Bank Marketing Analysis Project

D. Bracy , H.H. Nguyen, S.Purvis

04/02/2020

# I. Introduction

In this project, we will study different approaches to predict the success of bank telemarketing the Bank Marketing data set [1].

The retail banking industry provides financial services to families and individuals. Banks’ main functions are threefold; they issue credit in the forms of loans and credit lines, provide a secure location to deposit money, and allow a mechanism to manage finances in the form of checking and savings accounts. This analysis will focus specifically on the influential factors from direct marketing campaigns managed by a Portuguese banking institution in an attempt to get secure commitment for term deposits. Understanding not only which marketing campaigns were most effective, but also the timing of the campaign and the socioeconomic demographics will allow the retail banking industry to further target and tune their approach to securing term deposits.

Bank Marketing data from this data set were used to address two project objects:

1. Display the ability to perform EDA and perform a logistic regression analysis and provide interpretation of the regression coefficients including hypothesis testing, and confidence intervals.
2. With a simple logistic regression model as a baseline, perform additional competing models to improve on prediction performance metrics.

# II. Data Description

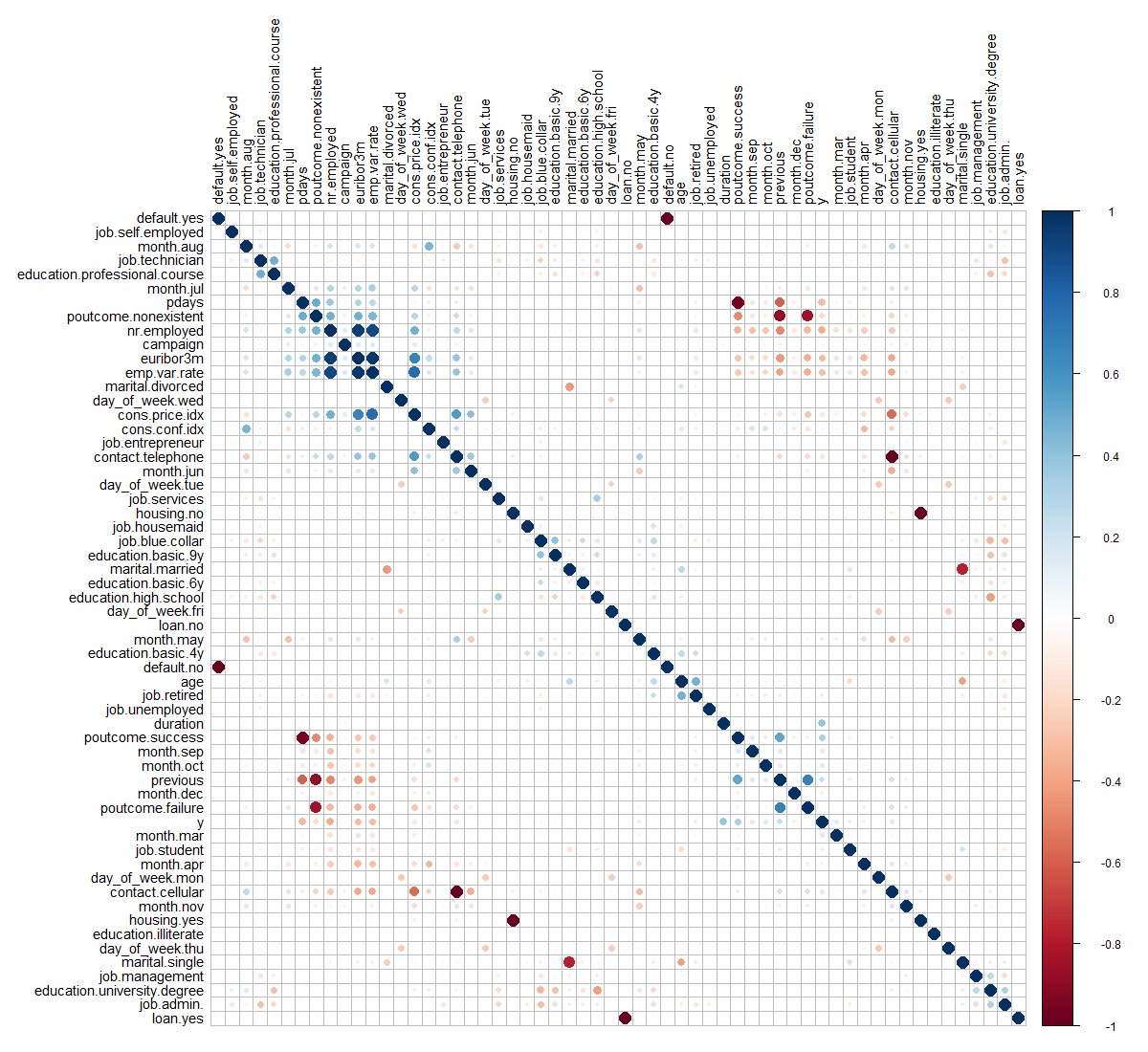
The team was provided a substantial marketing dataset. It was comprised of categorical and continuous variables and a resulting binary result (Y/N). The data ranges from May 2008 to November 2010. As described in the table below, we have equal counts of numeric and categorical variables. There are demographics, data related to the depth and breadth of the marketing campaign, and market indicators included in this set.

|  |  |  |
| --- | --- | --- |
| Variable | Type | Description |
| Age | Numeric | Age of the Individual |
| Job | Categorical | Type of job held |
| Marital | Categorical | Marital Status |
| Education | Categorical | Level of Education of individual |
| Default | Categorical | Y/N/Unknown on whether the individual has credit in default |
| Housing | Categorical | Y/N/Unknown on whether the individual has a housing loan |
| Loan | Categorical | Y/N/Unknown on whether the individual has a personal loan |
| Contact | Categorical | Contact Communication Type |
| Month | Categorical | Month of last contact |
| Day\_of\_Week | Categorical | Day of the week of last contact – Weekdays Only |
| Duration | Numeric | Duration of last contact, in seconds. \*should only be used as a benchmark, since it can’t be known beforehand |
| Campaign | Numeric | Number of contacts performed during this campaign for this client |
| Pdays | Numeric | Number of days that passed by after a client was contacted from a previous campaign (999 means not contacted previously) |
| Previous | Numeric | Number of contacts performed before this campaign for this client |
| Poutcome | Categorical | Outcome of previous marketing campaign |
| Emp.var.rate | Numeric | Employment variation rate – quarterly indicator |
| Cons.price.idx | Numeric | Consumer Price Index – monthly indicator |
| Cons.conf.idx | Numeric | Consumer confidence index – monthly indicator |
| Euribor3m | Numeric | Euribor (Euro Interbank Offered Rate) 3 month rate – daily indicator |
| Nr.employed | Numeric | Number of employees – quarterly indicator |
| Y | Binary | Did Client subscribe to a term deposit |

# III. Exploratory Data Analysis (EDA)

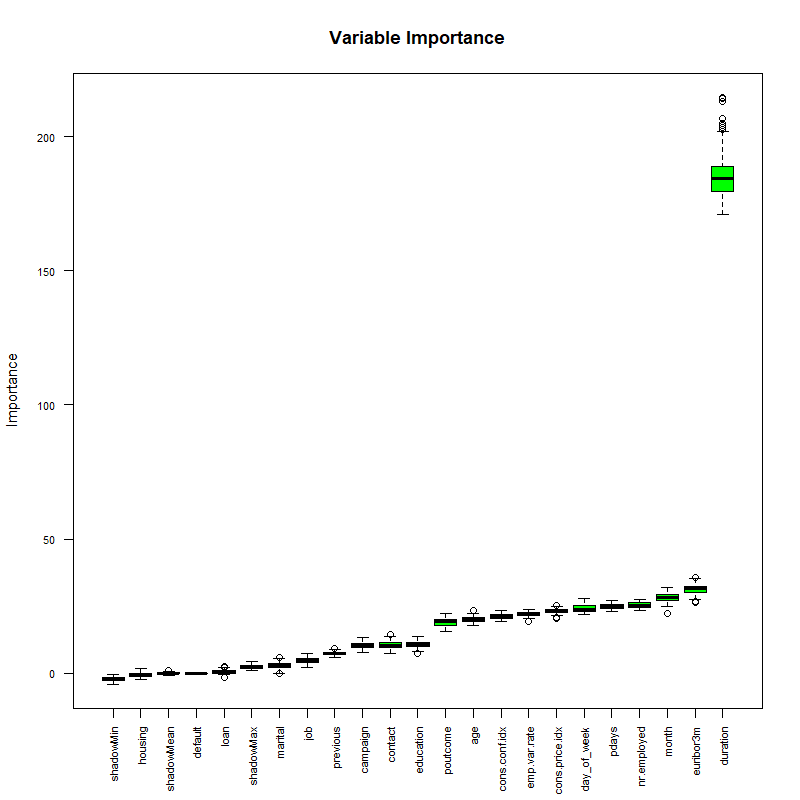
During our preliminarily assessment of the data, we first evaluated the impact of missing data. We found that technically we did not have any missing data, but we were provided a fair amount of unknown values. The original dataset has thousands of observations. Because we did not feel limited by our dataset size, we decided to exclude any observation that has an unknown record recorded in any of the variables. This left us over 30,000 complete observations to work with. Next, we evaluated the normality of all continuous variables. We employed boxplots and barcharts to visually inspect distribution. We observed right skewness in Age, but we can rely on central limit theorem for normality assumptions in spite of visual indications.

We next investigated correlation. As shown in the correlation plot,



there are some common sense correlations, particularly between specific factors in variables. For example, cons.price.idx & emp.var.rate are positively correlated. This is reasonable as both are market indicators that would naturally tether together.

As referenced above, we excluded duration from the model selection process. It is an indicator variable that can be utilized as a benchmark, but is not known before the calls are made. Additionally, we ran a test of variable importance (Boruta) that provided some additional insight into which variables are “important” and in what order. We noted that Marital Status, Loan, Default and Housing are all relatively less important than other variables. We will revisit this insight as we approach interactions.



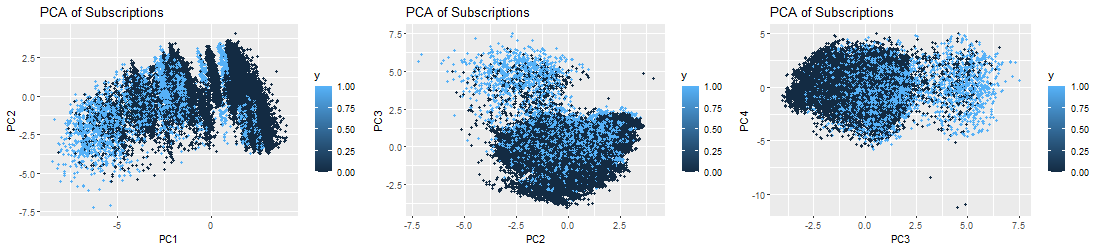
The final iteration that we did was to separate the data into a training and test set for all assessments going forward, consistently using the same split upon each interaction.

We now consider a dimension reduction method Principal Component Analysis (PCA).

### Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a tool for unsupervised learning. It is a common approach for deriving a low-dimensional set of features from a large set of variables. PCA creates new uncorrelated variables from a group of variables and the information of these new variables can be used to understand the relationship among the original variables and for other analyses such as regression model and classification that we will mention later.

First, we perform a PCA on the bank marketing data set after scaling the variables to have standard deviation. Then we plot the first few principal components to visualize the data. The three PCA plots don’t show as much separation as we would hope for. We can expect our prediction algorithms to struggle a little bit in providing accurate results, due to the tightly entangled results of our subscription (y) response variable.



# IV. Logistic Regression Analysis

## Building the model

### Feature Selection

In constructing our logistic regression model, we first included all variables in the model. We moved forward manually, by first reducing the initial model based on the the variables deemed insignificant. Those variables included Marital, Education, Default, Loan and Housing. All showed insignifance in predicting subscriptions.  
### Assessing the fit When assessing fit, we determined that we would include AIC, AUC and look at the specificity/sensitivity. Our “smaller model” produced an AIC of 14551, and AUC of 0.80888 and specificity of 0.933. ### Assumption Checking We evaluated the Cook’s D Plot and Leverage plot to assess assumption violations. Cook’s D did not return a value above 0.0020, showing that there are no outliers to consider. Leverage also supports this, with only one value above 0.020. ### Residual Diagnostic Plots Optional Residual Plots ## Interpretation The optimal model is best described as:

Recall that we are calculating toward a yes or no outocme. Each coefficient described adds or detracts from the total. The y-intercept is set at -291.40 (with confidence interval of [-355.9, -226.9]). When looking at the levels of Job, blue collar and services are both significant and detrimental toward the overall value (both at -0.20 with CI [-0.34, -0.06] and [-0.29, 0.23] respectively). Alternatively, retired and student both signifcantly impact the outcome positively. Retired increases the value by 0.25 (CI of [0.06, 0.43]) while students increase by 0.27 (with CI of [0.04, 0.50]). The method of contact is also of importance, where telephone calls (versus cell phone contact) decreases the overall outcome by -0.81 (with CI of [-0.98, -0.65]). Specific Months had disparate inpact. June produces the lowest coefficient with -0.72 and confidence interval of (-0.98, -0.45). March provided the highest coefficient of 1.64 with confidence intervals of (1.35, 1.92). The consumer price index increases the overall outcome score by 2.44 points for each point increase in the index number (CI of [1.97, 2.90]).

## Logistic Regression Conclusion

The simple model produced in logistic regression highlights a few key factors among the variables that indicate importance. As the presumped goal is achieving the Yes result, specific items could be focused on in future solicitation efforts to employ resources efficiently. Students and Retired persons both tested as factors worth expanding on. Additionally, cell phone contact would be advised. Specific months proved significant, but that is likely due to the index numbers more than the actual month itself. We must point out that this is an observational study, so no true conclusions can be made from this model about the larger population or around causality. The findings are nonetheless interesting.

# V. Alternative Models

With our simple logistic regression model as a baseline, the team performed additional competing models evaluations to improve on prediction performance metrics.

## Adding Complexity to Logistic Regression

When inserting complexity in the model, the team revisited the Boruta anaylsis of variable importance. Euribor3M was the most important single variable identified in the assessment. This variable, as mentioned above, is a lending rate banks use when specifically lending on loans with a 3 month maturity. We applied this to month as the variable is time based. Additionally, we included interactions with pdays, age and nr.employed believing we may find that socioeconomic factors and timing would impart significance. Finally, we included an interaction between pdays and campaign with success. Our final model produced an AIC of 14579, showing that interactions in this case didn’t add real value. Furthermore, they would require a more cumbersome interpretation of the model without clear influence by single variables.

## Looking at Continous Predictors

1. Create another competing model using just the continuous predictors and use LDA or QDA.

### LDA

## Non-parametric Models

1. (Optional) Use a nonparameteric model approach as a competing model. Random forest or decision tree for predictors that are both categorical and continuous or a k-nearest neighbors approach if just working with continuous predictors.

### KNN

KNN has an 88% overall accuracy, with 97% sensitivity when performed outright. However, specificity suffers with only 29.6% accuracy. Scaling the data alone did not improve the performance of KNN. We then fed KNN algorithm the limited set of predictors used in Logistic Regression, which also did not improve performance. Hypertuning the data maintained the overall accuracy of 88%. It improved specificity at the cost of sensitivity though. Sensitivity is instead 68.3% whereas specificity is now 89.7%.

### Decision Tree

### Random Forest

# VI. Model Performance Comparisons

1. Record the predictive performance metrics from your simple, highly interpretable model from Objective 1.
2. Provide a summary table of the performance across the competing methods. Summarize the overall findings. A really great report will also give insight as to why the “best” model won out. This is where a thorough EDA will always help. Logistical Considerations.
3. Make sure it is clear how many models were created to compete against the one in Objective 1. Make note of any tuning parameters that were used and how you came up with them (knn and random forest logistics) Required

* Setup a table here to compare model performance metrics

# VII. Conclusion

1. Overall report of the error metrics on a test set or CV run. Also if the two best models have error rates of .05 and .045, can we really say that one model is outperforming the other? For the ambitious, McNemar’s test could be helpful in answering that.
2. The conclusion should reprise the questions and conclusions of objective 2 with recommendations of the final model, what could be done to help analysis and model building in the future, and any insight as to why one method outshined all the rest if that is indeed the case. If they all are similar why did you go with your final model?

# 

# Acknowledgement

# References

[1] *Bank Marketing Data Set* [https://archive.ics.uci.edu/ml/datasets/Bank%20Marketing#](https://archive.ics.uci.edu/ml/datasets/Bank%20Marketing)  
[2] Albright, W. L. Winston, *Business Analytics - Data Analysis and Decision Making*, 7th Edition, 2019.  
[3] Anderson et al., *Statistics for Business & Economics*, Cengage, 2020.  
[4] James et al., *An Introduction to Statistical Learning with Application in R*, Springer, 2017.  
[5] T. Hastie et al., *The Elements of Statistical Learning, Data Mining, Inference*, and Prediction, 2nd Edition, Springer, 2017.  
[6] B. W. Lindgren, *Statistical Theory*, 3th Edion, MacMillan Publishing, 1976.  
[7] D. Montgomery, E. A. Peck, G. G. Vining, *Introdution to Linear Regression Analysis*, 5th Edion, John Wiley & Sons, 2012.  
[8] Ramsey and Schafer, *The Statistical Sleuth, A Course in Methods of Data Analysis*, 3rd Edition, Cengage, 2013.

# Appendix

## Code Section

### Feature Engineering

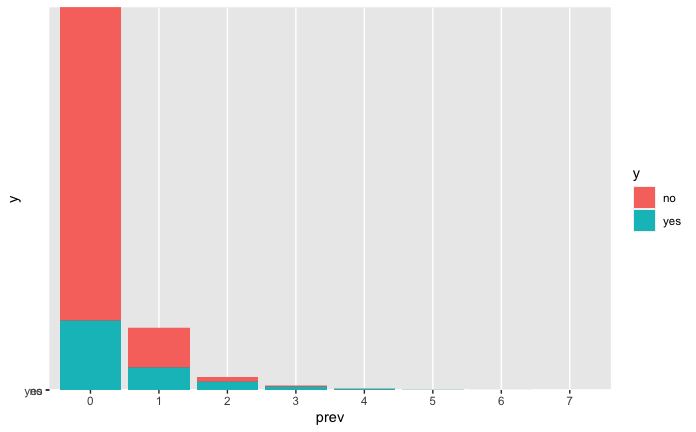
# read in 'Bank Additional Full' file  
bankfull = read.csv("./DataSets/bank-additional-full.csv",header = TRUE, sep = ";")  
  
# convert "unknown" values to NA and view percentage of missing values  
bankfull[bankfull == "unknown"] <- NA   
  
plotNAs(bankfull)



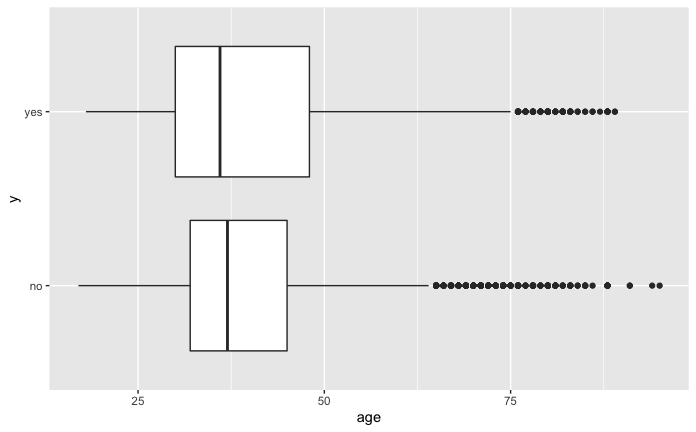
# Remove duration from model, as this isn't known until 'y' is known  
bankfull <- bankfull %>% dplyr::select(!duration)  
  
# Drop NAs  
bankfull <- bankfull %>% drop\_na()  
bankfull$job <- droplevels(bankfull$job, 'unknown')  
bankfull$loan <- droplevels(bankfull$loan, 'unknown')  
bankfull$default <- droplevels(bankfull$default, 'unknown')  
bankfull$education <- droplevels(bankfull$education, 'unknown')  
bankfull$housing <- droplevels(bankfull$housing, 'unknown')  
bankfull$marital <- droplevels(bankfull$marital, 'unknown')  
  
# Onehot encode categorical variables to binary:  
dmy <- dummyVars(" ~ .", data = bankfull)  
trsf <- data.frame(predict(dmy, newdata = bankfull))  
  
# Remove binary encoded response  
trsf$y <- ifelse(trsf$y.no == 1, 0, 1)  
bankbin <- subset(trsf, select = -c(y.no, y.yes))  
  
# Clean up environment variables:  
rm(dmy, trsf)  
  
# Split the data into training and test set  
set.seed(115)  
trainIndices = sample(1:dim(bankfull)[1],round(.8 \* dim(bankfull)[1]))  
  
# Build full test/train  
full.train = bankfull[trainIndices,]  
full.test = bankfull[-trainIndices,]  
  
# Build binary test/train  
bin.train = bankbin[trainIndices,]  
bin.test = bankbin[-trainIndices,]  
  
# Scale binary data   
scaledbin <- data.frame(scale(bankbin))  
scaledbin$y <- bankbin$y  
  
# Build scaled test/train  
scaled.train = scaledbin[trainIndices,]  
scaled.test = scaledbin[-trainIndices,]

### EDA

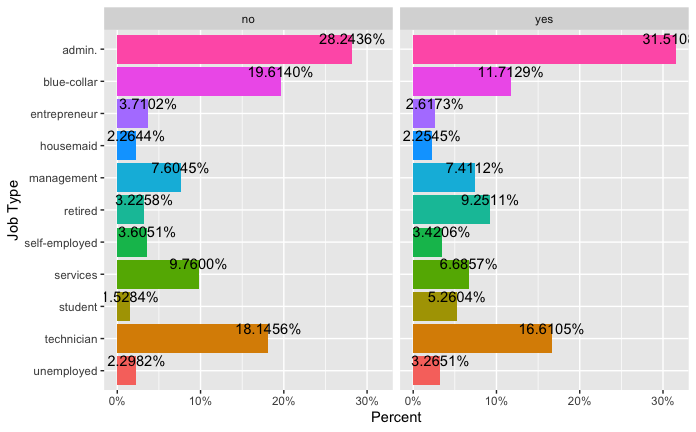
df <- bankfull # input dataframes for plots  
  
mutate(df, prev = as.factor(previous)) %>% ggplot(aes(prev, y, fill=y)) + geom\_col()



