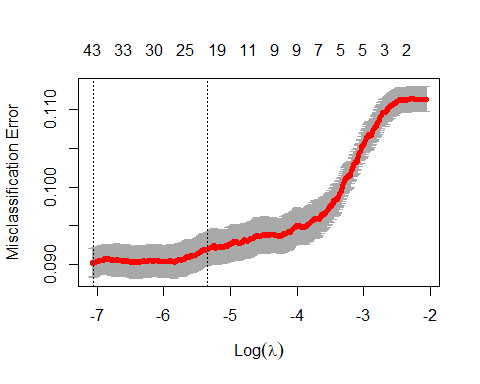
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 6609 473  
## yes 188 379  
##   
## Accuracy : 0.9136   
## 95% CI : (0.9071, 0.9198)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 3.77e-13   
##   
## Kappa : 0.4887   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.44484   
## Specificity : 0.97234   
## Pos Pred Value : 0.66843   
## Neg Pred Value : 0.93321   
## Prevalence : 0.11139   
## Detection Rate : 0.04955   
## Detection Prevalence : 0.07413   
## Balanced Accuracy : 0.70859   
##   
## 'Positive' Class : yes   
##

#Acc 91%, Sens. 44%, Spec. 97%

# LASSO

## Feature selection using lasso

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")

## 56 x 1 sparse Matrix of class "dgCMatrix"  
## 1  
## (Intercept) -4.207121e+01  
## (Intercept) .   
## age .   
## jobblue-collar -2.021790e-01  
## jobentrepreneur -1.633144e-02  
## jobhousemaid -1.480863e-02  
## jobmanagement .   
## jobretired 1.154413e-01  
## jobself-employed -1.198252e-02  
## jobservices -1.466821e-01  
## jobstudent 2.341095e-01  
## jobtechnician 3.873314e-02  
## jobunemployed .   
## maritalmarried .   
## maritalsingle 3.694940e-02  
## educationbasic.6y 2.719654e-02  
## educationbasic.9y -2.131391e-02  
## educationhigh.school .   
## educationilliterate 1.257055e+00  
## educationprofessional.course .   
## educationuniversity.degree 1.000798e-01  
## defaultunknown -2.729949e-01  
## housingyes .   
## loanyes .   
## contacttelephone -3.621510e-01  
## monthaug 3.138102e-01  
## monthdec -7.597149e-02  
## monthjul 1.880086e-01  
## monthjun 6.402111e-02  
## monthmar 1.685199e+00  
## monthmay -6.424520e-01  
## monthnov -2.876171e-01  
## monthoct 6.945301e-02  
## monthsep -1.093432e-01  
## day\_of\_weekmon -1.036189e-01  
## day\_of\_weekthu .   
## day\_of\_weektue 8.624198e-03  
## day\_of\_weekwed 2.008579e-02  
## duration 2.521093e-03  
## campaign -2.739075e-02  
## pdays -7.487669e-04  
## previous .   
## poutcomenonexistent 4.012496e-01  
## poutcomesuccess 1.112954e+00  
## emp\_var\_rate -6.018043e-01  
## cons\_price\_idx 6.946629e-01  
## cons\_conf\_idx 2.896983e-02  
## euribor3m .   
## nr\_employed -4.849376e-03  
## Age\_Grp32-37 -9.618446e-02  
## Age\_Grp38-47 -2.050658e-01  
## Age\_Grp47-55 -5.124525e-02  
## Age\_Grp>55 .   
## prevly\_CntctdYes 5.751142e-04  
## duration\_group5-10min 1.038409e+00  
## duration\_group10+ min 1.921510e+00

#CV misclassification error rate is little below .1  
  
print("CV Error Rate:")

## [1] "CV Error Rate:"

cvfit$cvm[which(cvfit$lambda==cvfit$lambda.min)]

## [1] 0.09021672

#"CV Error Rate:"  
#0.09021672  
  
#Optimal penalty  
print("Penalty Value:")

## [1] "Penalty Value:"

cvfit$lambda.min

## [1] 0.0008648178

#"Penalty Value:"  
#0.0008648178  
  
finalmodel<-glmnet(dat.train.x, dat.train.y, family = "binomial",lambda=cvfit$lambda.min)  
finalmodel$call

## glmnet(x = dat.train.x, y = dat.train.y, family = "binomial",   
## lambda = cvfit$lambda.min)

finalmodel

##   
## Call: glmnet(x = dat.train.x, y = dat.train.y, family = "binomial", lambda = cvfit$lambda.min)   
##   
## Df %Dev Lambda  
## 1 41 42.01 0.0008648

dat.test.x<-model.matrix(y~.,test)  
fit.pred.lasso <- predict(finalmodel, newx = dat.test.x, type = "response")  
test$y[1:15]

## [1] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes  
## Levels: no yes

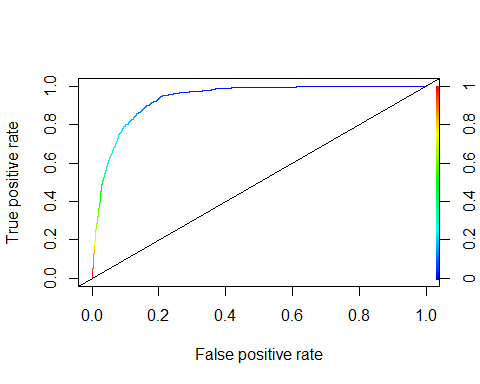
fit.pred.lasso[1:15]

## [1] 0.51648436 0.17875067 0.75496908 0.24614855 0.02907491 0.21843403  
## [7] 0.05749881 0.03117035 0.21369006 0.16038986 0.17173557 0.35568869  
## [13] 0.15018552 0.20535766 0.38774722

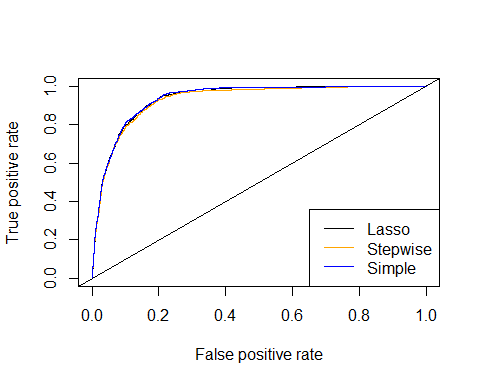
#confusion matrix at 0.5 cutoff  
class.lasso1<-factor(ifelse(fit.pred.lasso>0.5,"yes","no"),levels=c("no","yes"))  
# use caret and compute a confusion matrix  
confusionMatrix(class.lasso1,test$y, positive = "yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 6615 464  
## yes 182 388  
##   
## Accuracy : 0.9155   
## 95% CI : (0.9091, 0.9217)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 4.309e-15   
##   
## Kappa : 0.5012   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.45540   
## Specificity : 0.97322   
## Pos Pred Value : 0.68070   
## Neg Pred Value : 0.93445   
## Prevalence : 0.11139   
## Detection Rate : 0.05073   
## Detection Prevalence : 0.07452   
## Balanced Accuracy : 0.71431   
##   
## 'Positive' Class : 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",print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(1.2,1.2), avg="threshold", lwd=3)  
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", print.cutoffs.at = c("0.1", "0.15", "0.2"))  
  
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", print.cutoffs.at = c(0.1, 0.15, 0.2))  
  
#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)



Tried to find optimal cut-off using other packages, could not get to work

#finding optimal cut-off for step model  
perf.acc<-performance(results.step,"acc") #find list of accuracies  
perf.fpr<-performance(results.step,"fpr") # find list of fp rates  
perf.fnr<-performance(results.step,"fnr") # find list of fn rates  
  
acc.rocr<-max(perf.acc@y.values[[1]]) # accuracy using rocr  
  
#find cutoff list for accuracies  
cutoff.list.acc <- unlist(perf.acc@y.values[[1]])  
  
#find optimal cutoff point for accuracy  
optimal.cutoff.acc<-cutoff.list.acc[which.max(perf.acc@y.values[[1]])]  
  
#find optimal cutoff fpr, as numeric because a list is returned  
optimal.cutoff.fpr<-which(perf.fpr@x.values[[1]]==as.numeric(optimal.cutoff.acc))  
  
# find cutoff list for fpr  
cutoff.list.fpr <- unlist(perf.fpr@y.values[[1]])  
# find fpr using rocr  
fpr.rocr<-cutoff.list.fpr[as.numeric(optimal.cutoff.fpr)]

#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")

## [1] "Confusion matrix for LASSO"

conf.lasso

##   
## class.lasso no yes  
## no 6615 464  
## yes 182 388

#Confusion Matrix for step  
conf.step <- table(class.step, test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.step

##   
## class.step no yes  
## no 6609 473  
## yes 188 379

#Confusion Matrix for simple  
conf.simple <- table(class.simple, test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.simple

##   
## class.simple no yes  
## no 6604 464  
## yes 193 388

#Accuracy of LASSO and Stepwise  
print("Overall accuracy for LASSO and Stepwise respectively")

## [1] "Overall accuracy for LASSO and Stepwise respectively"

sum(diag(conf.lasso)) / sum(conf.lasso)

## [1] 0.9155445

sum(diag(conf.step)) / sum(conf.step)

## [1] 0.9135835

print("Alternative calculations of accuracy")

## [1] "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"))

## Confusion Matrix for Lasso

conf.lasso<-table(class.lasso,test$y)  
print("Confusion matrix for LASSO")

## [1] "Confusion matrix for LASSO"

conf.lasso

##   
## class.lasso no yes  
## no 5642 83  
## yes 1155 769

## Confusion Matrix for step

conf.step<-table(class.step,test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.step

##   
## class.step no yes  
## no 5611 88  
## yes 1186 764

## Confusion Matrix for simple

conf.simple<-table(class.simple,test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.simple

##   
## class.simple no yes  
## no 5629 87  
## yes 1168 765

## Accuracy of LASSO and Stepwise

print("Overall accuracy for LASSO and Stepwise respectively")

## [1] "Overall accuracy for LASSO and Stepwise respectively"

sum(diag(conf.lasso))/sum(conf.lasso)

## [1] 0.8381488

sum(diag(conf.step))/sum(conf.step)

## [1] 0.8334423

print("Alternative calculations of accuracy")

## [1] "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)

Confusion Matrix for cut off =0.1

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"))

Confusion Matrix for Lasso at 0.15 cut-off

conf.lasso<-table(class.lasso,test$y)  
print("Confusion matrix for LASSO")

## [1] "Confusion matrix for LASSO"

conf.lasso

##   
## class.lasso no yes  
## no 5896 127  
## yes 901 725

Confusion Matrix for step at 0.15 cut-off

conf.step<-table(class.step,test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.step

##   
## class.step no yes  
## no 5924 147  
## yes 873 705

confusionMatrix(conf.step, positive = "yes")

## Confusion Matrix and Statistics  
##   
##   
## class.step no yes  
## no 5924 147  
## yes 873 705  
##   
## Accuracy : 0.8666   
## 95% CI : (0.8588, 0.8742)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.5093   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.82746   
## Specificity : 0.87156   
## Pos Pred Value : 0.44677   
## Neg Pred Value : 0.97579   
## Prevalence : 0.11139   
## Detection Rate : 0.09217   
## Detection Prevalence : 0.20630   
## Balanced Accuracy : 0.84951   
##   
## 'Positive' Class : yes   
##

Confusion Matrix for simple at 0.15 cut-off

conf.simple<-table(class.simple,test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.simple

##   
## class.simple no yes  
## no 5924 146  
## yes 873 706

## Accuracy of LASSO and Stepwise

print("Overall accuracy for LASSO and Stepwise respectively")

## [1] "Overall accuracy for LASSO and Stepwise respectively"

sum(diag(conf.lasso))/sum(conf.lasso)

## [1] 0.8656033

sum(diag(conf.step))/sum(conf.step)

## [1] 0.8666492

print("Alternative calculations of accuracy")

## [1] "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)

Confusion Matrix for cut off =0.15

lasso\_0.15<-confusionMatrix(conf.lasso, positive = "yes")  
step\_0.15<-confusionMatrix(conf.step, positive = "yes")  
simple\_0.15<-confusionMatrix(conf.simple, positive = "yes")

Checking 0.2 cut off

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")

## [1] "Confusion matrix for LASSO"

conf.lasso

##   
## class.lasso no yes  
## no 6139 178  
## yes 658 674

#Confusion Matrix for step  
conf.step <- table(class.step, test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.step

##   
## class.step no yes  
## no 6168 199  
## yes 629 653

#Confusion Matrix for simple  
conf.simple <- table(class.simple, test$y)  
print("Confusion matrix for Stepwise")

