DATA MINING PROJECT
LUKE KILLORAN
(13434418)

Abstract: Seven classifiers were implemented on data of bank clients to compare which could best predict which clients would subscribe to a term deposit account. The most predictive algorithm was the Adaboost one. Although it correctly classified many clients, only around a quarter of subscribers were identified under its predictions. The hope is that if the bank collects much more data on its client the model may dramatically improve the rate at which it identifies potential susbcribers. On the other hand, it was quite accurate when it did predict that a client would subscribe (correct 3 out of every 4 occasions). This means the bank will be able to funnel its advertising more efficiently by targeting customers more likely to be swayed. Moreover, the analysis in this study has presented charts and figures both in its core and in the Appendix that highlight the most important and least important factors in subscribership. The former factors could yield more focus in the future, or can determine better advertising strategies, and the latter factors should garner less time and focus to improve productivity.

Contents

Introduction

The dataset was bank information on 4,521 clients, including 14 features (8 categorical, 6 numerical) and 1 response variable – whether the customer has subscribed to a term deposit account. 88.5% of clients did not subscribe. It seemed like the focus is devising a model that could accurately predict whether a customer will subscribe, rather than not subscribe, since this information is much more useful to the bank than the latter. Thus, this was examined with greater emphasis. Although, it was unclear as to what greater weighting should be afforded to predicting subscribership. If this study proved to be successful then this model could infer which clients are worth pursuing further with advertisements and/or promotions, and which are not. This could drastically shrink churn with those who benefit from ads, and save funds on those who don't. The model may even infer the relative effect of various variables on the response. This could provide valuable advice about better strategies for targeting a certain client. It could also indicate the general weakness or strength of certain strategies, which could inform management whether to scrap that program or bolster its funding of it., through labour or technology. The model could also be used to gauge the performance of current management in growing or maintaining market share, which could determine whether the board of directors will look elsewhere for company guidance. Overall, this type of analysis is crucial for all levels of management, and is a requirement in today's technologically-charged economy. For if this analysis is not completed, the company must expect to be undercut by competitors who will be sure to have conducted this analysis.

First, explanatory analysis was required. This provides an introduction to the relationships between the variables and the response. This introductory analysis (including its discussion and figures) is quite valuable , to improve the interpretation of the final model but also in its own right, as a general guide to the business – what variables are important and how does knowing the value of that variable impact the likelihood of subscribership.

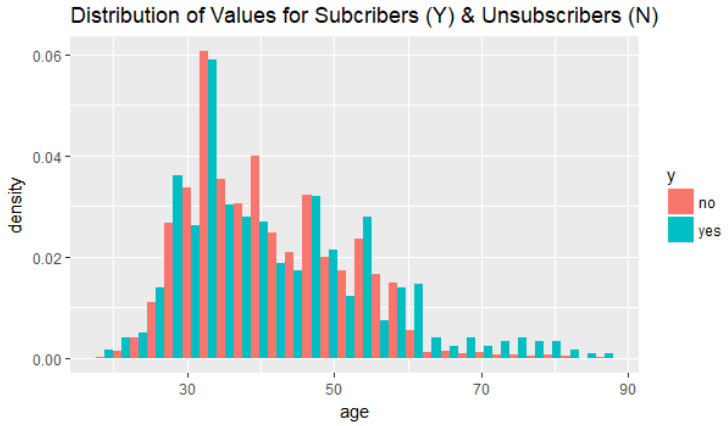
Counts of the response variable across the factor levels of each of the categorical variables were obtained. "no" counts were adjusted by dividing by total number of clients that had not subscribed and the same was done for the "yes" counts. This was to control for the preponderance of unsubscribed clients; and its conceptual interpretation is that, as a result, each count or density figure represents a count assuming the same number of total subscribers as unsubscribers. Following this, the proportion of each resultant density that is spread over unsubscribed clients (and subsequently over subscribed clients) for each factor level of the 8 categorical variables was calculated. Finally, the "no" proportions were subtracted from the "yes" proportions to obtain a kind

of likelihood percentage, which I graphed for all levels of the categorical features. The greater the percentage, the greater the dominance of subscribed clients over unsubscribed within that factor level. The opposite is true for lower (negative) percentages.