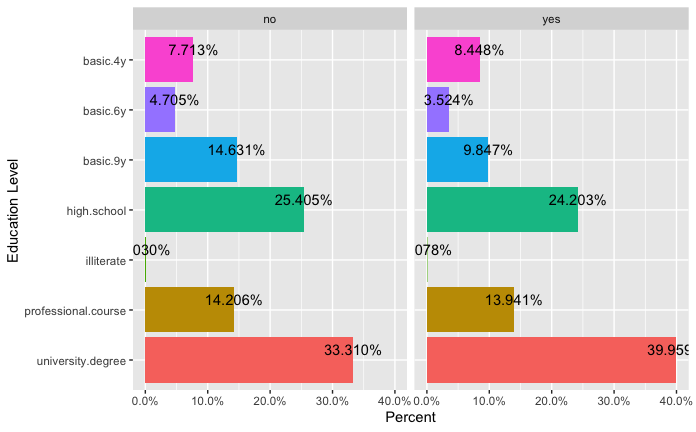
df %>% ggplot(aes(y, age)) + geom\_boxplot() + coord\_flip()



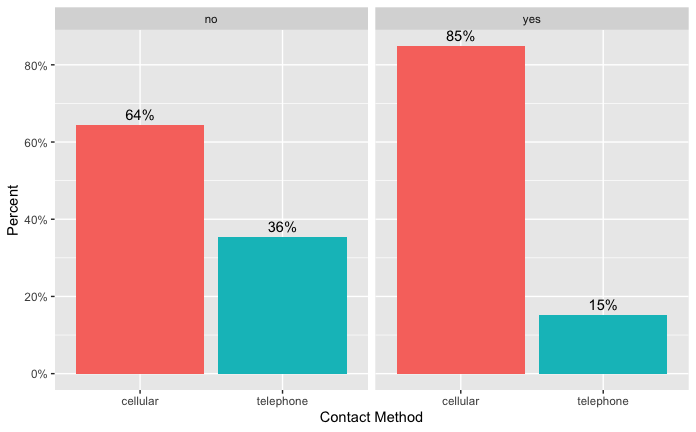
percentagePlot(df, fct\_rev(df$job), "Job Type") + coord\_flip()



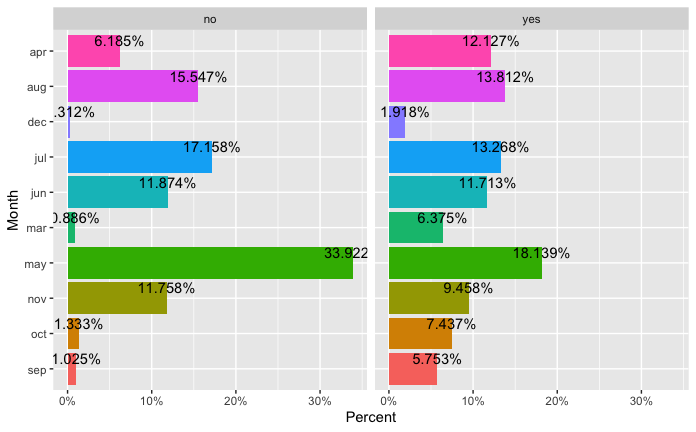
percentagePlot(df, fct\_rev(df$education), "Education Level") + coord\_flip()



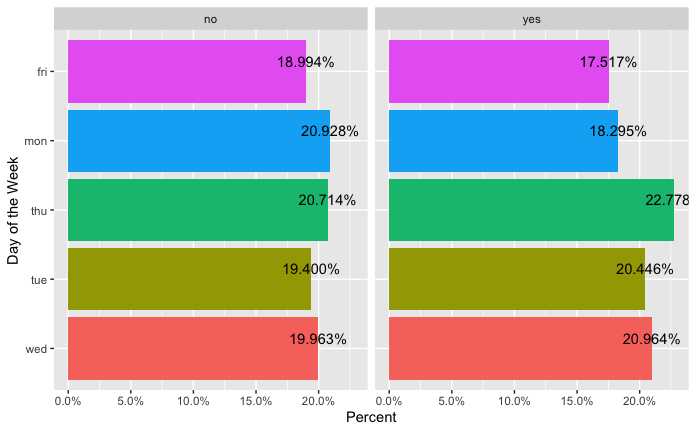
percentagePlot(df, df$contact, "Contact Method")



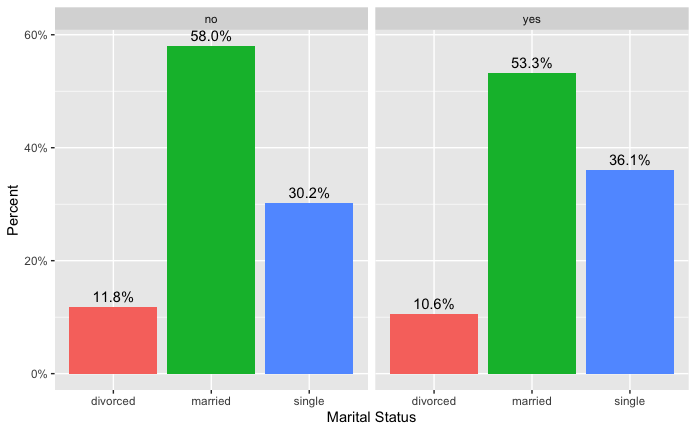
percentagePlot(df, fct\_rev(df$month), "Month") + coord\_flip()



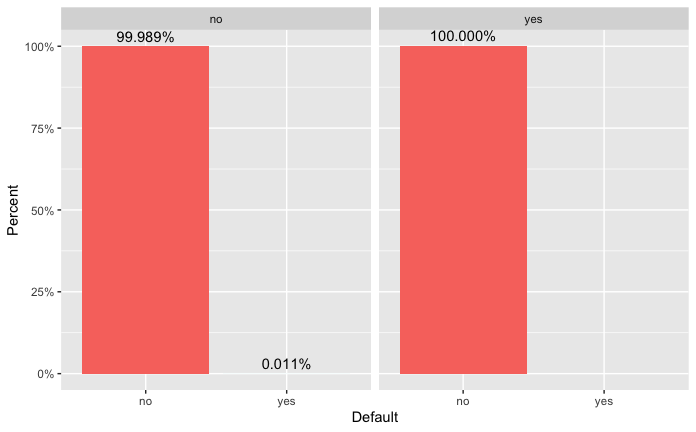
percentagePlot(df, fct\_rev(df$day\_of\_week), "Day of the Week") + coord\_flip()



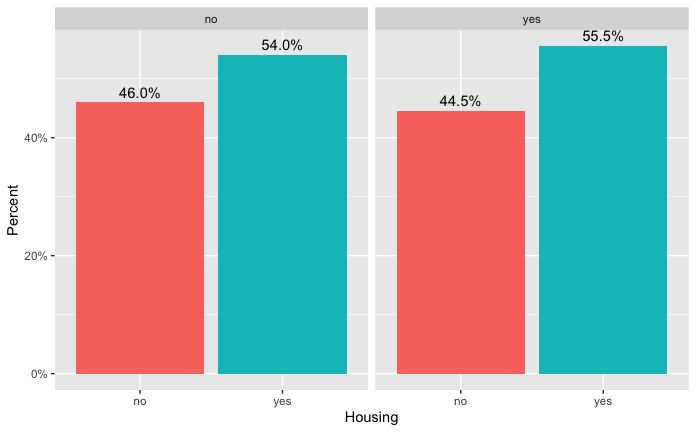
# Do we want to show the non-selected feature plots?  
percentagePlot(df, df$marital, "Marital Status")



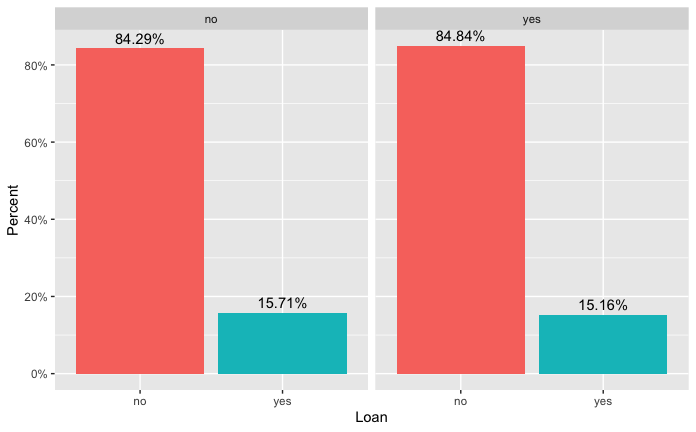
percentagePlot(df, df$default, "Default")



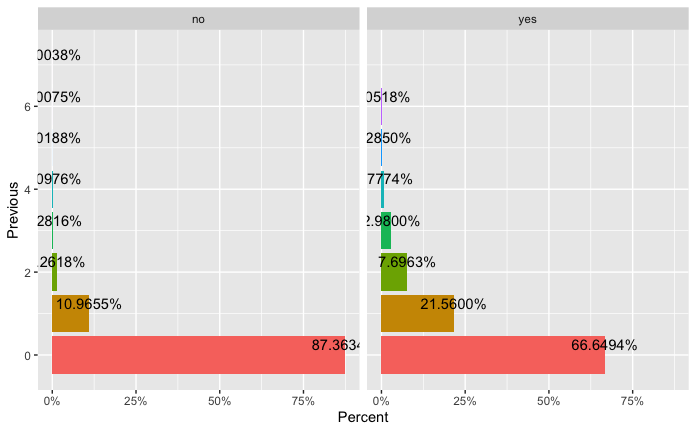
percentagePlot(df, df$housing, "Housing")



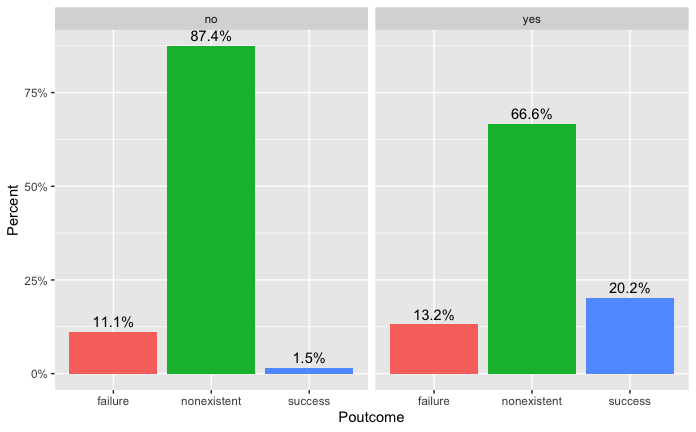
percentagePlot(df, df$loan, "Loan")