## [1] "Confusion matrix for Stepwise"

conf.simple

##   
## class.simple no yes  
## no 6158 197  
## yes 639 655

#Accuracy of LASSO and Stepwise  
print("Overall accuracy for LASSO and Stepwise respectively")

## [1] "Overall accuracy for LASSO and Stepwise respectively"

sum(diag(conf.lasso)) / sum(conf.lasso)

## [1] 0.8907047

sum(diag(conf.step)) / sum(conf.step)

## [1] 0.8917506

#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

## CutOff Simple\_Sensitivty Sensitivity\_step$Step\_Sensitivity  
## 1 0.1 0.8978873 0.8967136  
## 2 0.15 0.8286385 0.8274648  
## 3 0.2 0.9059879 0.9074592  
## 4 0.5 0.9716051 0.9723407  
## Sensitivity\_lasso$LASSO\_Sensitivity  
## 1 0.9025822  
## 2 0.8509390  
## 3 0.9031926  
## 4 0.9732235

Specificity

## CutOff Simple\_Specificity Specificity\_step$Step\_Specificity  
## 1 0.1 0.8281595 0.8255113  
## 2 0.15 0.8715610 0.8715610  
## 3 0.2 0.7687793 0.7664319  
## 4 0.5 0.4553991 0.4448357  
## Specificity\_lasso$LASSO\_Specificity  
## 1 0.8300721  
## 2 0.8674415  
## 3 0.7910798  
## 4 0.4553991

Accuracy

## CutOff Simple\_Accuracy Accuracy\_step$Step\_Accuracy  
## 1 0.1 0.8359263 0.8334423  
## 2 0.15 0.8667800 0.8666492  
## 3 0.2 0.8907047 0.8917506  
## 4 0.5 0.9141064 0.9135835  
## Accuracy\_lasso$LASSO\_Accuracy  
## 1 0.8381488  
## 2 0.8656033  
## 3 0.8907047  
## 4 0.9155445

#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

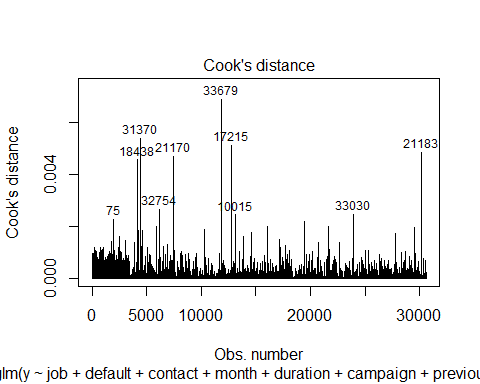
## Model Sensitivity Specificity Accuracy  
## 1 Simple 0.8286385 0.8715610 0.8667800  
## 2 Step 0.8274648 0.8715610 0.8666492  
## 3 LASSO 0.8509390 0.8674415 0.8656033

Based on Confusion matrix metrices, we chose the following model to be our final simple and interpretable mode: 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)

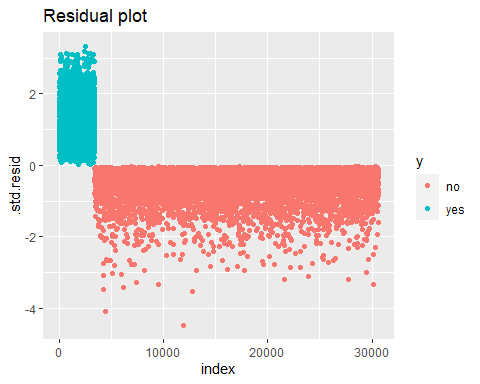
Our final model is stored as step.log. We will run residual diagnostics and outliers analysis on this model.

### Check residuals and outliers

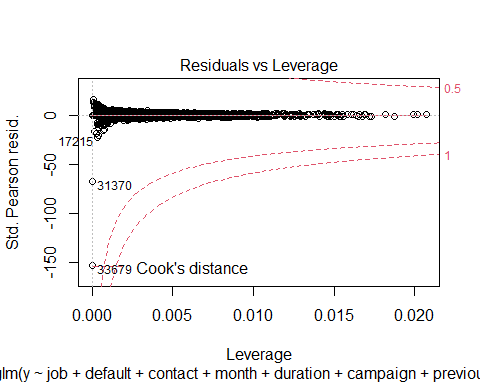
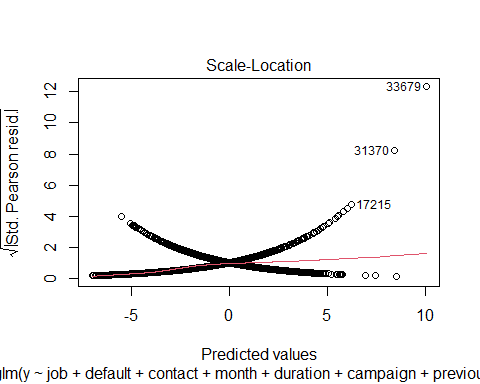
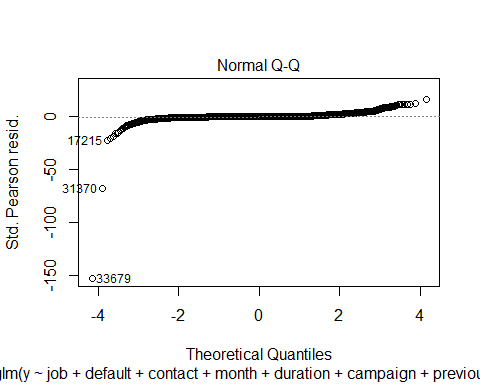
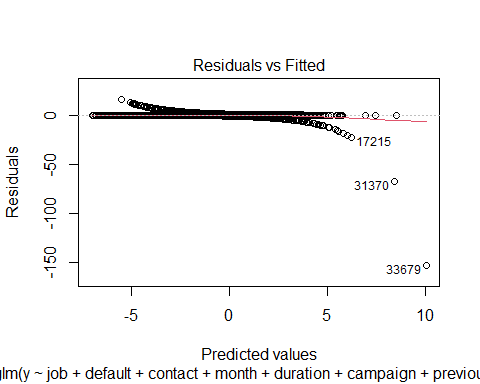
plot(step.log, which = 4, id.n = 10) #Cooks D plot



#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

## [1] 30593

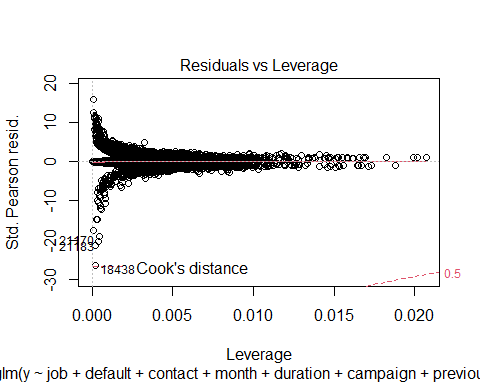
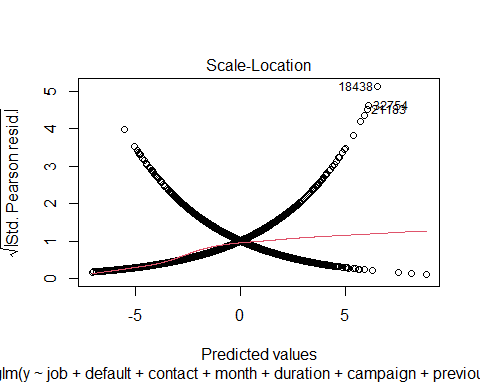
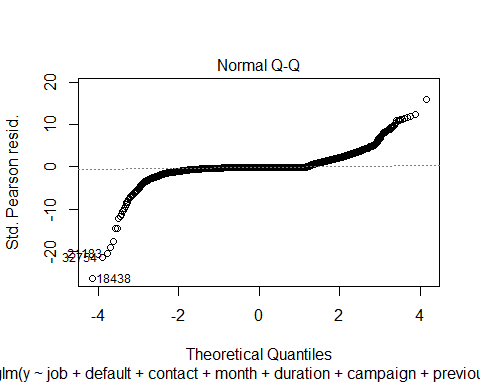
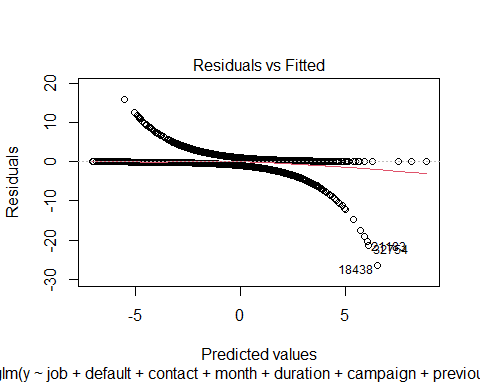
train2 <- train %>% dplyr::filter(!rownames(train) %in% c("17215","31370","33679"))  
nrow(train2)

## [1] 30590

#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)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.6185 -0.3015 -0.1734 -0.1207 3.3251   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.793e+01 5.197e+00 -9.222 < 2e-16 \*\*\*  
## jobblue-collar -3.461e-01 7.589e-02 -4.560 5.11e-06 \*\*\*  
## jobentrepreneur -1.405e-01 1.423e-01 -0.987 0.323585   
## jobhousemaid -2.036e-01 1.683e-01 -1.210 0.226442   
## jobmanagement -4.160e-02 9.775e-02 -0.426 0.670403   
## jobretired 8.181e-02 1.292e-01 0.633 0.526687   
## jobself-employed -1.145e-01 1.318e-01 -0.868 0.385200   
## jobservices -3.017e-01 9.532e-02 -3.165 0.001551 \*\*   
## jobstudent 1.689e-01 1.341e-01 1.259 0.207890   
## jobtechnician -7.230e-03 7.250e-02 -0.100 0.920555   
## jobunemployed -1.240e-01 1.537e-01 -0.806 0.419979   
## defaultunknown -3.301e-01 7.825e-02 -4.218 2.46e-05 \*\*\*  
## contacttelephone -3.629e-01 8.217e-02 -4.417 1.00e-05 \*\*\*  
## monthaug 2.527e-01 1.200e-01 2.106 0.035190 \*   
## monthdec 5.577e-02 2.511e-01 0.222 0.824221   
## monthjul 3.385e-01 1.114e-01 3.040 0.002365 \*\*   
## monthjun 3.970e-01 1.080e-01 3.676 0.000237 \*\*\*  
## monthmar 1.747e+00 1.438e-01 12.154 < 2e-16 \*\*\*  
## monthmay -6.332e-01 8.882e-02 -7.129 1.01e-12 \*\*\*  
## monthnov 7.351e-02 1.162e-01 0.633 0.526914   
## monthoct 4.219e-01 1.473e-01 2.864 0.004188 \*\*   
## monthsep -2.693e-03 1.605e-01 -0.017 0.986614   
## duration 2.743e-03 1.621e-04 16.919 < 2e-16 \*\*\*  
## campaign -4.637e-02 1.329e-02 -3.488 0.000486 \*\*\*  
## previous -2.731e-01 4.754e-02 -5.744 9.25e-09 \*\*\*  
## cons\_price\_idx 5.252e-01 5.708e-02 9.202 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.711e-02 6.137e-03 9.305 < 2e-16 \*\*\*  
## euribor3m -7.139e-01 2.232e-02 -31.982 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.931e-01 6.742e-02 -2.864 0.004186 \*\*   
## Age\_Grp38-47 -3.038e-01 7.159e-02 -4.243 2.20e-05 \*\*\*  
## Age\_Grp47-55 -1.537e-01 8.000e-02 -1.921 0.054734 .   
## Age\_Grp>55 -7.966e-02 1.082e-01 -0.736 0.461683   
## prevly\_CntctdYes 1.826e+00 1.041e-01 17.542 < 2e-16 \*\*\*  
## duration\_group5-10min 1.041e+00 7.093e-02 14.679 < 2e-16 \*\*\*  
## duration\_group10+ min 1.836e+00 1.334e-01 13.760 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21371 on 30589 degrees of freedom  
## Residual deviance: 12486 on 30555 degrees of freedom  
## AIC: 12556  
##   
## Number of Fisher Scoring iterations: 6

plot(step.log2)