The inference is that the greater the percentage in absolute terms, the more likely that factor is to be a strong predictor of subscribership. Therefore we would expect the variables with factor levels boasting large bars (positive or negative) to be the most important. For example, provided in the figures below are the variables with the largest bars; poutcome (86 percentage points), month (one bar over 74 pp, 4 bars over 55 pp), and job (one bar over 40 pp, 4 bars over 20 pp). These figures can be explained like so: the differential percentage is +86% when the previous advertising campaign was successful (poutcome=success), and therefore, this means that in these cases, it is much more probable that clients have subscribed to a term deposit account. This makes sense as usually the marker for success for advertisers is securing a subscription, so it is unsurprising that there is a significant correlation here. Similar inferences can be drawn for other factors. The rest of the figures are displayed in the Appendix.

Each variable was graphed using a histograms with bars for subscribers and unsubscribers side-by-side. Density was used rather than frequency as there were more “no” counts than “yes” counts so frequencies were normalised by dividing by total frequency – thus forming a comparative measure between subscribers and unsubscribers. From this, we can draw inferences such as that clients between the ages of 30 and 50 are more likely to be unsubscribed as can be seen in the figure below (since red bars are higher), yet clients in other age ranges are more likely to be subscribers. This is consistent with the graphs before indicating that students and retired people are much more likely

to be subscribers than not). The other figures are exhibited in the Appendix. Note some are adjusted to exclude the values near 0 due to the sheer volume of these values.



The Pearson correlation matrix for all numeric features was found (and is displayed in the figure below). Fortunately the variables are not that correlated with each other (so each adds new information) because the second greatest absolute correlation has a magnitude of 0.16. Previous and pdays are positively correlated to a degree of 0.59. The interpretation of this is that the greater the number of contacts with this client before this campaign, the greater the tendency to leave the client uncontacted for a longer period of time. This is understandable given that the client has been contacted enough so that either they have accepted the offers or they have declined. There is more to be gained for the advertiser by contacting relatively new and uncontacted clients. Although, for the rest of the numerical variables, there seems to be near independence. This means that examining each variable individually is not a futile task. As such, a t-test was conducted for each numeric feature to measure whether the mean of its values differs among the population of subscribers and the population of unsubscribers. Age, pdays, campaign, previous all exhibited statistically significantly different values (as measured by their means) between "yes" and "no" clients, at the 5% level of significance (and their p-values are detailed below with significant ones marked in green). On the other hand, balance and day values were not substantially different among subscribers and unsubscribers. Thus, the former variables are probably most important variables in the prediction process, however different means does not ensure linear separability of subscribers and unsubscribers along the values in the one-dimensional scale of numerical values for a feature. Note that none of this analysis was replicated for the categorical variables, especially the correlation investigation. Therefore some of the categorical variables could be adding more noise to the final model than predictive power, if they are included with other factors which already capture their effects.

T-Test P-Values	
age	0.01259757
balance	0.1517785
day	0.4493466
campaign	1.81E-08
pdays	7.03E-09
previous	8.46E-11

Pearson Correlation Matrix						
	age	balance	Day	campaign	pdays	previous
age	1	0.08	-0.02	-0.01	-0.01	0
balance	0.08	1	-0.01	-0.01	0.01	0.03
day	-0.02	-0.01	1	0.16	-0.09	-0.06
campaign	-0.01	-0.01	0.16	1	-0.09	-0.07
pdays	-0.01	0.01	-0.09	-0.09	1	0.58
previous	0	0.03	-0.06	-0.07	0.58	1

Methods

The 6 numerical variables were normalised by subtracting every value by that variable's minimum and dividing the result by the difference between that's variable's maximum and minimum values.

This was done to ensure that the numerical values have the same scale, and do not dominate the effect of one another.