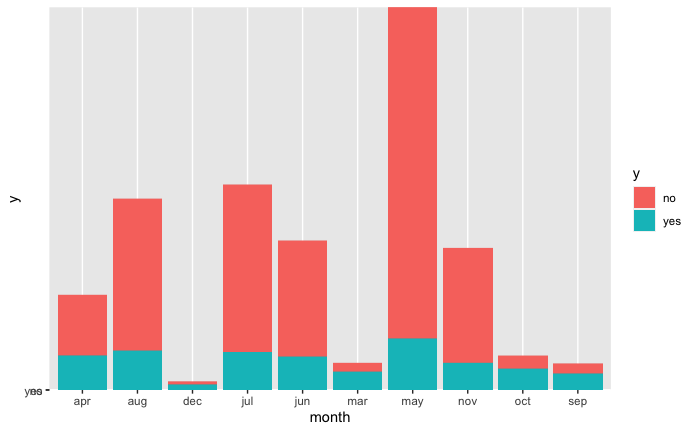
percentagePlot(df, df$previous, "Previous") + coord\_flip()



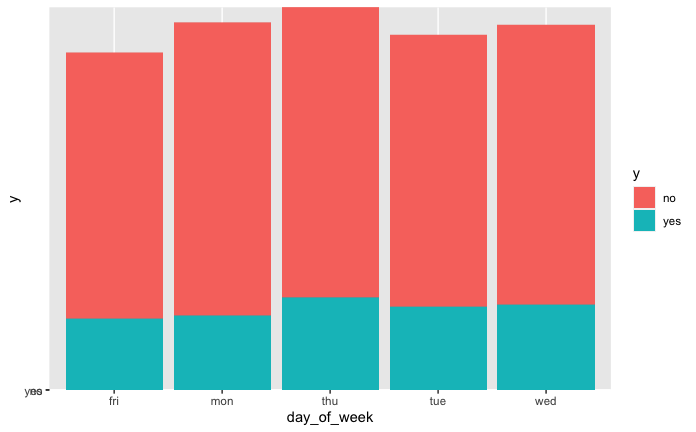
percentagePlot(df, df$poutcome, "Poutcome")



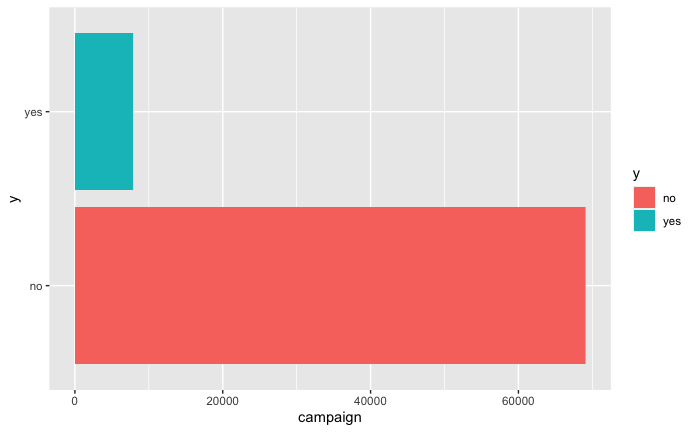
df %>% ggplot(aes(month, y, fill=y)) + geom\_col()



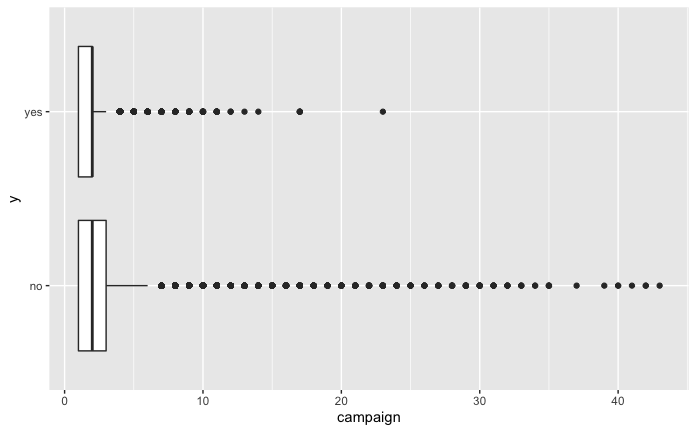
df %>% ggplot(aes(day\_of\_week, y, fill=y)) + geom\_col()



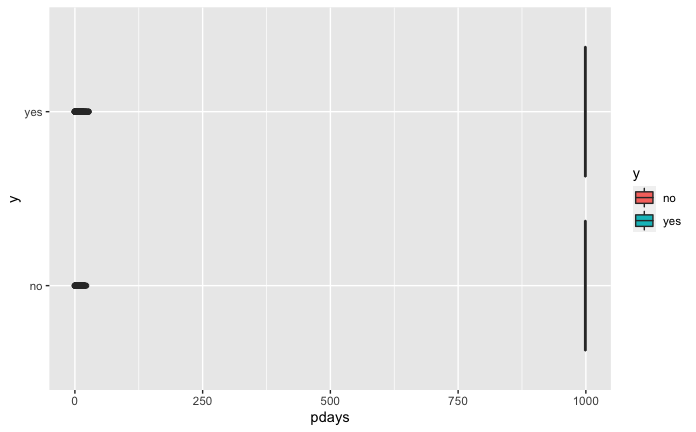
df %>% ggplot(aes(campaign, y, fill=y)) + geom\_col()



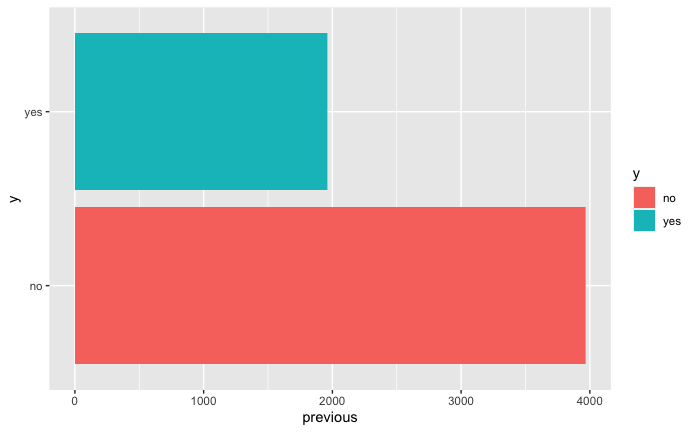
df %>% ggplot(aes(y, campaign)) + geom\_boxplot() + coord\_flip()



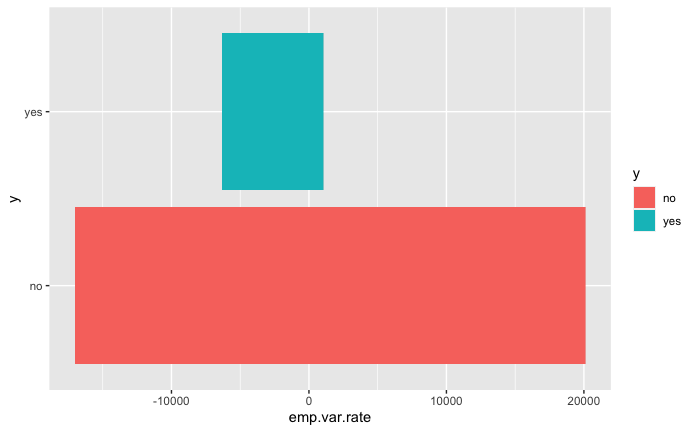
df %>% ggplot(aes(pdays, y, fill=y)) + geom\_boxplot()



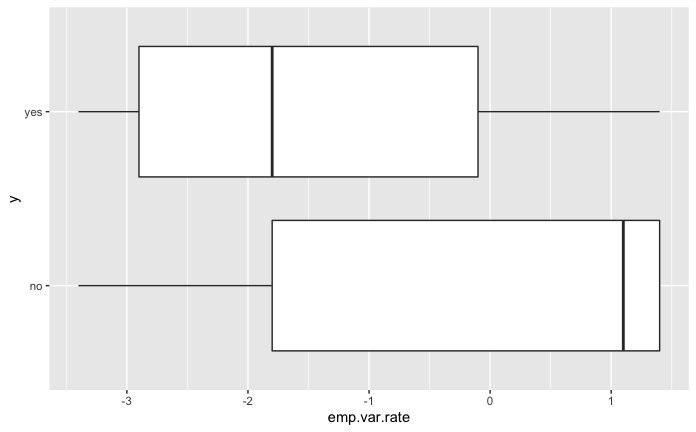
df %>% ggplot(aes(previous, y, fill=y)) + geom\_col()



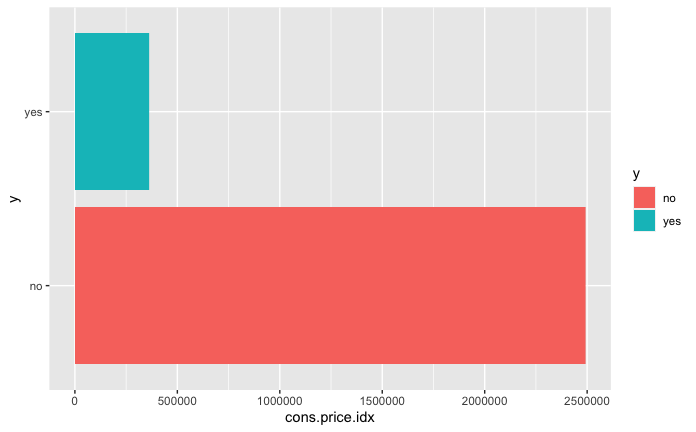
df %>% ggplot(aes(emp.var.rate, y, fill=y)) + geom\_col()



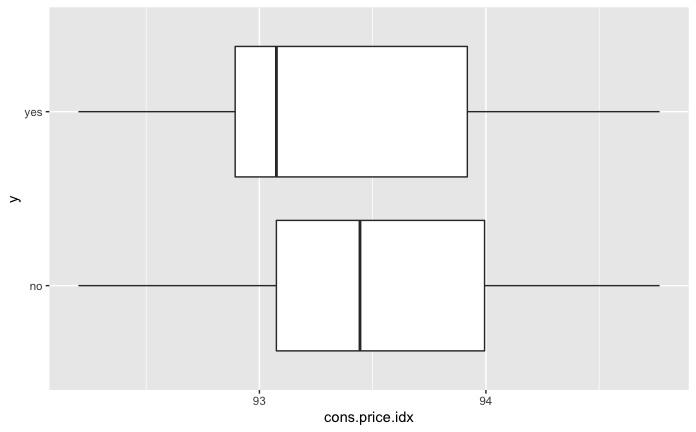
df %>% ggplot(aes(y, emp.var.rate)) + geom\_boxplot() + coord\_flip()