#examine outliers 2   
nrow(train2) #30590

## [1] 30590

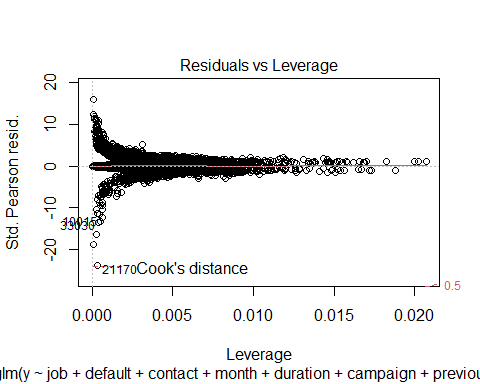
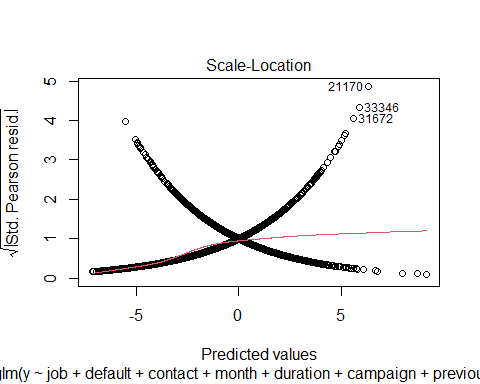
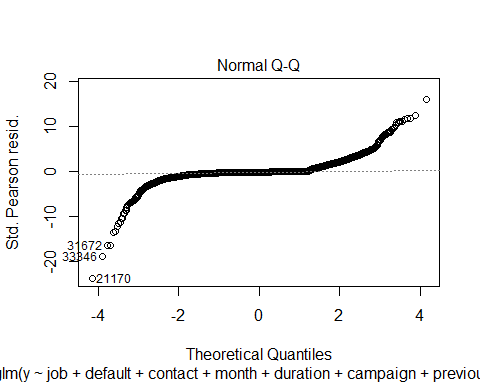
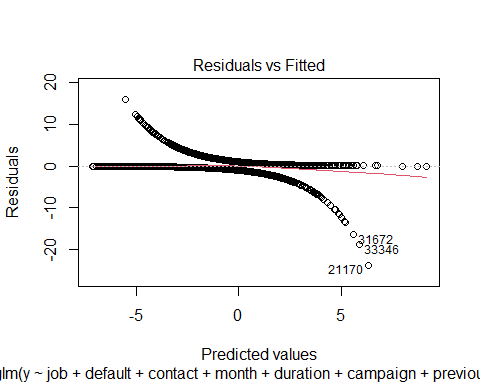
train3 <- train2 %>% dplyr::filter(!rownames(train2) %in% c("32754","18438","21183"))  
nrow(train3)

## [1] 30587

#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)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.5578 -0.3006 -0.1726 -0.1200 3.3267   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.746e+01 5.197e+00 -9.131 < 2e-16 \*\*\*  
## jobblue-collar -3.459e-01 7.604e-02 -4.548 5.41e-06 \*\*\*  
## jobentrepreneur -1.445e-01 1.427e-01 -1.012 0.311356   
## jobhousemaid -1.831e-01 1.679e-01 -1.090 0.275583   
## jobmanagement -4.197e-02 9.791e-02 -0.429 0.668166   
## jobretired 7.438e-02 1.294e-01 0.575 0.565434   
## jobself-employed -1.166e-01 1.321e-01 -0.883 0.377490   
## jobservices -3.037e-01 9.552e-02 -3.179 0.001477 \*\*   
## jobstudent 1.652e-01 1.342e-01 1.231 0.218408   
## jobtechnician -9.213e-03 7.264e-02 -0.127 0.899072   
## jobunemployed -1.257e-01 1.539e-01 -0.816 0.414230   
## defaultunknown -3.355e-01 7.851e-02 -4.273 1.93e-05 \*\*\*  
## contacttelephone -3.523e-01 8.221e-02 -4.285 1.82e-05 \*\*\*  
## monthaug 2.628e-01 1.200e-01 2.189 0.028598 \*   
## monthdec 5.986e-02 2.510e-01 0.238 0.811513   
## monthjul 3.409e-01 1.115e-01 3.057 0.002235 \*\*   
## monthjun 3.968e-01 1.081e-01 3.671 0.000242 \*\*\*  
## monthmar 1.748e+00 1.438e-01 12.157 < 2e-16 \*\*\*  
## monthmay -6.363e-01 8.889e-02 -7.158 8.20e-13 \*\*\*  
## monthnov 7.519e-02 1.163e-01 0.646 0.518044   
## monthoct 4.526e-01 1.473e-01 3.073 0.002116 \*\*   
## monthsep 5.133e-03 1.606e-01 0.032 0.974501   
## duration 2.907e-03 1.652e-04 17.599 < 2e-16 \*\*\*  
## campaign -4.734e-02 1.336e-02 -3.544 0.000394 \*\*\*  
## previous -2.737e-01 4.756e-02 -5.755 8.64e-09 \*\*\*  
## cons\_price\_idx 5.196e-01 5.708e-02 9.102 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.643e-02 6.140e-03 9.191 < 2e-16 \*\*\*  
## euribor3m -7.147e-01 2.235e-02 -31.980 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.968e-01 6.753e-02 -2.914 0.003569 \*\*   
## Age\_Grp38-47 -3.082e-01 7.173e-02 -4.297 1.73e-05 \*\*\*  
## Age\_Grp47-55 -1.527e-01 8.011e-02 -1.906 0.056592 .   
## Age\_Grp>55 -7.723e-02 1.083e-01 -0.713 0.475877   
## prevly\_CntctdYes 1.825e+00 1.041e-01 17.520 < 2e-16 \*\*\*  
## duration\_group5-10min 1.000e+00 7.140e-02 14.012 < 2e-16 \*\*\*  
## duration\_group10+ min 1.728e+00 1.348e-01 12.817 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21370 on 30586 degrees of freedom  
## Residual deviance: 12447 on 30552 degrees of freedom  
## AIC: 12517  
##   
## Number of Fisher Scoring iterations: 6

plot(step.log3)



Check what the outliers are

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  
)

## y job default contact month duration campaign previous  
## 18438 no admin. no cellular aug 3322 1 0  
## 31370 no blue-collar no cellular may 3509 2 2  
## 32754 no housemaid no telephone oct 2187 1 0  
## 33679 no admin. no telephone aug 3785 1 0  
## 17215 no management no cellular aug 3422 1 0  
## 21183 no blue-collar no telephone oct 3284 1 0  
## cons\_price\_idx cons\_conf\_idx euribor3m Age\_Grp prevly\_Cntctd  
## 18438 93.444 -36.1 4.964 47-55 No  
## 31370 92.893 -46.2 1.266 17-31 Yes  
## 32754 92.431 -26.9 0.737 >55 No  
## 33679 94.027 -38.3 0.888 17-31 No  
## 17215 93.444 -36.1 4.968 47-55 No  
## 21183 93.798 -40.4 4.912 17-31 No  
## duration\_group  
## 18438 10+ min  
## 31370 10+ min  
## 32754 10+ min  
## 33679 10+ min  
## 17215 10+ min  
## 21183 10+ min

Check if metrics change at all with the reduced model to determine if it’s worth removing outliers. No significant changes in coefficients and AIC only reduces slightly.

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")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 6608 471  
## yes 189 381  
##   
## Accuracy : 0.9137   
## 95% CI : (0.9072, 0.9199)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 2.83e-13   
##   
## Kappa : 0.4904   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.44718   
## Specificity : 0.97219   
## Pos Pred Value : 0.66842   
## Neg Pred Value : 0.93347   
## Prevalence : 0.11139   
## Detection Rate : 0.04981   
## Detection Prevalence : 0.07452   
## Balanced Accuracy : 0.70969   
##   
## 'Positive' Class : yes   
##

No real change. OK to keep outliers in the model.

# Complex Logistic Model

Run Initial Logistic Regression allowing for interaction

#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)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.2540 -0.3004 -0.1762 -0.1079 3.5076   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -4.703e+01 5.183e+00 -9.074 < 2e-16 \*\*\*  
## jobblue-collar -3.525e-01 7.601e-02 -4.637 3.53e-06 \*\*\*  
## jobentrepreneur -1.456e-01 1.417e-01 -1.027 0.304298   
## jobhousemaid -1.867e-01 1.686e-01 -1.108 0.268046   
## jobmanagement -4.556e-02 9.724e-02 -0.469 0.639400   
## jobretired 7.521e-02 1.297e-01 0.580 0.562028   
## jobself-employed -1.174e-01 1.316e-01 -0.892 0.372539   
## jobservices -2.995e-01 9.505e-02 -3.151 0.001628 \*\*   
## jobstudent 1.687e-01 1.337e-01 1.262 0.207063   
## jobtechnician -9.495e-03 7.204e-02 -0.132 0.895137   
## jobunemployed -1.301e-01 1.534e-01 -0.848 0.396525   
## defaultunknown -1.319e+00 1.669e-01 -7.900 2.78e-15 \*\*\*  
## contacttelephone -3.872e-01 8.203e-02 -4.721 2.35e-06 \*\*\*  
## monthaug 2.212e-01 1.197e-01 1.848 0.064627 .   
## monthdec 5.879e-02 2.500e-01 0.235 0.814102   
## monthjul 3.216e-01 1.110e-01 2.897 0.003764 \*\*   
## monthjun 3.820e-01 1.077e-01 3.547 0.000389 \*\*\*  
## monthmar 1.726e+00 1.428e-01 12.086 < 2e-16 \*\*\*  
## monthmay -6.342e-01 8.851e-02 -7.165 7.79e-13 \*\*\*  
## monthnov 5.860e-02 1.155e-01 0.507 0.611832   
## monthoct 4.087e-01 1.469e-01 2.783 0.005387 \*\*   
## monthsep -1.531e-02 1.599e-01 -0.096 0.923708   
## duration 2.246e-03 1.598e-04 14.050 < 2e-16 \*\*\*  
## campaign -4.499e-02 1.322e-02 -3.403 0.000665 \*\*\*  
## previous -2.665e-01 4.736e-02 -5.628 1.82e-08 \*\*\*  
## cons\_price\_idx 5.163e-01 5.692e-02 9.069 < 2e-16 \*\*\*  
## cons\_conf\_idx 5.606e-02 6.122e-03 9.157 < 2e-16 \*\*\*  
## euribor3m -6.988e-01 2.223e-02 -31.433 < 2e-16 \*\*\*  
## Age\_Grp32-37 -1.933e-01 6.703e-02 -2.883 0.003934 \*\*   
## Age\_Grp38-47 -2.979e-01 7.134e-02 -4.176 2.96e-05 \*\*\*  
## Age\_Grp47-55 -1.518e-01 7.986e-02 -1.902 0.057235 .   
## Age\_Grp>55 -5.958e-02 1.084e-01 -0.550 0.582506   
## prevly\_CntctdYes 1.815e+00 1.038e-01 17.493 < 2e-16 \*\*\*  
## duration\_group5-10min 1.144e+00 7.069e-02 16.182 < 2e-16 \*\*\*  
## duration\_group10+ min 2.014e+00 1.317e-01 15.295 < 2e-16 \*\*\*  
## defaultunknown:duration 1.729e-03 2.419e-04 7.148 8.80e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12478 on 30557 degrees of freedom  
## AIC: 12550  
##   
## Number of Fisher Scoring iterations: 7

#exp(cbind("Odds ratio" = coef(complex.log), confint.default(complex.log, level = 0.95)))

Testing for some interactions based on EDA

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  
 )