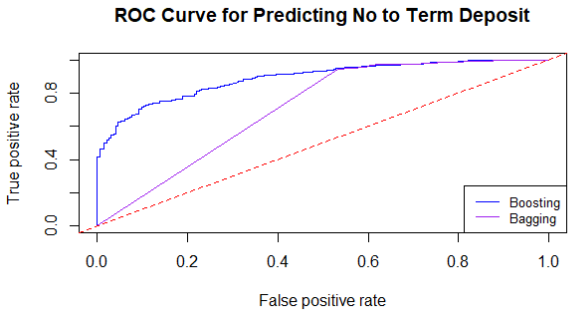
The dataset was split randomly into a training and validation set (2/3 of observations) and a test set (1/3 observations). Then within the training and validation set, 10 fold cross validation was carried out so as to produce predictions for every observation. 7 Classification models were implemented in this process. These included a decision tree, logistic regression, bagging (with trees), boosting (with trees), random forests (with trees), random selection, and ZeroR. (Random Selection and ZeroR were included for comparison sake.) They were each compared using overall measures of performance, and bilateral metrics to indicate relative performance in comparison with each peer. A final model was selected on the basis of this evidence. This model was used to predict the values on the test set. The most important variables according to the top 2 models were delineated. Following this, another model was used to evaluate the strength of effect of each variable.

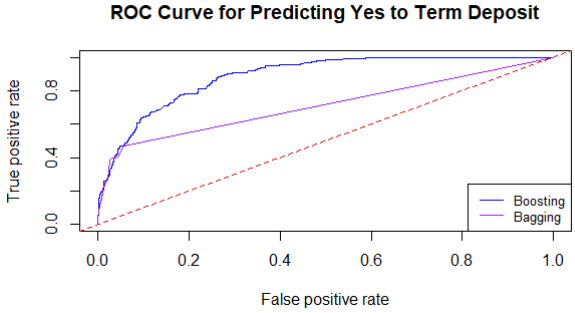
Results

Displayed below is a medley of metrics for each model on the validation set (in 10-fold CV), Mc Nemar's test statistic for each possible couple of models, some ROC plots, a table and two charts detailing variable importance, and test results for the chosen model.

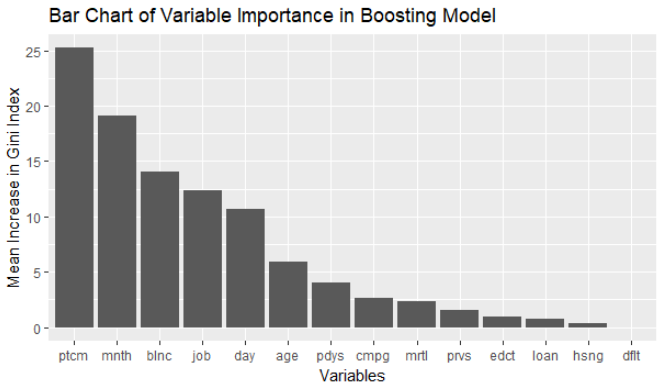
Results	Tree	Log	Bag	Boost	Forest	Random	ZeroR	Best in Category	Margin of Victory
CA	89.48%	89.72%	90.45%	91.04%	89.58%	85.24%	89.18%	Boost	1.63%
No_Recall	98.88%	98.92%	99.00%	98.81%	98.74%	94.75%	100.00%	ZeroR	1.13%
No_Precision	90.26%	90.44%	91.07%	91.78%	90.46%	89.34%	89.18%	Boost	1.47%
No_F1	94.37%	94.49%	94.87%	95.16%	94.42%	91.97%	94.28%	Boost	0.79%
Yes_Recall	11.96%	13.80%	19.94%	26.99%	14.11%	6.75%	0.00%	Boost	95.55%
Yes_Precision	56.52%	60.81%	70.65%	73.33%	57.50%	13.50%	0.00%	Boost	27.54%
Yes_F1	19.75%	22.50%	31.10%	39.46%	22.66%	9.00%	0.00%	Boost	75.39%