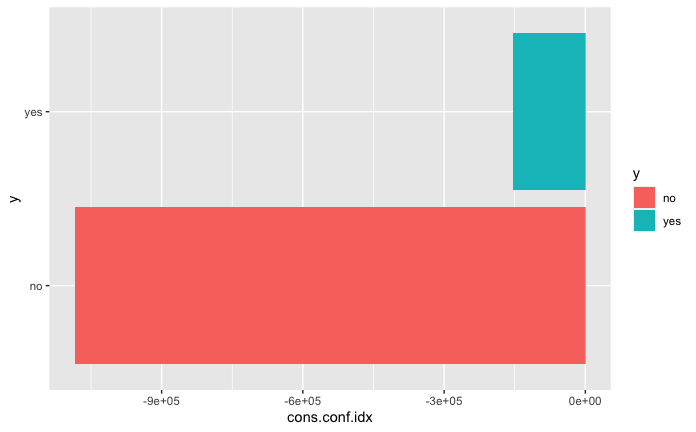
df %>% ggplot(aes(cons.price.idx, y, fill=y)) + geom\_col()



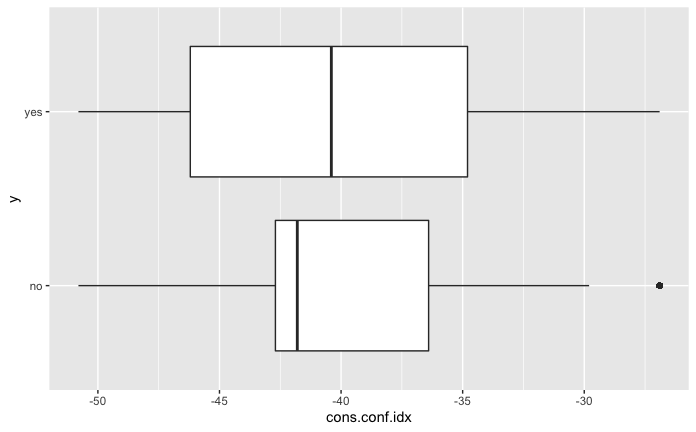
df %>% ggplot(aes(y, cons.price.idx)) + geom\_boxplot() + coord\_flip()



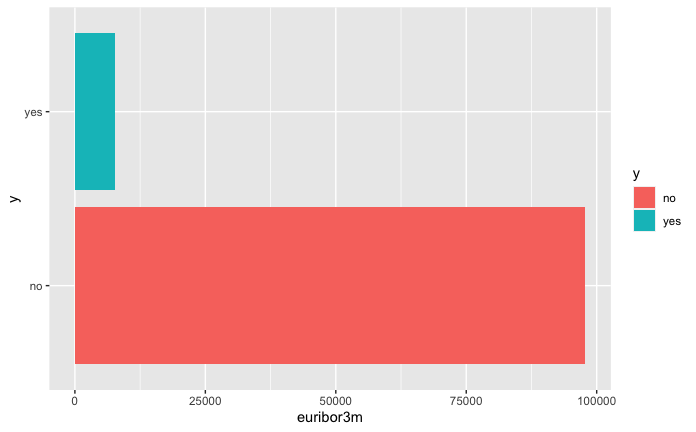
df %>% ggplot(aes(cons.conf.idx, y, fill=y)) + geom\_col()



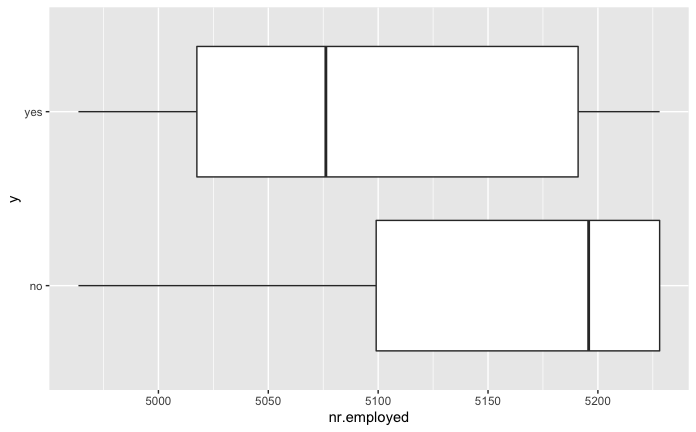
df %>% ggplot(aes(y, cons.conf.idx)) + geom\_boxplot() + coord\_flip()



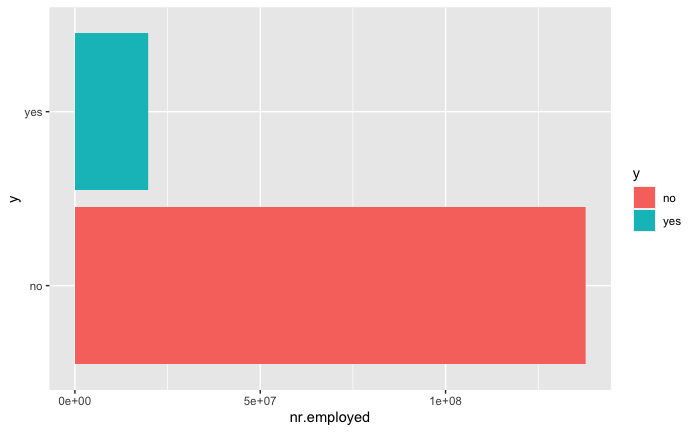
df %>% ggplot(aes(euribor3m, y, fill=y)) + geom\_col()



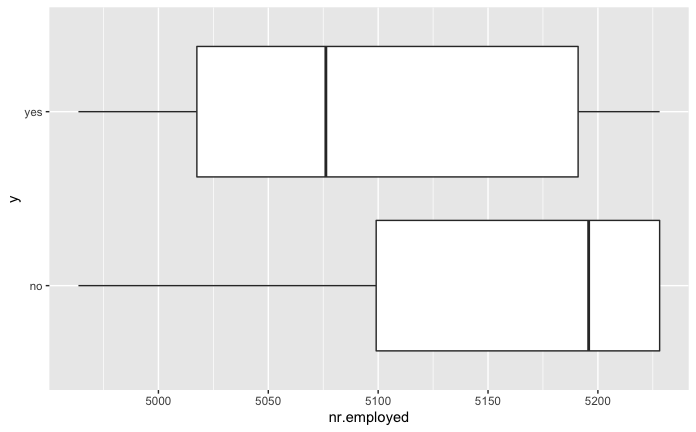
df %>% ggplot(aes(y, nr.employed)) + geom\_boxplot() + coord\_flip()



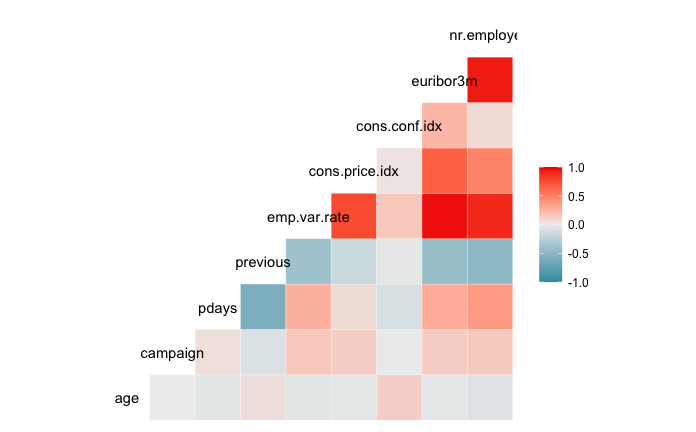
df %>% ggplot(aes(nr.employed, y, fill=y)) + geom\_col()



df %>% ggplot(aes(y, nr.employed)) + geom\_boxplot() + coord\_flip()



#additional EDA Graphics  
  
ggcorr(df)



# Build Correlation Plot  
buildCorrPlot(bankbin)  
  
# Build Pairs Plot  
png(height=800, width=800, pointsize=15, file="./figures/pairs.png")  
bankfull %>% keep(is.numeric) %>% ggpairs()  
dev.off()  
  
# Identify significant features  
boruta\_output <- Boruta(y ~ ., data=bankfull, doTrace=2)  
boruta\_signif <- names(boruta\_output$finalDecision[boruta\_output$finalDecision %in% c("Confirmed", "Tentative")])   
print(boruta\_signif)   
  
# Build Variable Importance Plot  
png(height=800, width=800, pointsize=15, file="./figures/variableImportance.png")  
plot(boruta\_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")  
dev.off()

### Logistic Regression

# Build feature list:  
x<-colnames(bankbin)  
x<-x[x != "y"]  
x<-paste(x, collapse='+')  
x # copy this printed value into the model

## [1] "age+job.admin.+job.blue.collar+job.entrepreneur+job.housemaid+job.management+job.retired+job.self.employed+job.services+job.student+job.technician+job.unemployed+marital.divorced+marital.married+marital.single+education.basic.4y+education.basic.6y+education.basic.9y+education.high.school+education.illiterate+education.professional.course+education.university.degree+default.no+default.yes+housing.no+housing.yes+loan.no+loan.yes+contact.cellular+contact.telephone+month.apr+month.aug+month.dec+month.jul+month.jun+month.mar+month.may+month.nov+month.oct+month.sep+day\_of\_week.fri+day\_of\_week.mon+day\_of\_week.thu+day\_of\_week.tue+day\_of\_week.wed+campaign+pdays+previous+poutcome.failure+poutcome.nonexistent+poutcome.success+emp.var.rate+cons.price.idx+cons.conf.idx+euribor3m+nr.employed"

rm(x)  
  
# Everything model:  
full.model <- glm(y ~ age+job+marital+education+default+housing+loan+contact+month+day\_of\_week+campaign+pdays+previous+poutcome+  
 emp.var.rate+cons.price.idx+cons.conf.idx+euribor3m+nr.employed, data = full.train, family = "binomial")  
  
summary(full.model)