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(complex.log)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.3666 -0.2915 -0.1614 -0.1138 3.2713   
##   
## Coefficients: (2 not defined because of singularities)  
## Estimate Std. Error z value  
## (Intercept) -1.158e+02 4.544e+01 -2.548  
## jobblue-collar -2.074e-01 9.648e-02 -2.150  
## jobentrepreneur -1.081e-01 1.451e-01 -0.745  
## jobhousemaid -1.572e-01 1.791e-01 -0.878  
## jobmanagement -8.738e-02 1.005e-01 -0.869  
## jobretired 8.941e-02 1.357e-01 0.659  
## jobself-employed -1.469e-01 1.347e-01 -1.090  
## jobservices -1.717e-01 1.014e-01 -1.693  
## jobstudent 1.712e-01 1.407e-01 1.217  
## jobtechnician 1.850e-02 8.278e-02 0.223  
## jobunemployed -1.836e-02 1.571e-01 -0.117  
## defaultunknown -2.734e-01 7.953e-02 -3.438  
## contacttelephone -5.725e-01 9.870e-02 -5.801  
## monthaug 3.082e-01 5.606e-01 0.550  
## monthdec 3.861e+01 3.332e+01 1.159  
## monthjul 1.787e-01 4.752e-01 0.376  
## monthjun 3.173e-01 5.571e-01 0.570  
## monthmar -7.670e-01 5.416e-01 -1.416  
## monthmay -7.036e-01 5.542e-01 -1.270  
## monthnov 3.596e-01 6.079e-01 0.592  
## monthoct -1.103e+00 5.051e-01 -2.184  
## monthsep 1.363e+00 2.665e+00 0.512  
## duration 2.044e-03 1.838e-04 11.121  
## campaign -1.464e-01 2.568e-02 -5.701  
## previous -2.719e-01 4.805e-02 -5.659  
## cons\_price\_idx 1.268e+00 4.889e-01 2.593  
## cons\_conf\_idx 1.040e-01 1.676e-02 6.206  
## euribor3m 4.579e+01 3.953e+01 1.158  
## Age\_Grp32-37 -1.034e-01 3.894e-01 -0.265  
## Age\_Grp38-47 -1.406e-01 3.578e-01 -0.393  
## Age\_Grp47-55 2.801e-01 3.433e-01 0.816  
## Age\_Grp>55 5.294e-01 3.313e-01 1.598  
## prevly\_CntctdYes 1.787e+00 1.048e-01 17.046  
## duration\_group5-10min 1.178e+00 7.186e-02 16.394  
## duration\_group10+ min 2.098e+00 1.331e-01 15.769  
## educationbasic.6y 6.288e-01 3.979e-01 1.580  
## educationbasic.9y 4.231e-01 3.197e-01 1.323  
## educationhigh.school 4.739e-01 3.089e-01 1.534  
## educationilliterate 6.616e-01 1.845e+00 0.359  
## educationprofessional.course 5.351e-01 3.284e-01 1.630  
## educationuniversity.degree 5.429e-01 3.097e-01 1.753  
## Age\_Grp32-37:educationbasic.6y -2.819e-02 5.173e-01 -0.054  
## Age\_Grp38-47:educationbasic.6y -1.123e-01 4.814e-01 -0.233  
## Age\_Grp47-55:educationbasic.6y -6.306e-01 5.096e-01 -1.237  
## Age\_Grp>55:educationbasic.6y -9.872e-01 5.704e-01 -1.731  
## Age\_Grp32-37:educationbasic.9y -2.718e-01 4.345e-01 -0.625  
## Age\_Grp38-47:educationbasic.9y -1.880e-01 4.013e-01 -0.468  
## Age\_Grp47-55:educationbasic.9y -3.200e-01 3.999e-01 -0.800  
## Age\_Grp>55:educationbasic.9y -6.031e-01 4.157e-01 -1.451  
## Age\_Grp32-37:educationhigh.school -1.781e-01 4.096e-01 -0.435  
## Age\_Grp38-47:educationhigh.school -2.637e-01 3.813e-01 -0.692  
## Age\_Grp47-55:educationhigh.school -4.348e-01 3.769e-01 -1.154  
## Age\_Grp>55:educationhigh.school -5.231e-01 3.726e-01 -1.404  
## Age\_Grp32-37:educationilliterate 2.759e+00 2.337e+00 1.180  
## Age\_Grp38-47:educationilliterate 1.436e+00 2.245e+00 0.640  
## Age\_Grp47-55:educationilliterate NA NA NA  
## Age\_Grp>55:educationilliterate NA NA NA  
## Age\_Grp32-37:educationprofessional.course -2.605e-01 4.294e-01 -0.607  
## Age\_Grp38-47:educationprofessional.course -1.178e-01 4.041e-01 -0.292  
## Age\_Grp47-55:educationprofessional.course -5.221e-01 4.046e-01 -1.290  
## Age\_Grp>55:educationprofessional.course -5.532e-01 3.918e-01 -1.412  
## Age\_Grp32-37:educationuniversity.degree 4.261e-02 4.025e-01 0.106  
## Age\_Grp38-47:educationuniversity.degree -1.108e-01 3.759e-01 -0.295  
## Age\_Grp47-55:educationuniversity.degree -3.819e-01 3.661e-01 -1.043  
## Age\_Grp>55:educationuniversity.degree -7.745e-01 3.575e-01 -2.167  
## duration:campaign 2.009e-04 3.860e-05 5.204  
## cons\_price\_idx:euribor3m -4.952e-01 4.205e-01 -1.178  
## monthaug:euribor3m -2.863e-01 4.545e-01 -0.630  
## monthdec:euribor3m -5.432e+01 4.678e+01 -1.161  
## monthjul:euribor3m -1.359e-02 3.756e-01 -0.036  
## monthjun:euribor3m 1.390e-01 4.403e-01 0.316  
## monthmar:euribor3m 2.076e+00 4.231e-01 4.905  
## monthmay:euribor3m 2.314e-02 4.055e-01 0.057  
## monthnov:euribor3m -3.732e-01 4.912e-01 -0.760  
## monthoct:euribor3m 1.197e+00 3.944e-01 3.035  
## monthsep:euribor3m -2.077e+00 3.139e+00 -0.662  
## Pr(>|z|)   
## (Intercept) 0.010820 \*   
## jobblue-collar 0.031582 \*   
## jobentrepreneur 0.456251   
## jobhousemaid 0.380182   
## jobmanagement 0.384687   
## jobretired 0.509872   
## jobself-employed 0.275594   
## jobservices 0.090453 .   
## jobstudent 0.223758   
## jobtechnician 0.823158   
## jobunemployed 0.906943   
## defaultunknown 0.000585 \*\*\*  
## contacttelephone 6.60e-09 \*\*\*  
## monthaug 0.582408   
## monthdec 0.246549   
## monthjul 0.706901   
## monthjun 0.568986   
## monthmar 0.156711   
## monthmay 0.204226   
## monthnov 0.554100   
## monthoct 0.028989 \*   
## monthsep 0.608926   
## duration < 2e-16 \*\*\*  
## campaign 1.19e-08 \*\*\*  
## previous 1.52e-08 \*\*\*  
## cons\_price\_idx 0.009504 \*\*   
## cons\_conf\_idx 5.44e-10 \*\*\*  
## euribor3m 0.246739   
## Age\_Grp32-37 0.790669   
## Age\_Grp38-47 0.694277   
## Age\_Grp47-55 0.414541   
## Age\_Grp>55 0.110018   
## prevly\_CntctdYes < 2e-16 \*\*\*  
## duration\_group5-10min < 2e-16 \*\*\*  
## duration\_group10+ min < 2e-16 \*\*\*  
## educationbasic.6y 0.114104   
## educationbasic.9y 0.185731   
## educationhigh.school 0.125013   
## educationilliterate 0.719855   
## educationprofessional.course 0.103187   
## educationuniversity.degree 0.079660 .   
## Age\_Grp32-37:educationbasic.6y 0.956540   
## Age\_Grp38-47:educationbasic.6y 0.815514   
## Age\_Grp47-55:educationbasic.6y 0.215968   
## Age\_Grp>55:educationbasic.6y 0.083504 .   
## Age\_Grp32-37:educationbasic.9y 0.531665   
## Age\_Grp38-47:educationbasic.9y 0.639512   
## Age\_Grp47-55:educationbasic.9y 0.423509   
## Age\_Grp>55:educationbasic.9y 0.146851   
## Age\_Grp32-37:educationhigh.school 0.663689   
## Age\_Grp38-47:educationhigh.school 0.489132   
## Age\_Grp47-55:educationhigh.school 0.248630   
## Age\_Grp>55:educationhigh.school 0.160299   
## Age\_Grp32-37:educationilliterate 0.237803   
## Age\_Grp38-47:educationilliterate 0.522307   
## Age\_Grp47-55:educationilliterate NA   
## Age\_Grp>55:educationilliterate NA   
## Age\_Grp32-37:educationprofessional.course 0.544059   
## Age\_Grp38-47:educationprofessional.course 0.770646   
## Age\_Grp47-55:educationprofessional.course 0.196934   
## Age\_Grp>55:educationprofessional.course 0.157950   
## Age\_Grp32-37:educationuniversity.degree 0.915688   
## Age\_Grp38-47:educationuniversity.degree 0.768119   
## Age\_Grp47-55:educationuniversity.degree 0.296940   
## Age\_Grp>55:educationuniversity.degree 0.030262 \*   
## duration:campaign 1.95e-07 \*\*\*  
## cons\_price\_idx:euribor3m 0.238959   
## monthaug:euribor3m 0.528713   
## monthdec:euribor3m 0.245523   
## monthjul:euribor3m 0.971134   
## monthjun:euribor3m 0.752260   
## monthmar:euribor3m 9.33e-07 \*\*\*  
## monthmay:euribor3m 0.954503   
## monthnov:euribor3m 0.447411   
## monthoct:euribor3m 0.002408 \*\*   
## monthsep:euribor3m 0.508166   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12220 on 30519 degrees of freedom  
## AIC: 12368  
##   
## Number of Fisher Scoring iterations: 10

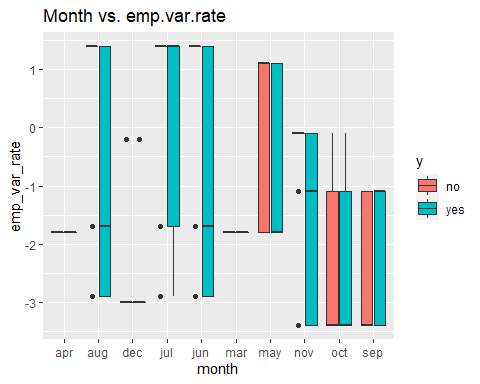
#exp(cbind("Odds ratio" = coef(complex.log), confint.default(complex.log, level = 0.95)))  
#complex.pred <- predict(complex.log, newdata = test, type="response")

Duration and campaign is significant but has a small coefficient/odds ratio. Month and euribor3m are significant as an interaction term with some odds ratios being very large. Education and age group is not significant for interaction. We also likely don’t want to use education for any interaction due to no observations falling into that group.

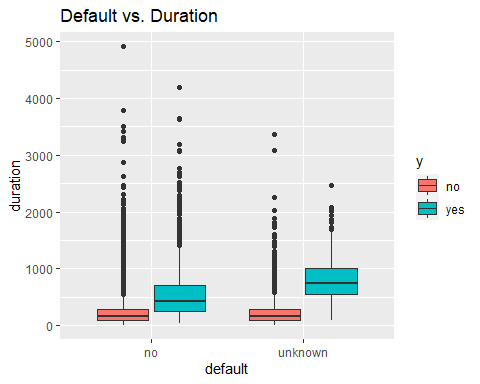
# Additional EDA for interaction

Let’s run some more EDA with specific focus on our variables from the stepwise model.

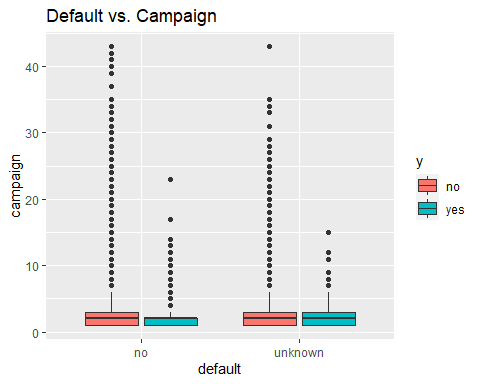
#numerical y vars  
ggplot(df, aes(x=month , y=emp\_var\_rate, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Month vs. emp.var.rate")



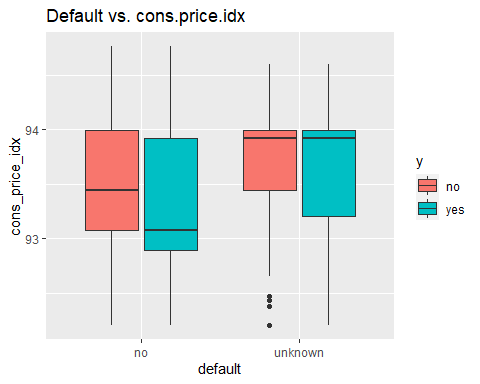
ggplot(df, aes(x=default , y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. Duration")



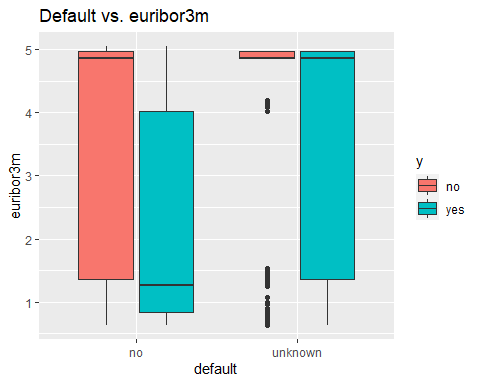
ggplot(df, aes(x=default , y=campaign, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. Campaign")



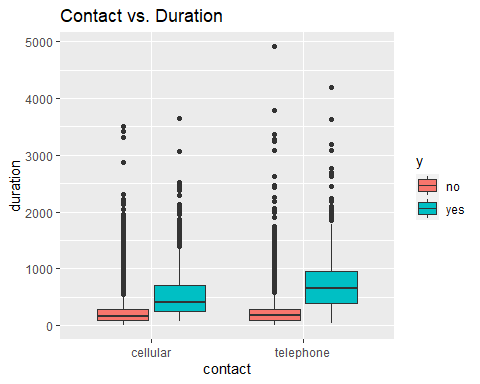
ggplot(df, aes(x=default , y=cons\_price\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. cons.price.idx")