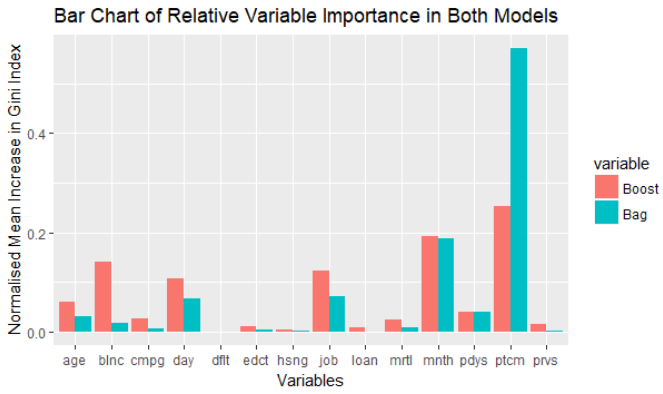
Mc Nemar's Test Values (Critical Value = 3.84)			
Boost & ZeroR :	112.550	Log & ZeroR :	80.520
Boost & Random :	25.208	Log & Random :	3.041
Boost & Tree :	20.951	Log & Tree :	0.766
Boost & RF :	20.098	Log & RF :	0.214
Boost & Log :	17.284	RF & ZeroR :	72.532
Boost & Bag:	3.524	RF & Random :	1.513
Bag & ZeroR :	102.684	RF & Tree :	0.093
Bag & Tree :	16.681	Tree & ZeroR :	72.653
Bag & Random :	14.880	Tree & Random :	0.928
Bag & RF :	13.587	Random & ZeroR :	85.423
Bag & Log:	11.025		





Log Model Var. Imp & Effect	Coefficients	Effect on Subscribership	P-Values
poutcomesuccess	2.3632253	Positive	2.95E-14
maritalmarried	-0.7582138	Negative	4.49E-05
loanyes	-0.7559627	Negative	0.001119
monthoct	1.2566579	Positive	0.001495
monthmar	1.1491376	Positive	0.011114
campaign	-3.7970324	Negative	0.015159
monthmay	-0.6131984	Negative	0.025466
jobretired	0.6700377	Positive	0.046731





Test Set Results	CA	No_Recall	No_Precision	No_F1	Yes_Recall	Yes_Precision	Yes_F1
Boost	88.852%	98.780%	89.503%	93.913%	22.051%	72.881%	33.858%

Discussion

The boosting model obtained the best scores in almost all of the performance metrics, with the bagging model a close second. The total classification accuracy metric was not really of prime interest due to both the preponderance of unsubscribers and the goal of predicting a "yes" client (also known as a subscriber). Rather, the two most important measures of success were recall for subscribers and precision for subscribers since these examine the capability of the model to find all the potential subscribers and to be accurate when predicting a client is subscriber, respectively. On both these metrics the boosting model attained a significant margin of victory in comparison to its closest peer. Yet, the recall mustered by the boosting model was quite underwhelming – nearly ¾ of subscribers were not identified as so.

The Mc Nemar's test compared each pair of models with each other using the statistic which is given by $\frac{(|n_{01}-n_{10}|-1)^2}{n_{01}+n_{10}}$, where n_{01} is the number of predictions classified incorrectly by model 1 but by model 2 and n_{10} is the opposite. The critical value was 3.84 which is the chi-square value for a 5% level of significance (and 1 degree of freedom). The test was to measure whether the classification accuracies of the models were statistically significantly different from each other given the same set of observations. The boosting model and bagging model were statistically significantly more accurate than all other models at the 5% level of significance. However, the boosting model was not statistically significantly than its adversary, the bagging model.

Although, the ROC plots displayed the boosting model's dominance over the bagging model. The vital chart was the ROC plot for the "yes" clients. The goal was to achieve a curve that got close to the top left of the graph. Here points would signify a threshold on the model for finding most subscribers *and* not misclassifying many unsubscribers as subscribers. Since the boosting model is much closer to this area, it can be considered the more preferable model.