##   
## Call:  
## glm(formula = y ~ age + job + marital + education + default +   
## housing + loan + contact + month + day\_of\_week + campaign +   
## pdays + previous + poutcome + emp.var.rate + cons.price.idx +   
## cons.conf.idx + euribor3m + nr.employed, family = "binomial",   
## data = full.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0914 -0.4184 -0.3302 -0.2666 2.9609   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.788e+02 4.094e+01 -6.811 9.69e-12 \*\*\*  
## age -9.766e-04 2.687e-03 -0.363 0.716273   
## jobblue-collar -1.259e-01 8.785e-02 -1.433 0.151768   
## jobentrepreneur -3.910e-02 1.327e-01 -0.295 0.768303   
## jobhousemaid 9.258e-02 1.592e-01 0.581 0.560974   
## jobmanagement 7.418e-03 9.046e-02 0.082 0.934648   
## jobretired 3.512e-01 1.200e-01 2.926 0.003435 \*\*   
## jobself-employed 2.837e-02 1.223e-01 0.232 0.816623   
## jobservices -1.822e-01 9.545e-02 -1.908 0.056332 .   
## jobstudent 3.046e-01 1.262e-01 2.414 0.015789 \*   
## jobtechnician 8.343e-02 7.473e-02 1.116 0.264223   
## jobunemployed 2.575e-02 1.363e-01 0.189 0.850151   
## maritalmarried 9.890e-03 7.290e-02 0.136 0.892080   
## maritalsingle 1.128e-02 8.269e-02 0.136 0.891521   
## educationbasic.6y 2.198e-01 1.369e-01 1.605 0.108412   
## educationbasic.9y -3.016e-02 1.073e-01 -0.281 0.778630   
## educationhigh.school 1.132e-01 1.026e-01 1.103 0.269967   
## educationilliterate 7.569e-01 8.657e-01 0.874 0.381966   
## educationprofessional.course 6.860e-03 1.119e-01 0.061 0.951104   
## educationuniversity.degree 1.432e-01 1.027e-01 1.394 0.163175   
## defaultyes -8.550e+00 1.136e+02 -0.075 0.939997   
## housingyes -6.221e-02 4.421e-02 -1.407 0.159402   
## loanyes 2.032e-02 6.040e-02 0.336 0.736586   
## contacttelephone -8.239e-01 8.197e-02 -10.052 < 2e-16 \*\*\*  
## monthaug 5.045e-01 1.320e-01 3.821 0.000133 \*\*\*  
## monthdec 4.685e-01 2.258e-01 2.075 0.038025 \*   
## monthjul 8.291e-02 1.043e-01 0.795 0.426447   
## monthjun -7.266e-01 1.344e-01 -5.407 6.40e-08 \*\*\*  
## monthmar 1.618e+00 1.567e-01 10.321 < 2e-16 \*\*\*  
## monthmay -3.173e-01 8.822e-02 -3.597 0.000322 \*\*\*  
## monthnov -4.289e-01 1.309e-01 -3.277 0.001051 \*\*   
## monthoct 2.320e-01 1.668e-01 1.391 0.164350   
## monthsep 4.072e-01 1.957e-01 2.081 0.037458 \*   
## day\_of\_weekmon -1.494e-01 7.192e-02 -2.077 0.037813 \*   
## day\_of\_weekthu 1.137e-01 6.991e-02 1.626 0.103913   
## day\_of\_weektue 6.536e-02 7.227e-02 0.904 0.365804   
## day\_of\_weekwed 2.104e-01 7.093e-02 2.966 0.003021 \*\*   
## campaign -4.245e-02 1.169e-02 -3.631 0.000282 \*\*\*  
## pdays -8.724e-04 2.509e-04 -3.477 0.000507 \*\*\*  
## previous -5.005e-02 6.864e-02 -0.729 0.465915   
## poutcomenonexistent 4.768e-01 1.058e-01 4.508 6.54e-06 \*\*\*  
## poutcomesuccess 9.695e-01 2.456e-01 3.947 7.92e-05 \*\*\*  
## emp.var.rate -1.610e+00 1.484e-01 -10.847 < 2e-16 \*\*\*  
## cons.price.idx 2.389e+00 2.681e-01 8.912 < 2e-16 \*\*\*  
## cons.conf.idx 3.321e-02 8.515e-03 3.900 9.62e-05 \*\*\*  
## euribor3m 9.145e-02 1.444e-01 0.633 0.526613   
## nr.employed 1.060e-02 3.371e-03 3.146 0.001656 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 18496 on 24389 degrees of freedom  
## Residual deviance: 14472 on 24343 degrees of freedom  
## AIC: 14566  
##   
## Number of Fisher Scoring iterations: 10

# Smaller model:  
simple.model <- glm(y~job+contact+month+day\_of\_week+campaign+pdays+poutcome+emp.var.rate+cons.price.idx+cons.conf.idx+nr.employed, data=full.train, family="binomial")  
  
summary(simple.model)

##   
## Call:  
## glm(formula = y ~ job + contact + month + day\_of\_week + campaign +   
## pdays + poutcome + emp.var.rate + cons.price.idx + cons.conf.idx +   
## nr.employed, family = "binomial", data = full.train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0543 -0.4176 -0.3308 -0.2661 2.9373   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.914e+02 3.292e+01 -8.853 < 2e-16 \*\*\*  
## jobblue-collar -2.092e-01 7.139e-02 -2.930 0.003392 \*\*   
## jobentrepreneur -7.340e-02 1.310e-01 -0.560 0.575325   
## jobhousemaid 1.342e-02 1.521e-01 0.088 0.929681   
## jobmanagement 6.638e-03 8.822e-02 0.075 0.940022   
## jobretired 2.504e-01 9.269e-02 2.702 0.006902 \*\*   
## jobself-employed 1.512e-02 1.211e-01 0.125 0.900626   
## jobservices -2.050e-01 9.065e-02 -2.261 0.023738 \*   
## jobstudent 2.780e-01 1.180e-01 2.356 0.018473 \*   
## jobtechnician 2.306e-02 6.628e-02 0.348 0.727864   
## jobunemployed -2.732e-02 1.344e-01 -0.203 0.838896   
## contacttelephone -8.184e-01 8.172e-02 -10.015 < 2e-16 \*\*\*  
## monthaug 5.193e-01 1.294e-01 4.014 5.96e-05 \*\*\*  
## monthdec 5.088e-01 2.140e-01 2.378 0.017414 \*   
## monthjul 9.887e-02 1.026e-01 0.964 0.335084   
## monthjun -7.200e-01 1.336e-01 -5.387 7.15e-08 \*\*\*  
## monthmar 1.641e+00 1.469e-01 11.170 < 2e-16 \*\*\*  
## monthmay -3.103e-01 8.640e-02 -3.591 0.000329 \*\*\*  
## monthnov -3.815e-01 1.022e-01 -3.731 0.000191 \*\*\*  
## monthoct 2.943e-01 1.346e-01 2.186 0.028825 \*   
## monthsep 4.646e-01 1.700e-01 2.732 0.006286 \*\*   
## day\_of\_weekmon -1.500e-01 7.178e-02 -2.090 0.036653 \*   
## day\_of\_weekthu 1.118e-01 6.980e-02 1.602 0.109217   
## day\_of\_weektue 6.460e-02 7.206e-02 0.896 0.370004   
## day\_of\_weekwed 2.092e-01 7.084e-02 2.954 0.003138 \*\*   
## campaign -4.265e-02 1.166e-02 -3.658 0.000255 \*\*\*  
## pdays -8.143e-04 2.331e-04 -3.493 0.000478 \*\*\*  
## poutcomenonexistent 5.418e-01 6.893e-02 7.860 3.84e-15 \*\*\*  
## poutcomesuccess 1.017e+00 2.343e-01 4.341 1.42e-05 \*\*\*  
## emp.var.rate -1.593e+00 1.482e-01 -10.754 < 2e-16 \*\*\*  
## cons.price.idx 2.443e+00 2.372e-01 10.302 < 2e-16 \*\*\*  
## cons.conf.idx 3.704e-02 5.969e-03 6.206 5.45e-10 \*\*\*  
## nr.employed 1.214e-02 2.146e-03 5.658 1.53e-08 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 18496 on 24389 degrees of freedom  
## Residual deviance: 14485 on 24357 degrees of freedom  
## AIC: 14551  
##   
## Number of Fisher Scoring iterations: 5

#confidence intervals for simple model  
summary(simple.model$coefficients)

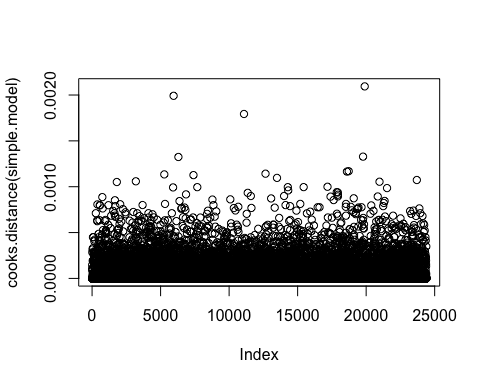
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -291.39304 -0.15000 0.01512 -8.70833 0.27804 2.44302

CI\_lower <- coefficients(simple.model)[2] - 1.96\*summary(simple.model)$coefficients[2,2]  
CI\_upper <- coefficients(simple.model)[2] + 1.96\*summary(simple.model)$coefficients[2,2]  
  
confint(simple.model)

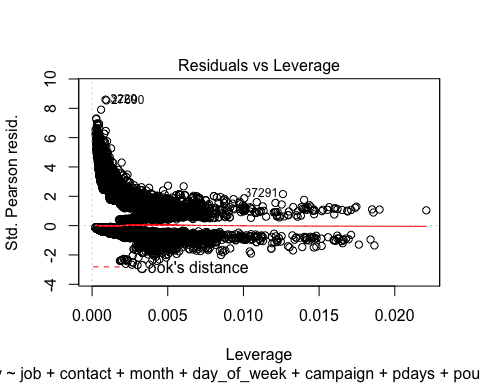
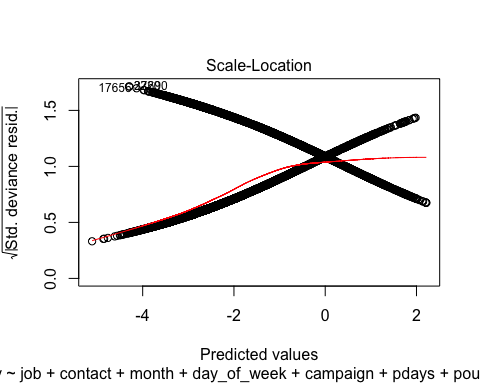
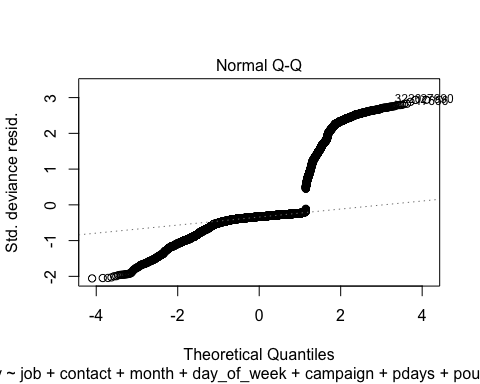
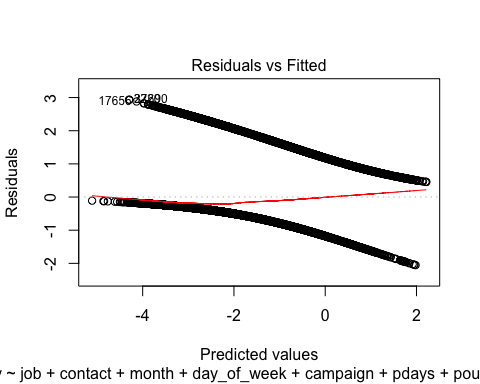
## Waiting for profiling to be done...