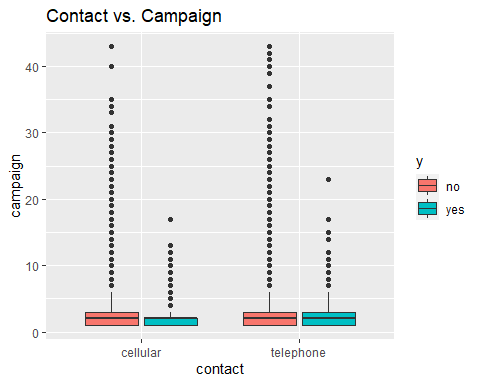
ggplot(df, aes(x=default , y=euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Default vs. euribor3m")



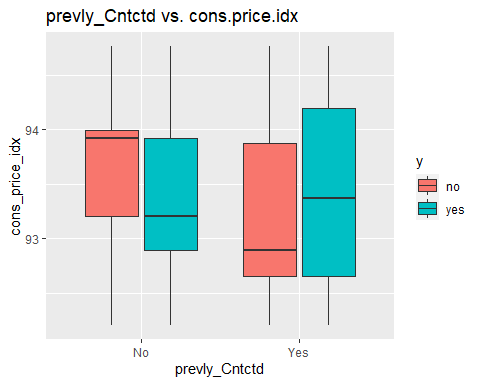
ggplot(df, aes(x=contact , y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Contact vs. Duration")



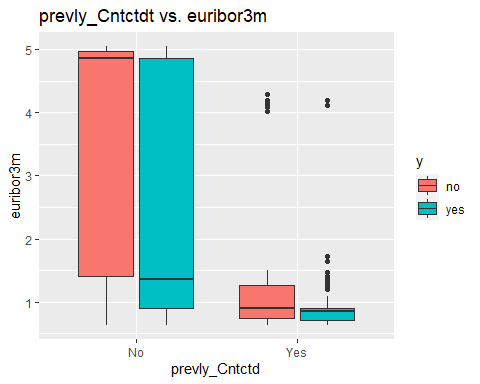
ggplot(df, aes(x=contact , y=campaign, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("Contact vs. Campaign")



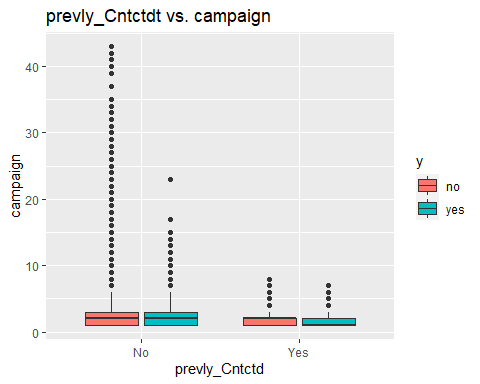
ggplot(df, aes(x=prevly\_Cntctd , y=cons\_price\_idx, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("prevly\_Cntctd vs. cons.price.idx")



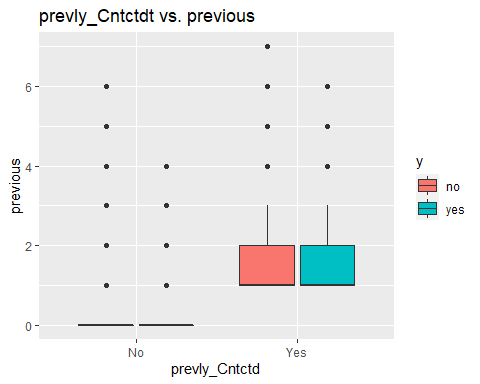
ggplot(df, aes(x=prevly\_Cntctd , y=euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("prevly\_Cntctdt vs. euribor3m")



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)

##   
## apr aug dec jul jun mar  
## no 0.96296296 0.89429530 0.96103896 0.85475793 0.88454012 0.97619048  
## unknown 0.03703704 0.10570470 0.03896104 0.14524207 0.11545988 0.02380952  
##   
## may nov oct sep  
## no 0.84337349 0.94805195 0.96632997 0.98666667  
## unknown 0.15662651 0.05194805 0.03367003 0.01333333

prop.table(table(df\_No$y,df\_No$month),2)

##   
## apr aug dec jul jun mar may nov oct sep  
## no 1 1 1 1 1 1 1 1 1 1  
## yes 0 0 0 0 0 0 0 0 0 0

Add in some interactions based on what we see from these charts: default/duration, contact/duration, default/month and keeping in month/euribor3m

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  
 )

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(complex.log)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.7860 -0.2907 -0.1637 -0.0999 3.5331   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -6.313e+01 7.540e+00 -8.372 < 2e-16 \*\*\*  
## jobblue-collar -3.049e-01 7.692e-02 -3.964 7.37e-05 \*\*\*  
## jobentrepreneur -1.487e-01 1.442e-01 -1.031 0.302526   
## jobhousemaid -1.942e-01 1.711e-01 -1.135 0.256263   
## jobmanagement -4.135e-02 9.846e-02 -0.420 0.674512   
## jobretired 5.670e-02 1.310e-01 0.433 0.665166   
## jobself-employed -9.699e-02 1.320e-01 -0.735 0.462549   
## jobservices -2.508e-01 9.597e-02 -2.613 0.008963 \*\*   
## jobstudent 1.053e-01 1.354e-01 0.778 0.436833   
## jobtechnician -7.349e-03 7.280e-02 -0.101 0.919593   
## jobunemployed -9.814e-02 1.547e-01 -0.634 0.525803   
## defaultunknown -2.498e+00 3.965e-01 -6.301 2.97e-10 \*\*\*  
## contacttelephone -8.919e-01 1.194e-01 -7.469 8.11e-14 \*\*\*  
## monthaug 2.267e-02 4.707e-01 0.048 0.961585   
## monthdec 4.625e+01 3.348e+01 1.382 0.167108   
## monthjul 2.144e-01 4.713e-01 0.455 0.649218   
## monthjun 7.549e-01 4.944e-01 1.527 0.126749   
## monthmar -8.752e-01 4.989e-01 -1.754 0.079432 .   
## monthmay -5.930e-01 5.487e-01 -1.081 0.279827   
## monthnov -6.922e-02 4.668e-01 -0.148 0.882120   
## monthoct -1.306e+00 4.646e-01 -2.811 0.004939 \*\*   
## monthsep -4.590e-01 2.249e+00 -0.204 0.838292   
## duration 1.984e-03 1.709e-04 11.613 < 2e-16 \*\*\*  
## campaign -4.119e-02 1.337e-02 -3.081 0.002066 \*\*   
## previous -2.695e-01 4.770e-02 -5.650 1.61e-08 \*\*\*  
## cons\_price\_idx 7.057e-01 8.200e-02 8.607 < 2e-16 \*\*\*  
## cons\_conf\_idx 8.912e-02 1.096e-02 8.131 4.27e-16 \*\*\*  
## euribor3m -6.521e-01 3.748e-01 -1.740 0.081892 .   
## Age\_Grp32-37 -1.994e-01 6.771e-02 -2.944 0.003239 \*\*   
## Age\_Grp38-47 -2.988e-01 7.205e-02 -4.147 3.37e-05 \*\*\*  
## Age\_Grp47-55 -1.325e-01 8.063e-02 -1.643 0.100418   
## Age\_Grp>55 -7.743e-02 1.094e-01 -0.708 0.479093   
## prevly\_CntctdYes 1.766e+00 1.042e-01 16.945 < 2e-16 \*\*\*  
## duration\_group5-10min 1.250e+00 7.285e-02 17.154 < 2e-16 \*\*\*  
## duration\_group10+ min 2.175e+00 1.345e-01 16.175 < 2e-16 \*\*\*  
## defaultunknown:duration 1.447e-03 2.516e-04 5.752 8.82e-09 \*\*\*  
## contacttelephone:duration 8.966e-04 1.872e-04 4.789 1.67e-06 \*\*\*  
## defaultunknown:monthaug 1.452e+00 4.271e-01 3.400 0.000675 \*\*\*  
## defaultunknown:monthdec 3.870e+00 1.460e+00 2.651 0.008027 \*\*   
## defaultunknown:monthjul 1.640e+00 4.118e-01 3.982 6.83e-05 \*\*\*  
## defaultunknown:monthjun 1.324e+00 4.412e-01 3.001 0.002689 \*\*   
## defaultunknown:monthmar 1.380e+00 7.203e-01 1.915 0.055459 .   
## defaultunknown:monthmay 1.320e+00 3.982e-01 3.316 0.000912 \*\*\*  
## defaultunknown:monthnov 1.650e+00 4.989e-01 3.307 0.000943 \*\*\*  
## defaultunknown:monthoct 2.585e+00 7.703e-01 3.355 0.000793 \*\*\*  
## defaultunknown:monthsep 2.110e+00 1.212e+00 1.740 0.081856 .   
## monthaug:euribor3m -9.386e-02 3.798e-01 -0.247 0.804796   
## monthdec:euribor3m -6.547e+01 4.702e+01 -1.392 0.163808   
## monthjul:euribor3m -9.619e-02 3.727e-01 -0.258 0.796357   
## monthjun:euribor3m -2.696e-01 3.862e-01 -0.698 0.485144   
## monthmar:euribor3m 2.113e+00 3.759e-01 5.620 1.91e-08 \*\*\*  
## monthmay:euribor3m -1.312e-01 4.030e-01 -0.325 0.744805   
## monthnov:euribor3m -7.810e-02 3.657e-01 -0.214 0.830900   
## monthoct:euribor3m 1.242e+00 3.794e-01 3.273 0.001062 \*\*   
## monthsep:euribor3m 1.686e-02 2.662e+00 0.006 0.994947   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12181 on 30538 degrees of freedom  
## AIC: 12291  
##   
## Number of Fisher Scoring iterations: 10

Default being unknown seems to carry some sort of significance in our model. We are keeping it in as we are not counting it as ‘missing’ but moreso as an actual ‘unknown’ value.

### Run step-wise selection on this model

step.complex<-complex.log %>% stepAIC(trace=FALSE)

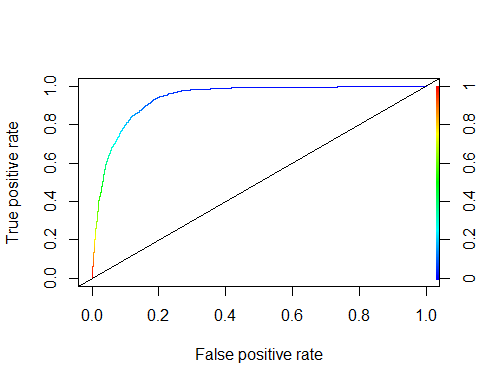
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

summary(step.complex)

##   
## Call:  
## glm(formula = 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)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.7860 -0.2907 -0.1637 -0.0999 3.5331   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -6.313e+01 7.540e+00 -8.372 < 2e-16 \*\*\*  
## jobblue-collar -3.049e-01 7.692e-02 -3.964 7.37e-05 \*\*\*  
## jobentrepreneur -1.487e-01 1.442e-01 -1.031 0.302526   
## jobhousemaid -1.942e-01 1.711e-01 -1.135 0.256263   
## jobmanagement -4.135e-02 9.846e-02 -0.420 0.674512   
## jobretired 5.670e-02 1.310e-01 0.433 0.665166   
## jobself-employed -9.699e-02 1.320e-01 -0.735 0.462549   
## jobservices -2.508e-01 9.597e-02 -2.613 0.008963 \*\*   
## jobstudent 1.053e-01 1.354e-01 0.778 0.436833   
## jobtechnician -7.349e-03 7.280e-02 -0.101 0.919593   
## jobunemployed -9.814e-02 1.547e-01 -0.634 0.525803   
## defaultunknown -2.498e+00 3.965e-01 -6.301 2.97e-10 \*\*\*  
## contacttelephone -8.919e-01 1.194e-01 -7.469 8.11e-14 \*\*\*  
## monthaug 2.267e-02 4.707e-01 0.048 0.961585   
## monthdec 4.625e+01 3.348e+01 1.382 0.167108   
## monthjul 2.144e-01 4.713e-01 0.455 0.649218   
## monthjun 7.549e-01 4.944e-01 1.527 0.126749   
## monthmar -8.752e-01 4.989e-01 -1.754 0.079432 .   
## monthmay -5.930e-01 5.487e-01 -1.081 0.279827   
## monthnov -6.922e-02 4.668e-01 -0.148 0.882120   
## monthoct -1.306e+00 4.646e-01 -2.811 0.004939 \*\*   
## monthsep -4.590e-01 2.249e+00 -0.204 0.838292   
## duration 1.984e-03 1.709e-04 11.613 < 2e-16 \*\*\*  
## campaign -4.119e-02 1.337e-02 -3.081 0.002066 \*\*   
## previous -2.695e-01 4.770e-02 -5.650 1.61e-08 \*\*\*  
## cons\_price\_idx 7.057e-01 8.200e-02 8.607 < 2e-16 \*\*\*  
## cons\_conf\_idx 8.912e-02 1.096e-02 8.131 4.27e-16 \*\*\*  
## euribor3m -6.521e-01 3.748e-01 -1.740 0.081892 .   
## Age\_Grp32-37 -1.994e-01 6.771e-02 -2.944 0.003239 \*\*   
## Age\_Grp38-47 -2.988e-01 7.205e-02 -4.147 3.37e-05 \*\*\*  
## Age\_Grp47-55 -1.325e-01 8.063e-02 -1.643 0.100418   
## Age\_Grp>55 -7.743e-02 1.094e-01 -0.708 0.479093   
## prevly\_CntctdYes 1.766e+00 1.042e-01 16.945 < 2e-16 \*\*\*  
## duration\_group5-10min 1.250e+00 7.285e-02 17.154 < 2e-16 \*\*\*  
## duration\_group10+ min 2.175e+00 1.345e-01 16.175 < 2e-16 \*\*\*  
## defaultunknown:duration 1.447e-03 2.516e-04 5.752 8.82e-09 \*\*\*  
## contacttelephone:duration 8.966e-04 1.872e-04 4.789 1.67e-06 \*\*\*  
## defaultunknown:monthaug 1.452e+00 4.271e-01 3.400 0.000675 \*\*\*  
## defaultunknown:monthdec 3.870e+00 1.460e+00 2.651 0.008027 \*\*   
## defaultunknown:monthjul 1.640e+00 4.118e-01 3.982 6.83e-05 \*\*\*  
## defaultunknown:monthjun 1.324e+00 4.412e-01 3.001 0.002689 \*\*   
## defaultunknown:monthmar 1.380e+00 7.203e-01 1.915 0.055459 .   
## defaultunknown:monthmay 1.320e+00 3.982e-01 3.316 0.000912 \*\*\*  
## defaultunknown:monthnov 1.650e+00 4.989e-01 3.307 0.000943 \*\*\*  
## defaultunknown:monthoct 2.585e+00 7.703e-01 3.355 0.000793 \*\*\*  
## defaultunknown:monthsep 2.110e+00 1.212e+00 1.740 0.081856 .   
## monthaug:euribor3m -9.386e-02 3.798e-01 -0.247 0.804796   
## monthdec:euribor3m -6.547e+01 4.702e+01 -1.392 0.163808   
## monthjul:euribor3m -9.619e-02 3.727e-01 -0.258 0.796357   
## monthjun:euribor3m -2.696e-01 3.862e-01 -0.698 0.485144   
## monthmar:euribor3m 2.113e+00 3.759e-01 5.620 1.91e-08 \*\*\*  
## monthmay:euribor3m -1.312e-01 4.030e-01 -0.325 0.744805   
## monthnov:euribor3m -7.810e-02 3.657e-01 -0.214 0.830900   
## monthoct:euribor3m 1.242e+00 3.794e-01 3.273 0.001062 \*\*   
## monthsep:euribor3m 1.686e-02 2.662e+00 0.006 0.994947   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 21372 on 30592 degrees of freedom  
## Residual deviance: 12181 on 30538 degrees of freedom  
## AIC: 12291  
##   
## Number of Fisher Scoring iterations: 10

#Check metrics with standard 0.5 accuracy and our 0.15 accuracy for complex log model

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

##   
## class.complex no yes  
## no 6608 463  
## yes 189 389

complex<-confusionMatrix(conf.complex, positive = "yes")  
complex

## Confusion Matrix and Statistics  
##   
##   
## class.complex no yes  
## no 6608 463  
## yes 189 389  
##   
## Accuracy : 0.9148   
## 95% CI : (0.9083, 0.9209)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 2.695e-14   
##   
## Kappa : 0.4989   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.45657   
## Specificity : 0.97219   
## Pos Pred Value : 0.67301   
## Neg Pred Value : 0.93452   
## Prevalence : 0.11139   
## Detection Rate : 0.05086   
## Detection Prevalence : 0.07557   
## Balanced Accuracy : 0.71438   
##   
## 'Positive' Class : yes   
##

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

##   
## class.complex no yes  
## no 5910 122  
## yes 887 730

complex<-confusionMatrix(conf.complex, positive = "yes")  
complex

## Confusion Matrix and Statistics  
##   
##   
## class.complex no yes  
## no 5910 122  
## yes 887 730  
##   
## Accuracy : 0.8681   
## 95% CI : (0.8603, 0.8756)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.5215   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.85681   
## Specificity : 0.86950   
## Pos Pred Value : 0.45145   
## Neg Pred Value : 0.97977   
## Prevalence : 0.11139   
## Detection Rate : 0.09544   
## Detection Prevalence : 0.21140   
## Balanced Accuracy : 0.86315   
##   
## 'Positive' Class : yes   
##

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

##   
## class.complex no yes  
## no 6388 273  
## yes 409 579

complex<-confusionMatrix(conf.complex, positive = "yes")  
complex

## Confusion Matrix and Statistics  
##   
##   
## class.complex no yes  
## no 6388 273  
## yes 409 579  
##   
## Accuracy : 0.9108   
## 95% CI : (0.9042, 0.9171)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1.067e-10   
##   
## Kappa : 0.579   
##   
## Mcnemar's Test P-Value : 2.348e-07   
##   
## Sensitivity : 0.6796   
## Specificity : 0.9398   
## Pos Pred Value : 0.5860   
## Neg Pred Value : 0.9590   
## Prevalence : 0.1114   
## Detection Rate : 0.0757   
## Detection Prevalence : 0.1292   
## Balanced Accuracy : 0.8097   
##   
## 'Positive' Class : yes   
##

cutoff<-0.108  
class.complex\_.108<-factor(ifelse(complex.pred>cutoff,"yes","no"),levels=c("no","yes"))  
  
#Confusion Matrix  
conf.complex\_.108<-table(class.complex\_.108,test$y)  
conf.complex\_.108

##   
## class.complex\_.108 no yes  
## no 5727 89  
## yes 1070 763

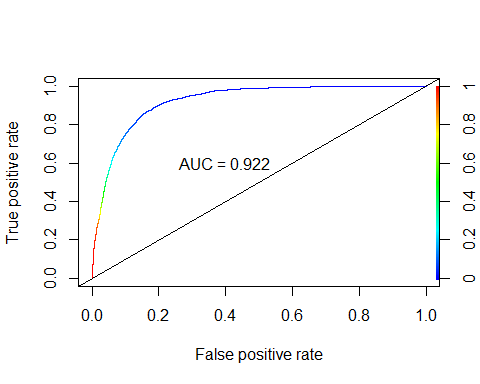
complex\_.108<-confusionMatrix(conf.complex\_.108, positive = "yes")  
complex\_.108

## Confusion Matrix and Statistics  
##   
##   
## class.complex\_.108 no yes  
## no 5727 89  
## yes 1070 763  
##   
## Accuracy : 0.8485   
## 95% CI : (0.8402, 0.8564)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4909   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.89554   
## Specificity : 0.84258   
## Pos Pred Value : 0.41626   
## Neg Pred Value : 0.98470   
## Prevalence : 0.11139   
## Detection Rate : 0.09975   
## Detection Prevalence : 0.23964   
## Balanced Accuracy : 0.86906   
##   
## 'Positive' Class : yes   
##

Our metrics look much better with the 0.15 cut-off, similar to our simpler model above. Not much performance enhancement by making the model more complex.

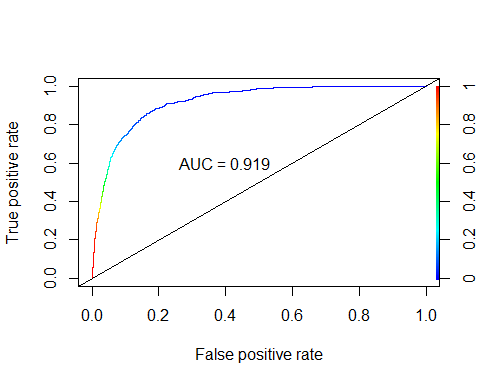
# LDA & QDA

#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

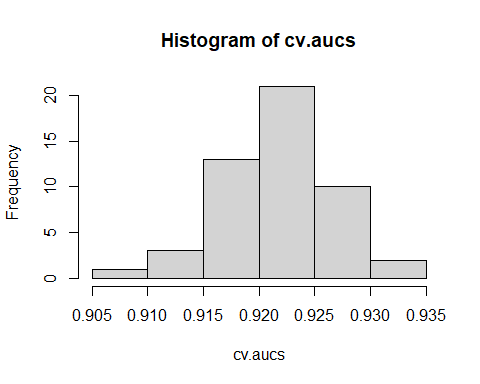
# Test Set  
test.lda.x <- test[ , sapply(test, is.numeric)]  
test.lda.y <- test$y  
pred.lda1 <- predict(fit.lda, newdata = test.lda.x)  
preds1 <- pred.lda1$posterior  
preds1 <- as.data.frame(preds1)  
pred1 <- prediction(preds1[,2],test.lda.y)  
roc.perf = performance(pred1, measure = "tpr", x.measure = "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

Run CV loop for LDA

#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)



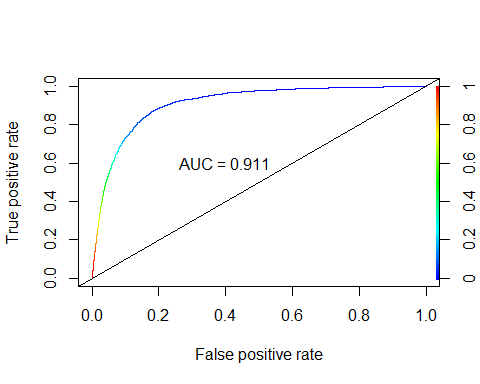
summary(cv.aucs)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.9100 0.9187 0.9219 0.9217 0.9248 0.9336

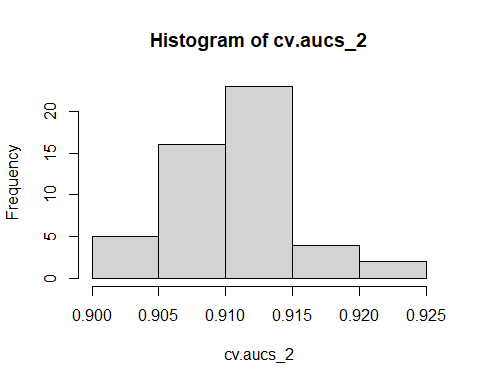
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.9100 0.9187 0.9219 0.9217 0.9248 0.9336

LDA had good results keeping all numeric variables in the data with 50-fold CV

#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)



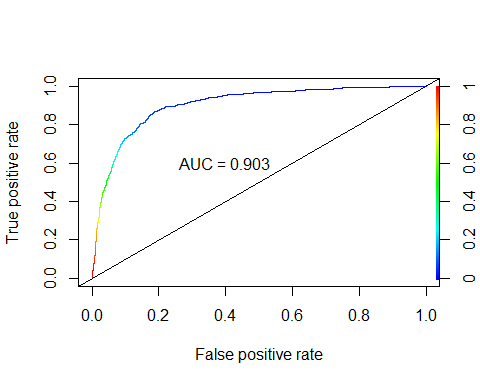
summary(cv.aucs\_2)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.9003 0.9074 0.9114 0.9107 0.9137 0.9233

# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.9003 0.9074 0.9114 0.9107 0.9137 0.9233

The subset using the numeric vars from our step logistic model has very similar performance

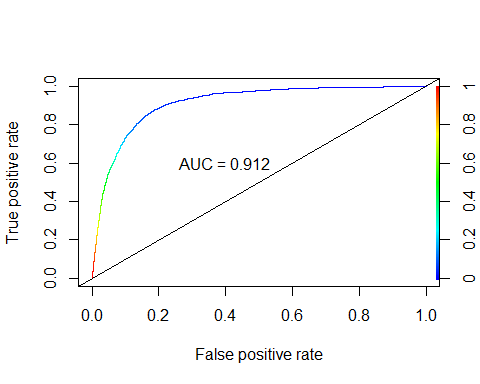
#run on test set  
# Test Set  
pred.lda1\_step <- predict(fit.lda\_step, newdata = test.lda.x)  
preds1\_step <- pred.lda1\_step$posterior  
preds1\_step <- as.data.frame(preds1\_step)  
pred1\_step <- prediction(preds1\_step[, 2], test.lda.y)  
roc.perf\_step2 = performance(pred1\_step, measure = "tpr", 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

Run once more, but just the numeric from above, removing those that act more like factors.