## 2.5 % 97.5 %  
## (Intercept) -3.559724e+02 -2.269053e+02  
## jobblue-collar -3.498570e-01 -6.991316e-02  
## jobentrepreneur -3.364049e-01 1.777719e-01  
## jobhousemaid -2.915926e-01 3.052414e-01  
## jobmanagement -1.679445e-01 1.779925e-01  
## jobretired 6.784552e-02 4.312522e-01  
## jobself-employed -2.265947e-01 2.484438e-01  
## jobservices -3.848375e-01 -2.931157e-02  
## jobstudent 4.511668e-02 5.079147e-01  
## jobtechnician -1.072618e-01 1.526135e-01  
## jobunemployed -2.950338e-01 2.321575e-01  
## contacttelephone -9.803456e-01 -6.599246e-01  
## monthaug 2.657572e-01 7.729136e-01  
## monthdec 8.805410e-02 9.278428e-01  
## monthjul -1.022502e-01 2.998912e-01  
## monthjun -9.821161e-01 -4.581550e-01  
## monthmar 1.352700e+00 1.928742e+00  
## monthmay -4.792488e-01 -1.405040e-01  
## monthnov -5.825216e-01 -1.816081e-01  
## monthoct 2.979286e-02 5.576979e-01  
## monthsep 1.308387e-01 7.974710e-01  
## day\_of\_weekmon -2.906939e-01 -9.236710e-03  
## day\_of\_weekthu -2.483954e-02 2.488401e-01  
## day\_of\_weektue -7.659218e-02 2.059596e-01  
## day\_of\_weekwed 7.055308e-02 3.482888e-01  
## campaign -6.613937e-02 -2.044046e-02  
## pdays -1.270531e-03 -3.543067e-04  
## poutcomenonexistent 4.077606e-01 6.780320e-01  
## poutcomesuccess 5.586907e-01 1.479515e+00  
## emp.var.rate -1.883960e+00 -1.303009e+00  
## cons.price.idx 1.978549e+00 2.908427e+00  
## cons.conf.idx 2.535306e-02 4.875696e-02  
## nr.employed 7.938339e-03 1.635443e-02

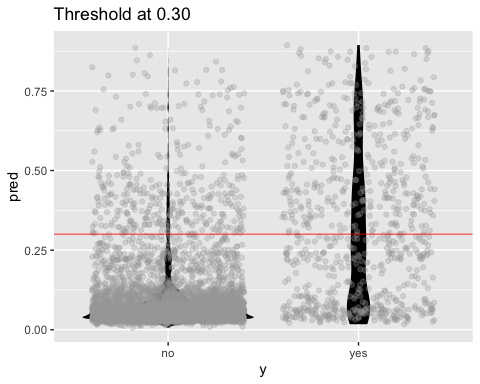
#cooks D and residual plots for simple.model  
plot(cooks.distance(simple.model))



plot(simple.model)



pred\_lm = predict(simple.model, type='response', newdata=full.test)  
  
# plot the prediction distribution  
predictions\_LR <- data.frame(y = full.test$y, pred = NA)  
predictions\_LR$pred <- pred\_lm  
  
plot\_pred\_type\_distribution(predictions\_LR,0.30)



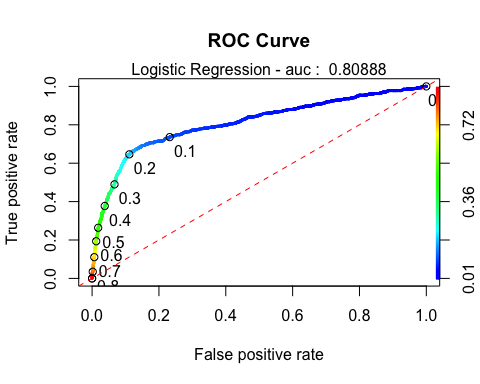
# choose the best threshold as 0.30  
test.eval.LR = binclass\_eval(as.integer(full.test$y)-1, pred\_lm > 0.30)  
  
# Making the Confusion Matrix  
test.eval.LR$cm

## Predicted  
## Actual 0 1  
## 0 4962 356  
## 1 398 382

# calculate accuracy, precision   
acc\_LR=test.eval.LR$accuracy  
prc\_LR=test.eval.LR$precision  
recall\_LR=test.eval.LR$recall  
fscore\_LR=test.eval.LR$fscore  
  
# calculate ROC  
rocr.pred.lr = prediction(predictions = pred\_lm, labels = full.test$y)  
rocr.perf.lr = performance(rocr.pred.lr, measure = "tpr", x.measure = "fpr")  
rocr.auc.lr = as.numeric(performance(rocr.pred.lr, "auc")@y.values)  
  
# print ROC AUC  
rocr.auc.lr

## [1] 0.808875

# plot ROC curve for Logistic Regression  
plot(rocr.perf.lr,  
 lwd = 3, colorize = TRUE,  
 print.cutoffs.at = seq(0, 1, by = 0.1),  
 text.adj = c(-0.2, 1.7),  
 main = 'ROC Curve')  
mtext(paste('Logistic Regression - auc : ', round(rocr.auc.lr, 5)))  
abline(0, 1, col = "red", lty = 2)



# Stepwise Selection commented out to speed up knit process:  
step(full.model,direction="both")  
  
step(simple.model,direction="both")

interaction.model <- glm(y~euribor3m+contact+month+cons.price.idx+emp.var.rate+euribor3m\*nr.employed+pdays+campaign+pdays\*campaign+job+euribor3m\*month+euribor3m\*age+euribor3m\*pdays, data=full.train, family="binomial")  
summary(interaction.model)  
  
  
#confidence intervals for simple model  
summary(interaction.model$coefficients)  
CI\_lower <- coefficients(interaction.model)[2] - 1.96\*summary(interaction.model)$coefficients[2,2]  
CI\_upper <- coefficients(interaction.model)[2] + 1.96\*summary(interaction.model)$coefficients[2,2]  
  
confint(interaction.model)  
  
othermodel<-glm(y~job+marital+education+contact+month+campaign+poutcome+emp.var.rate+cons.conf.idx, traindata,   
 family = binomial(link="logit"))  
summary(othermodel)  
  
(vif(othermodel)[,3])^2  
  
#previous VIF=5  
hoslem.test(othermodel$y, fitted(othermodel), g=10)  
  
# Sig. lack of fit, but large n  
fit1.pred.train <- predict(othermodel, newdata = traindata)   
  
#Create ROC curves  
pred <- prediction(fit1.pred.train, traindata$y)  
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")  
auc.train <- performance(pred, measure = "auc")  
auc.train <- auc.train@y.values  
  
#Plot ROC  
plot(roc.perf)  
abline(a=0, b= 1) #Ref line indicating poor performance  
text(x = .40, y = .6,paste("AUC = ", round(auc.train[[1]],3), sep = ""))  
title(main="Train Set ROC")  
  
#Run model from training set on valid set   
fit1.pred.test <- predict(model.main, newdata = bank\_test)  
  
#ROC curves  
pred1 <- prediction(fit1.pred.test, bank\_test$y)  
roc.perf1 = performance(pred1, measure = "tpr", x.measure = "fpr")  
auc.val1 <- performance(pred1, measure = "auc")  
auc.val1 <- auc.val1@y.values  
plot(roc.perf1)  
abline(a=0, b= 1)  
text(x = .40, y = .6,paste("AUC = ", round(auc.val1[[1]],3), sep = ""))  
title(main="Test Set ROC")

### LDA

LDA.model <- lda(y~., data=scaled.train)

## Warning in lda.default(x, grouping, ...): variables are collinear

pred<-predict(LDA.model, newdata=scaled.test)$class #Predictions can come in many forms, the class form provides the categorical level of your response.  
  
Truth<-scaled.test$y  
  
LDA.confusionMatrix<-table(pred,Truth) # Creating a confusion matrix  
  
LDA.confusionMatrix

## Truth  
## pred 0 1  
## 0 5073 463  
## 1 245 317

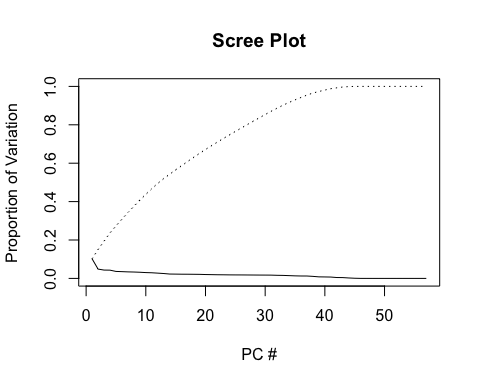
#Missclassification Error  
ME<-(LDA.confusionMatrix[2,1]+LDA.confusionMatrix[1,2])/1000  
ME

## [1] 0.708

#qda.fit <- qda(y ~ age+job+marital+education+default+housing+loan+contact+month+day\_of\_week+campaign+pdays+previous+poutcome+emp.var.rate+cons.price.idx+cons.conf.idx+euribor3m+nr.employed  
 # ,data = bankfull)  
#qda.fit

### PCA

PCA.result<-prcomp(bankbin[,-64],scale.=TRUE)  
PCA.scores<-PCA.result$x  
  
# Add the response column to the PC's data frame  
PCA.scores<-data.frame(PCA.scores)  
PCA.scores$y<-bankbin$y  
  
# Loadings for interpretation  
#PCA.result$rotation  
  
# Scree plot  
PCA.eigen<-(PCA.result$sdev)^2  
PCA.prop<-PCA.eigen/sum(PCA.eigen)  
PCA.cumprop<-cumsum(PCA.prop)  
plot(1:57,PCA.prop,type="l",main="Scree Plot",ylim=c(0,1),xlab="PC #",ylab="Proportion of Variation")  
lines(1:57,PCA.cumprop,lty=3)



# Store PCA Plots in a list  
PCA.plots <- list(  
  
ggplot(data = PCA.scores, aes(x = PC1, y = PC2)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Subscriptions"),  
  
ggplot(data = PCA.scores, aes(x = PC2, y = PC3)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Subscriptions"),  
  
ggplot(data = PCA.scores, aes(x = PC3, y = PC4)) +  
 geom\_point(aes(col=y), size=1)+  
 ggtitle("PCA of Subscriptions")  
)  
  
# Display first three PCA Plots  
png('./figures/PCA.png', width = 1100, height = 250)  
grid.arrange(PCA.plots[[1]], PCA.plots[[2]], PCA.plots[[3]], ncol=3, nrow=1)  
dev.off()

### Recursive Partitioning

# Classification and Regression Trees  
bank.cart<-rpart(y~job+contact+month+day\_of\_week+campaign+pdays+poutcome+emp.var.rate+cons.price.idx+cons.conf.idx+nr.employed, full.train , method = 'class')  
  
#par(mfrow=c(1,1))  
#fancyRpartPlot(bank.cart , digits=2 , palettes = c("Purples", "Oranges"))  
  
#predict  
cart\_pred <- predict( bank.cart , full.train , type = "class")  
cart\_prob <- predict( bank.cart , full.train , type = "prob")  
  
# Confusion matrix  
confusionMatrix(cart\_pred , full.train$y)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 21099 2524  
## yes 212 555  
##   
## Accuracy : 0.8878   
## 95% CI : (0.8838, 0.8918)  
## No Information Rate : 0.8738   
## P-Value [Acc > NIR] : 9.663e-12   
##   
## Kappa : 0.2509   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9901   
## Specificity : 0.1803   
## Pos Pred Value : 0.8932   
## Neg Pred Value : 0.7236   
## Prevalence : 0.8738   
## Detection Rate : 0.8651   
## Detection Prevalence : 0.9686   
## Balanced Accuracy : 0.5852   
##   
## 'Positive' Class : no   
##

### Cross table validation for CART  
CrossTable(full.train$y, cart\_pred,  
 prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,  
 dnn = c('actual default', 'predicted default'))

## Cell Contents   
## |-------------------------|  
## | N |   
## | N / Table Total |   
## |-------------------------|  
##   
## =======================================  
## predicted default  
## actual default no yes Total  
## ---------------------------------------  
## no 21099 212 21311  
## 0.865 0.009   
## ---------------------------------------  
## yes 2524 555 3079  
## 0.103 0.023   
## ---------------------------------------  
## Total 23623 767 24390  
## =======================================

### KNN

KNN Has an 88% overall accuracy, with 97% sensitivity. However, specificity suffers with only 29.6% accuracy. Scaling the data alone did not improve the performance of KNN. We then fed KNN algorithm the limited set of predictors used in Logistic Regression, which also did not improve performance.