#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 = ""))

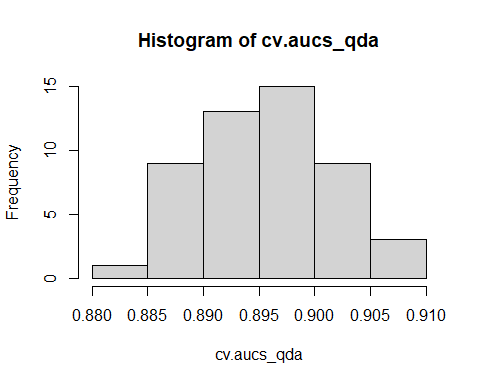


#AUC = 0.912

No significant change.

### Running QDA

#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)



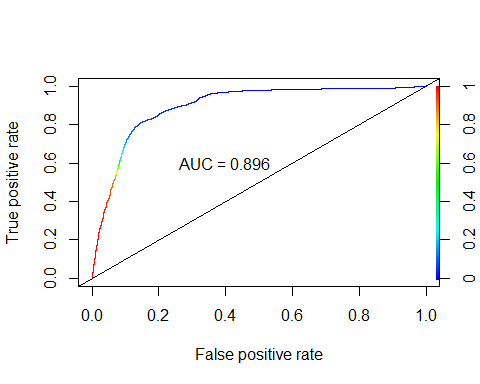
summary(cv.aucs\_qda)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.8845 0.8911 0.8958 0.8955 0.8993 0.9097

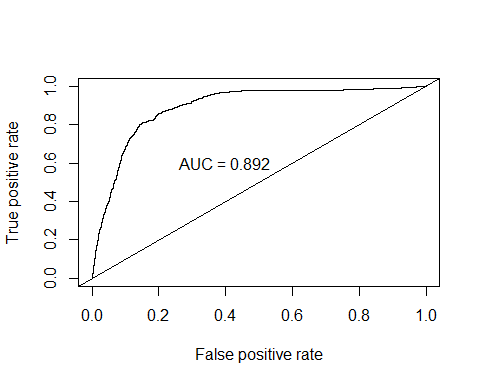
# Min. 1st Qu. Median Mean 3rd Qu. Max.   
# 0.8845 0.8911 0.8958 0.8955 0.8993 0.9097

### QDA on train set

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 = ""))

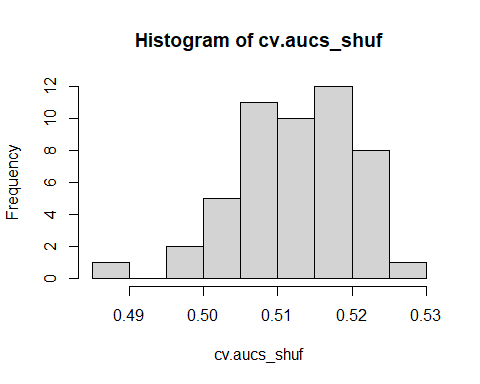


#AUC = 0.896  
  
# Test Set  
pred.qda1 <- predict(fit.qda, newdata = test.lda.x)  
preds1\_qda <- pred.qda1$posterior  
preds1\_qda <- as.data.frame(preds1\_qda)  
pred1\_qda <- prediction(preds1\_qda[, 2], test.lda.y)  
roc.perf\_qda1 = performance(pred1\_qda, measure = "tpr", x.measure = "fpr")  
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

# Min. 1st Qu. Median Mean 3rd Qu. Max.   
 #0.4871 0.5081 0.5127 0.5125 0.5186 0.5299

The AUC degrades to ~0.5 so we seem to be doing something right.

Generate confusion matrix and accuracy/sens/spec for best LDA model with matching vars to logistic model

From our ROC curve, the best cut-off looks to be between 0.1 and 0.2

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 matrices

#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")

## [1] "Confusion matrix for LDA with all Vars"

conf.lda\_all

## test.lda.y  
## class.lda\_all no yes  
## no 6213 241  
## yes 584 611

#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")

## [1] "Confusion matrix for LDA with some Vars"

conf.lda\_step

## test.lda.y  
## class.lda\_step no yes  
## no 6053 223  
## yes 744 629

#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")

## [1] "Confusion matrix for QDA with some Vars"

conf.qda\_step

## test.lda.y  
## class.qda\_step no yes  
## no 6065 246  
## yes 732 606

#Accuracy of LASSO and Stepwise  
print("Overall accuracy for LDA w/ all vars, LDA w/ some vars, and QDA respectively")

## [1] "Overall accuracy for LDA w/ all vars, LDA w/ some vars, and QDA respectively"

sum(diag(conf.lda\_all))/sum(conf.lda\_all)

## [1] 0.8921428

sum(diag(conf.lda\_all))/sum(conf.lda\_all)

## [1] 0.8921428

sum(diag(conf.qda\_step))/sum(conf.qda\_step)

## [1] 0.8721401

#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

## Confusion Matrix and Statistics  
##   
## test.lda.y  
## class.lda\_all no yes  
## no 6213 241  
## yes 584 611  
##   
## Accuracy : 0.8921   
## 95% CI : (0.885, 0.899)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 0.1678   
##   
## Kappa : 0.5367   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9141   
## Specificity : 0.7171   
## Pos Pred Value : 0.9627   
## Neg Pred Value : 0.5113   
## Prevalence : 0.8886   
## Detection Rate : 0.8123   
## Detection Prevalence : 0.8438   
## Balanced Accuracy : 0.8156   
##   
## 'Positive' Class : no   
##

lda\_step\_0.15

## Confusion Matrix and Statistics  
##   
## test.lda.y  
## class.lda\_step no yes  
## no 6053 223  
## yes 744 629  
##   
## Accuracy : 0.8736   
## 95% CI : (0.8659, 0.8809)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4961   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8905   
## Specificity : 0.7383   
## Pos Pred Value : 0.9645   
## Neg Pred Value : 0.4581   
## Prevalence : 0.8886   
## Detection Rate : 0.7913   
## Detection Prevalence : 0.8205   
## Balanced Accuracy : 0.8144   
##   
## 'Positive' Class : no   
##

qda\_0.15

## Confusion Matrix and Statistics  
##   
## test.lda.y  
## class.qda\_step no yes  
## no 6065 246  
## yes 732 606  
##   
## Accuracy : 0.8721   
## 95% CI : (0.8644, 0.8795)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4831   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.8923   
## Specificity : 0.7113   
## Pos Pred Value : 0.9610   
## Neg Pred Value : 0.4529   
## Prevalence : 0.8886   
## Detection Rate : 0.7929   
## Detection Prevalence : 0.8251   
## Balanced Accuracy : 0.8018   
##   
## 'Positive' Class : no   
##

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

## Model Sensitivity Specificity Accuracy  
## 1 LDA All 0.9140797 0.7171362 0.8921428  
## 2 LDA Stepwise 0.8905399 0.7382629 0.8735782  
## 3 QDA Stepwise 0.8923054 0.7112676 0.8721401

# Random Forest

## Load the data

#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

## Train a Random Forest, tuning mtry and splitrule

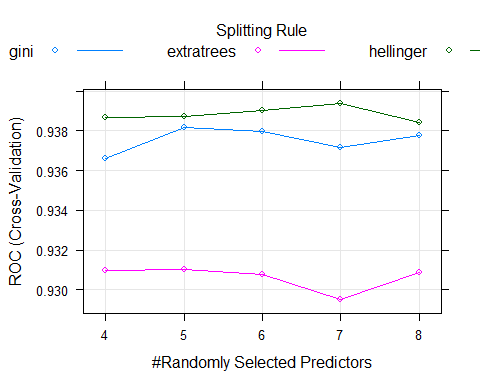
set.seed(1234)  
cv\_control <- trainControl(method="cv",   
 classProbs = TRUE,  
 savePredictions = TRUE,  
 summaryFunction = twoClassSummary,  
 num = 5)  
rf\_grid <- expand.grid(  
 mtry = 4:8,  
 splitrule = c("gini","extratrees", "hellinger"),  
 min.node.size = c(1)  
)  
fitRF <- train(y ~ .,   
 data = train,   
 method = "ranger",   
 metric = "ROC",  
 importance = "impurity",  
 trControl = cv\_control,  
 num.threads = 6,  
 num.trees = 30,  
 tuneGrid=rf\_grid)   
fitRF

## Random Forest   
##   
## 30593 samples  
## 23 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 24473, 24475, 24474, 24475, 24475   
## Resampling results across tuning parameters:  
##   
## mtry splitrule ROC Sens Spec   
## 4 gini 0.9366184 0.9707947 0.4189638  
## 4 extratrees 0.9309623 0.9773053 0.3355838  
## 4 hellinger 0.9386809 0.9741420 0.3922432  
## 5 gini 0.9381864 0.9673373 0.4609484  
## 5 extratrees 0.9310408 0.9707580 0.3943076  
## 5 hellinger 0.9387271 0.9676314 0.4509670  
## 6 gini 0.9379654 0.9643211 0.4882470  
## 6 extratrees 0.9307549 0.9675211 0.4233730  
## 6 hellinger 0.9390297 0.9653879 0.4765129  
## 7 gini 0.9371883 0.9624085 0.4952980  
## 7 extratrees 0.9295099 0.9644314 0.4362883  
## 7 hellinger 0.9393942 0.9634751 0.4870795  
## 8 gini 0.9377880 0.9627763 0.4908953  
## 8 extratrees 0.9308877 0.9628130 0.4577174  
## 8 hellinger 0.9384500 0.9632176 0.4985324  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 7, splitrule = hellinger  
## and min.node.size = 1.

I chose to tune 2 hyper parameters for Random Forest 1. mtry which represents the number of predictors considered when splitting a node in a tree 2. splitrule which determines the rule used for the actual splitting based on the above predictors Note: I set min.node.size to 1 as appropriate for classification

## Performance on Training Set

plot(fitRF)

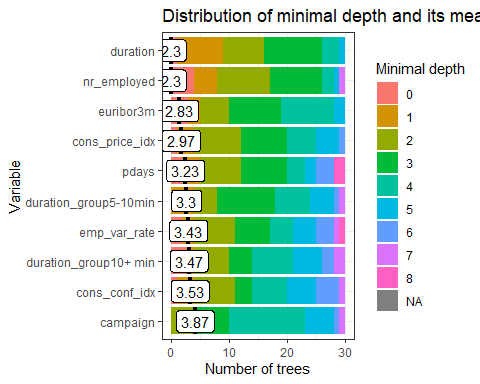


confusionMatrix(fitRF, positive = "yes")

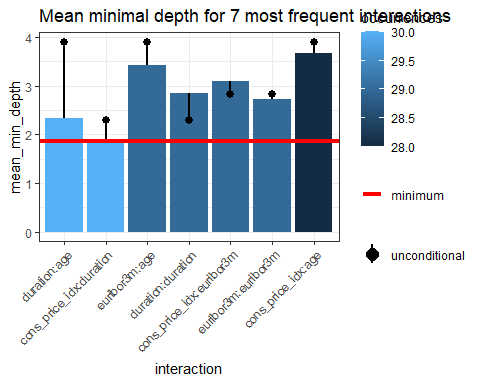
## Cross-Validated (5 fold) Confusion Matrix   
##   
## (entries are percentual average cell counts across resamples)  
##   
## Reference  
## Prediction no yes  
## no 85.6 5.7  
## yes 3.2 5.4  
##   
## Accuracy (average) : 0.9104

### Min Depth Distribution

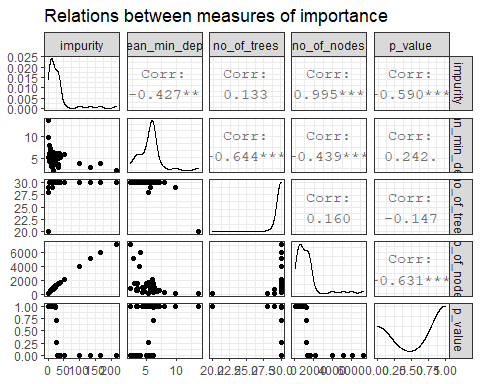
library(randomForestExplainer)  
forest\_frame <- min\_depth\_distribution(fitRF$finalModel)  
plot\_min\_depth\_distribution(forest\_frame)



### Mean minimal depth for most frequent interactions



# Other measures of importance



## ranger variable importance  
##   
## only 20 most important variables shown (out of 54)  
##   
## Overall  
## duration 100.00  
## age 76.06  
## euribor3m 61.59  
## campaign 46.51  
## housingyes 24.41  
## maritalmarried 18.30  
## educationuniversity.degree 17.50  
## educationhigh.school 16.29  
## day\_of\_weekthu 16.22  
## Age\_Grp32-37 15.88  
## day\_of\_weekwed 15.72  
## maritalsingle 15.69  
## cons\_price\_idx 15.49  
## day\_of\_weektue 15.35  
## loanyes 15.26  
## day\_of\_weekmon 15.06  
## cons\_conf\_idx 14.88  
## jobtechnician 14.56  
## Age\_Grp38-47 13.75  
## jobblue-collar 12.25