# Implementing KNN  
###########################################  
  
# Model the same sample we fed to logistic regression (KNN is sensitive to noise)  
  
KNN.model <- train(y~job+contact+month+day\_of\_week+campaign+pdays+poutcome+emp.var.rate+cons.price.idx+cons.conf.idx+nr.employed,   
 data = full.train, method = "knn",   
 maximize = TRUE,  
 trControl = trainControl(method = "cv", number = 10),  
 preProcess=c("center", "scale")  
 )  
  
KNN.preds <- predict(KNN.model , newdata = full.test)  
confusionMatrix(KNN.preds , full.test$y)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 5178 546  
## yes 140 234  
##   
## Accuracy : 0.8875   
## 95% CI : (0.8793, 0.8953)  
## No Information Rate : 0.8721   
## P-Value [Acc > NIR] : 0.0001339   
##   
## Kappa : 0.3518   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9737   
## Specificity : 0.3000   
## Pos Pred Value : 0.9046   
## Neg Pred Value : 0.6257   
## Prevalence : 0.8721   
## Detection Rate : 0.8491   
## Detection Prevalence : 0.9387   
## Balanced Accuracy : 0.6368   
##   
## 'Positive' Class : no   
##

### Cross table validation for KNN  
CrossTable(full.test$y, KNN.preds,  
 prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,  
 dnn = c('actual default', 'predicted default'))

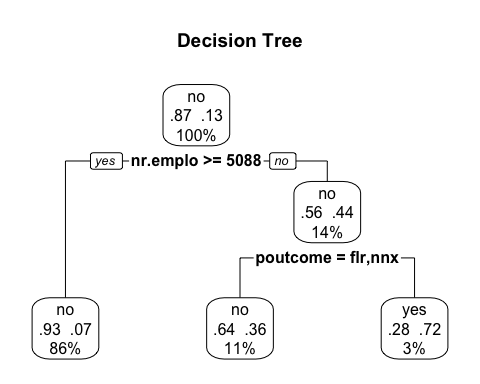
## Cell Contents   
## |-------------------------|  
## | N |   
## | N / Table Total |   
## |-------------------------|  
##   
## =======================================  
## predicted default  
## actual default no yes Total  
## ---------------------------------------  
## no 5178 140 5318  
## 0.849 0.023   
## ---------------------------------------  
## yes 546 234 780  
## 0.090 0.038   
## ---------------------------------------  
## Total 5724 374 6098  
## =======================================

# Hypertuning identified 60 as the best K  
classifications = knn(scaled.train,scaled.test,as.factor(scaled.train$y), prob = TRUE, k = 60)  
confusionMatrix(table(scaled.test$y,classifications, dnn = c("Prediction", "Reference")), positive = '1')

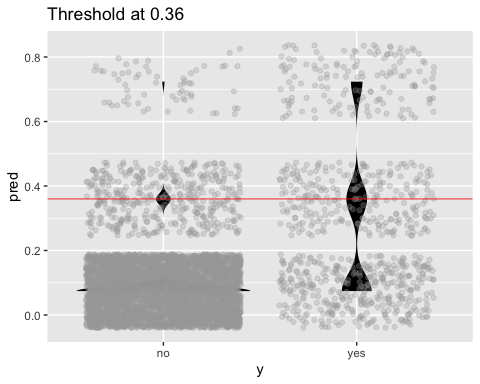
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 5239 79  
## 1 598 182  
##   
## Accuracy : 0.889   
## 95% CI : (0.8808, 0.8968)  
## No Information Rate : 0.9572   
## P-Value [Acc > NIR] : 1   
##   
## Kappa : 0.3051   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.69732   
## Specificity : 0.89755   
## Pos Pred Value : 0.23333   
## Neg Pred Value : 0.98514   
## Prevalence : 0.04280   
## Detection Rate : 0.02985   
## Detection Prevalence : 0.12791   
## Balanced Accuracy : 0.79743   
##   
## 'Positive' Class : 1   
##

iterations = 1  
set.seed(115)  
numks = round(sqrt(dim(scaledbin)[1])\*1.2)  
masterAcc = matrix(nrow = iterations, ncol = numks)  
masterSpec = matrix(nrow = iterations, ncol = numks)  
masterSen = matrix(nrow = iterations, ncol = numks)  
knnArray <- c(  
 "job.admin.",  
 "job.blue.collar",  
 "job.entrepreneur",  
 "job.housemaid",  
 "job.management",  
 "job.retired",  
 "job.self.employed",  
 "job.services",  
 "job.student",  
 "job.technician",  
 "job.unemployed",  
 "contact.cellular",  
 "contact.telephone",  
 "month.apr",  
 "month.aug",  
 "month.dec",  
 "month.jul",  
 "month.jun",  
 "month.mar",  
 "month.may",  
 "month.nov",  
 "month.oct",  
 "month.sep",  
 "day\_of\_week.fri",  
 "day\_of\_week.mon",  
 "day\_of\_week.thu",  
 "day\_of\_week.tue",  
 "day\_of\_week.wed",  
 "campaign",  
 "pdays",  
 "poutcome.failure",  
 "poutcome.nonexistent",  
 "poutcome.success",  
 "emp.var.rate",  
 "cons.price.idx",  
 "cons.conf.idx",  
 "nr.employed"  
)  
  
for(j in 1:iterations) {  
 # resample data  
 KNN.trainIndices = sample(1:dim(scaledbin)[1],round(.8 \* dim(scaledbin)[1]))  
 KNN.train = scaledbin[trainIndices,]  
 KNN.test = scaledbin[-trainIndices,]  
 for(i in 1:numks) {  
 # predict using i-th value of k  
 classifications = knn(KNN.train[,knnArray],KNN.test[,knnArray],as.factor(KNN.train$y), prob = TRUE, k = i)  
 CM = confusionMatrix(table(as.factor(KNN.test$y),classifications, dnn = c("Prediction", "Reference")), positive = '1')  
 masterAcc[j,i] = CM$overall[1]  
 masterSen[j,i] = CM$byClass[1]  
 masterSpec[j,i] = ifelse(is.na(CM$byClass[2]),0,CM$byClass[2])  
 print(i)  
 }  
}  
  
MeanAcc <- colMeans(masterAcc)  
MeanSen <- colMeans(masterSen)  
MeanSpec <- colMeans(masterSpec)  
png('./figures/bestK.png')  
plot(seq(1,numks), MeanAcc, main="K value determination", xlab="Value of K")  
dev.off()  
k <- which.max(MeanAcc)  
specs <- c(MeanAcc[k],MeanSen[k],MeanSpec[k])  
names(specs) <- c("Avg Accuracy", "Avg Sensitivity", "Avg Specificity")  
specs %>% kable("html") %>% kable\_styling  
  
classifications = knn(scaled.train[,knnArray],scaled.test[,knnArray],as.factor(scaled.train$y), prob = TRUE, k = k)  
confusionMatrix(table(scaled.test$y,classifications, dnn = c("Prediction", "Reference")), positive = '1')

# fit the decision tree classification #p1,2,3 is in logistic  
classifier = rpart(formula = y~job+contact+month+day\_of\_week+campaign+pdays+poutcome+emp.var.rate+cons.price.idx+cons.conf.idx+nr.employed,  
 data = full.train, method = "class")  
  
# plot  
prp(classifier, type = 2, extra = 104, fallen.leaves = TRUE, main="Decision Tree")



# predict test data by probability  
pred.DT = predict(classifier, newdata = full.test, type = 'prob')  
  
# find the threshold for prediction optimization  
predictions\_DT <- data.frame(y = full.test$y, pred = NA)  
predictions\_DT$pred <- pred.DT[,2]  
plot\_pred\_type\_distribution(predictions\_DT,0.36)



# choose the best threshold as 0.36  
test.eval.DT = binclass\_eval(as.integer(full.test$y)-1, pred.DT[,2] > 0.36)  
  
# Making the Confusion Matrix  
test.eval.DT$cm

## Predicted  
## Actual 0 1  
## 0 5256 62  
## 1 635 145

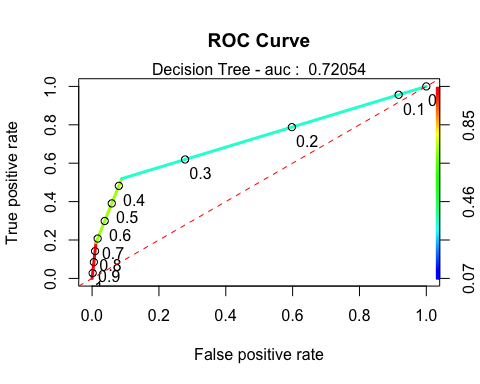
# print evaluation  
cat("Accuracy: ", test.eval.DT$accuracy,  
 "\nPrecision: ", test.eval.DT$precision,  
 "\nRecall: ", test.eval.DT$recall,  
 "\nFScore: ", test.eval.DT$fscore  
 )

## Accuracy: 0.8857002   
## Precision: 0.7004831   
## Recall: 0.1858974   
## FScore: 0.2938197

# calculate ROC curve  
rocr.pred = prediction(predictions = pred.DT[,2], labels = full.test$y)  
rocr.perf = performance(rocr.pred, measure = "tpr", x.measure = "fpr")  
rocr.auc = as.numeric(performance(rocr.pred, "auc")@y.values)  
  
# print ROC AUC  
rocr.auc

## [1] 0.7205373

# plot ROC curve  
plot(rocr.perf,  
 lwd = 3, colorize = TRUE,  
 print.cutoffs.at = seq(0, 1, by = 0.1),  
 text.adj = c(-0.2, 1.7),  
 main = 'ROC Curve')  
mtext(paste('Decision Tree - auc : ', round(rocr.auc, 5)))  
abline(0, 1, col = "red", lty = 2)



## EDA Plots

### Correlation Plot

##### 

### Scatterplot Matrix

##### 

### Generalized Pairs Plot

##### 