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; being insensitive to skew. CITATION: <https://www3.nd.edu/~nchawla/papers/DMKD11.pdf> \* CITATION: <https://medium.com/@evgeni.dubov/classifying-imbalanced-data-using-hellinger-distance-f6a4330d6f9a>

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")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 6561 408  
## yes 236 444  
##   
## Accuracy : 0.9158   
## 95% CI : (0.9094, 0.9219)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 2.308e-15   
##   
## Kappa : 0.5335   
##   
## Mcnemar's Test P-Value : 1.602e-11   
##   
## Sensitivity : 0.52113   
## Specificity : 0.96528   
## Pos Pred Value : 0.65294   
## Neg Pred Value : 0.94146   
## Prevalence : 0.11139   
## Detection Rate : 0.05805   
## Detection Prevalence : 0.08890   
## Balanced Accuracy : 0.74320   
##   
## 'Positive' Class : yes   
##

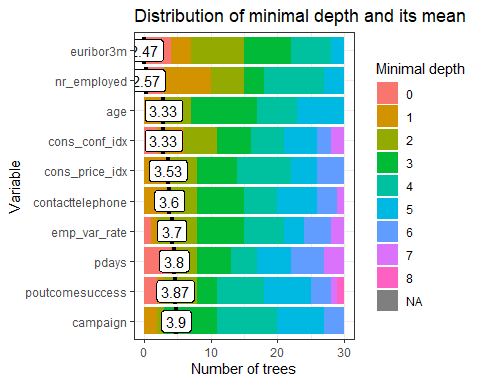
Using the default cutoff, our random forest gets a test accuracy of 0.9158, with Sensitivity 0.52113 and Specificity 0.96528.

Testing without duration

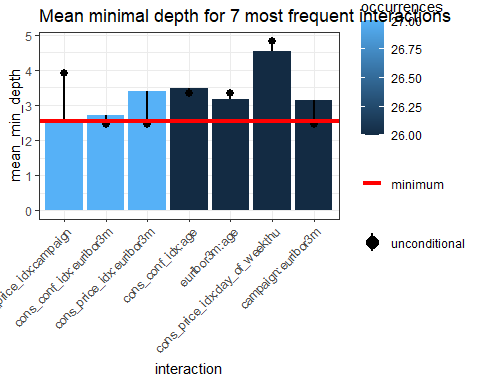
# set up train2/test2 to explore modeling without duration  
train2 <- train %>% dplyr::select(c(-duration, -duration\_group))  
test2 <- test %>% dplyr::select(c(-duration, -duration\_group))  
  
#train <- train2  
#test <- test2  
  
set.seed(1234)  
cv\_control <- trainControl(method="cv",   
 classProbs = 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\_2 <- train(y ~ .,   
 data = train2,   
 method = "ranger",   
 metric = "ROC",  
 importance = "impurity",  
 trControl = cv\_control,  
 num.threads = 6,  
 num.trees = 30,  
 tuneGrid=rf\_grid)   
fitRF\_2

## Random Forest   
##   
## 30593 samples  
## 21 predictor  
## 2 classes: 'no', 'yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 24473, 24475, 24474, 24475, 24475   
## Resampling results across tuning parameters:  
##   
## mtry splitrule ROC Sens Spec   
## 4 gini 0.7785654 0.9810203 0.2401622  
## 4 extratrees 0.7784767 0.9822341 0.2363503  
## 4 hellinger 0.7827989 0.9812410 0.2445679  
## 5 gini 0.7758389 0.9777467 0.2607116  
## 5 extratrees 0.7753017 0.9793650 0.2454442  
## 5 hellinger 0.7819047 0.9774525 0.2566038  
## 6 gini 0.7720719 0.9767535 0.2739274  
## 6 extratrees 0.7748643 0.9764224 0.2633534  
## 6 hellinger 0.7730089 0.9764593 0.2683487  
## 7 gini 0.7727809 0.9751352 0.2730434  
## 7 extratrees 0.7699251 0.9761650 0.2648262  
## 7 hellinger 0.7744577 0.9754294 0.2683461  
## 8 gini 0.7727039 0.9735168 0.2733383  
## 8 extratrees 0.7686904 0.9747673 0.2657077  
## 8 hellinger 0.7705509 0.9745467 0.2706943  
##   
## Tuning parameter 'min.node.size' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 4, splitrule = hellinger  
## and min.node.size = 1.

forest\_frame\_2 <- min\_depth\_distribution(fitRF\_2$finalModel)  
plot\_min\_depth\_distribution(forest\_frame\_2)

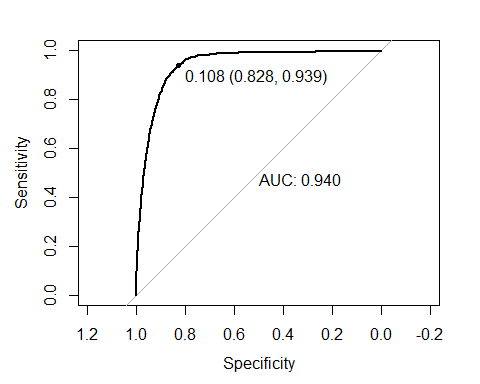


plot\_min\_depth\_interactions(fitRF\_2$finalModel, k=7)

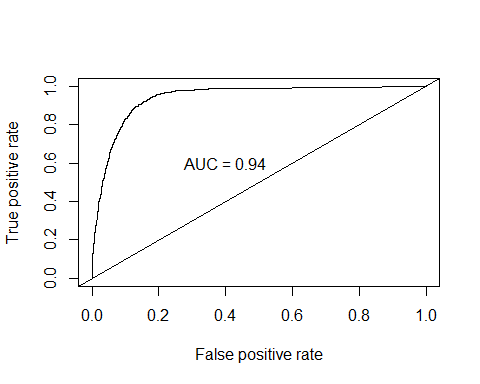


# ROC Curve and Optimal Cutoff

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")



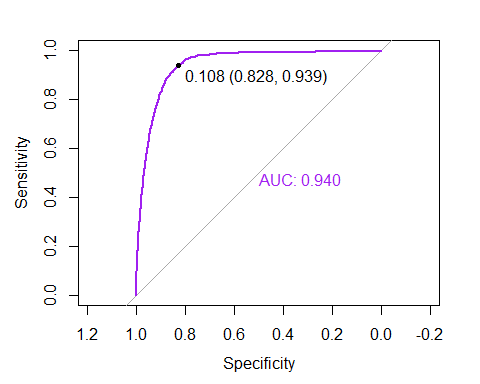
# If we wanted cutoffs for specific specificities we specifically specify, we could do THIS:  
#roc.data <- data\_frame(  
# thresholds = res.roc$thresholds,  
# sensitivity = res.roc$sensitivities,  
# specificity = res.roc$specificities  
#)  
# Then we can get the cutoff for specificity = <something> 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 = ""))



The most balanced cutoff for this model is 0.108

## ROC Curve and Optimal Cutoff

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")



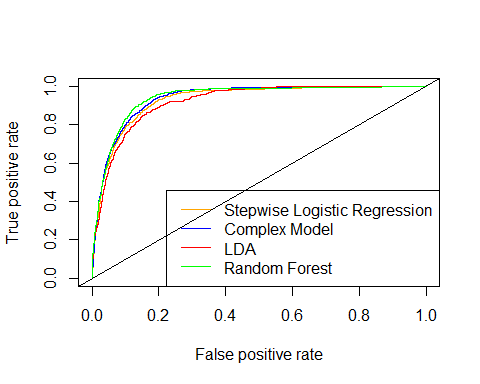
# Get the best cutoff for balancing Sensitivity and Specificity  
cutoff <- coords(roc.randomforest, "best", ret="threshold", transpose = FALSE)$threshold  
  
# Predict using the best cutoff and confirm with a Confusion Matrix  
predicted.classes.balanced <- factor(  
 ifelse( fitRF.predictions.prob$yes > cutoff, "yes", "no"), levels=c("no","yes"))  
confusionMatrix(predicted.classes.balanced, test$y, positive="yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 5631 52  
## yes 1166 800  
##   
## Accuracy : 0.8408   
## 95% CI : (0.8324, 0.8489)  
## No Information Rate : 0.8886   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.4882   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.9390   
## Specificity : 0.8285   
## Pos Pred Value : 0.4069   
## Neg Pred Value : 0.9908   
## Prevalence : 0.1114   
## Detection Rate : 0.1046   
## Detection Prevalence : 0.2570   
## Balanced Accuracy : 0.8837   
##   
## 'Positive' Class : 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

# Model Comparison

#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, 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)



## 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(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\_bar(mapping = aes(x = month, fill = y)) +  
 ggtitle("Distribution of 'y' by duration")  
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(df) + geom\_histogram(mapping = aes(x = duration, fill = y)) +  
 ggtitle("Distribution of 'y' by duration")  
  
#ggplot(df3, aes(x=duration\_group, y=duration\_group\_conv)) + geom\_point() + facet\_wrap(~ y)  
ggplot(df, aes(x = month , y = euribor3m, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("euribor3m by month")  
  
ggplot(df, aes(x=contact, y=duration, color = y)) +   
 geom\_point(size=1, shape="circle") +   
 ggtitle("contact vs duration")  
  
ggplot(df, aes(x=contact, y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("contact vs duration") +  
 facet\_grid(~y)  
  
  
ggplot(df, aes(x=default, y=duration, color = y)) +   
 geom\_point(size=1, shape="circle") +   
 ggtitle("contact vs duration")  
  
ggplot(df, aes(x=default, y=duration, fill = y)) +   
 geom\_boxplot() +   
 ggtitle("default vs duration") +  
 facet\_grid(~y)  
  
ggplot(df) +   
 geom\_bar(mapping = aes(x=month, fill = y)) +   
 ggtitle("Number of 'y' by month") +  
 ylab("Cnt") + xlab("month")  
  
plot(default~month,data=df,col=c("purple","green"))  
  
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",print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(1.2,1.2), avg="threshold", lwd=3)  
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", print.cutoffs.at = c("0.1", "0.15", "0.2"))  
  
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", print.cutoffs.at = c(0.1, 0.15, 0.2))  
  
#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)  
#finding optimal cut-off for step model  
perf.acc<-performance(results.step,"acc") #find list of accuracies  
perf.fpr<-performance(results.step,"fpr") # find list of fp rates  
perf.fnr<-performance(results.step,"fnr") # find list of fn rates  
  
acc.rocr<-max(perf.acc@y.values[[1]]) # accuracy using rocr  
  
#find cutoff list for accuracies  
cutoff.list.acc <- unlist(perf.acc@y.values[[1]])  
  
#find optimal cutoff point for accuracy  
optimal.cutoff.acc<-cutoff.list.acc[which.max(perf.acc@y.values[[1]])]  
  
#find optimal cutoff fpr, as numeric because a list is returned  
optimal.cutoff.fpr<-which(perf.fpr@x.values[[1]]==as.numeric(optimal.cutoff.acc))  
  
# find cutoff list for fpr  
cutoff.list.fpr <- unlist(perf.fpr@y.values[[1]])  
# find fpr using rocr  
fpr.rocr<-cutoff.list.fpr[as.numeric(optimal.cutoff.fpr)]  
#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  
  
  
cutoff<-0.108  
class.complex\_.108<-factor(ifelse(complex.pred>cutoff,"yes","no"),levels=c("no","yes"))  
  
#Confusion Matrix  
conf.complex\_.108<-table(class.complex\_.108,test$y)  
conf.complex\_.108  
complex\_.108<-confusionMatrix(conf.complex\_.108, positive = "yes")  
complex\_.108  
#Training Set  
train.lda.x <- train[ , sapply(train, is.numeric)]  
train.lda.y <- train$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)  
  
varImp(fitRF)  
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")  
# 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 <- 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\_2 <- train(y ~ .,   
 data = train2,   
 method = "ranger",   
 metric = "ROC",  
 importance = "impurity",  
 trControl = cv\_control,  
 num.threads = 6,  
 num.trees = 30,  
 tuneGrid=rf\_grid)   
fitRF\_2  
  
forest\_frame\_2 <- min\_depth\_distribution(fitRF\_2$finalModel)  
plot\_min\_depth\_distribution(forest\_frame\_2)  
  
plot\_min\_depth\_interactions(fitRF\_2$finalModel, k=7)  
prediction.probabilities <- fitRF.predictions.prob$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, 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)