Moreover, due to the superior performance of the boosting model in nearly all of the tests thusfar, it was chosen as the final model. It uses the Adaboost algorithm whereby a bootstrapped sample of the dataset is taken (which is a sample with replacement). The probability of an observation being sampled is 1/(number of observations in dataset), which is the initial *weight* of each observation. Then the model is run and the weights of the misclassified examples are adjusted by multiplying by $e^{\alpha} = \left(\frac{1-error}{error}\right)$, where error is the weighted sum of misclassified observations. The weights of other examples are found by normalising all the weights to sum to 1. Then another sample is drawn and another model is fit. Eventually this process was carried out 100 times but only 25 models (decision trees) were retained. Thus, predictions using this algorithm featured an ensemble of these 25 models.

It is unsurprising that the boosting algorithm was the best performing one. It allows us to improve accuracy by focusing on misclassified samples by giving them greater weight. This is what gives it the edge over bagging and random forests. Note that decision trees are so unstable that they can still provide diverse votes despite the samples becoming more concentrated of the same observations. The combination of weak learners also allows for better accuracy in predictions among the ensemble due to a diverse yet accurate set of votes. This is what gives it the edge over the logistic regression and decision tree models. However, its performance on the test set (detailed above) is disappointing due to its low recall for subscribers. In all other metrics, it performs adequately.

Now that the final model has been decided, it might be worth to consider which variables provide the greatest impact on predictions. The rankings listed poutcome, month, balance, job, day, age as the most pivotal features respectively. This provides confirmation of the earlier explanatory analysis on categorical variables (due to the appearance of highlighted variables poutcome, month and job), but is evidence against our method for assessing predictive power amongst the numerical variables (due to the appearance of balance and day which scored low t-test statistics for difference of two means between their subscriber and unsubscriber populations). Obviously this is understandable. – the t-test had clear limitations, which were delineated before, and these variables are not generally the ones with the most predictive power. They are the ones with the most predictive power for *this* model. Interestingly, though, the bagging algorithm returned the same set of variables with the top rankings. These rankings were determined by calculating, for each feature, the increase in the Gini Index when that feature was included as opposed to being excluded (with all other variables included on both occasions). The Gini Index for a node in a decision tree is given by $1 - \sum_{i=0}^{1} p_i^2$ where p_0 is the proportion of observations in that node that are unsubscribers and p_1 is the same but for subscribers. The total Gini Index is calculated by using a weighted sum across all leaves of the weak classifiers.

Moreover, a logistic regression was conducted over the training and validation set. This was completed because the logistic regression model is an easily interpretable model unlike the boosting algorithm. Thus, it should give us insight directly into the predictive power and effect of various factor levels rather than whole variables. As such, provided above is the only factor levels / numeric variables that were significant at the 5% level of significance. Some of these factor levels were noticed before, such as when poutcome (previous advertising campaign) was successful. Clearly this is a great predictor (hence its significant p-value), and its effect is positive so that when the binary variable poutcomesuccessful is true (i.e. poutcome = success), the likelihood of the client subscribing is greater than when poutcomesuccessful is not true. The other variables can be interpreted similarly.

Conclusions

The Adaboost algorithm is clearly the best performing model for predicting subscribership. Although, it only classifies ¼ of subscribers as such. This is a major flaw. However, the hope is that if the bank collects much more data on its client the model may dramatically improve its predictions. On the other hand, since the model is relatively accurate when it predicts that client would subscribe (correct on close to 3 out of every 4 predictions), this means the bank will be able to funnel its advertising more efficiently by targeting customers more likely to be swayed. Moreover, this analysis has presented charts and figures both in its core and in the Appendix that highlight the most important and least important factors in subscribership. The former factors could yield more focus in the future, or can determine better advertising strategies, and the latter factors should garner less time and focus to improve productivity.

References

Notes provided in UCD module "Data Mining" by Dr. Niamh Cahill, as well as the following R packages: reshape2, ggplot2, plotly, rpart, partykit, nnet, adabag, randomForest, mvtnorm, ROCR, pROC.

Appendix

Figures

Figure 1

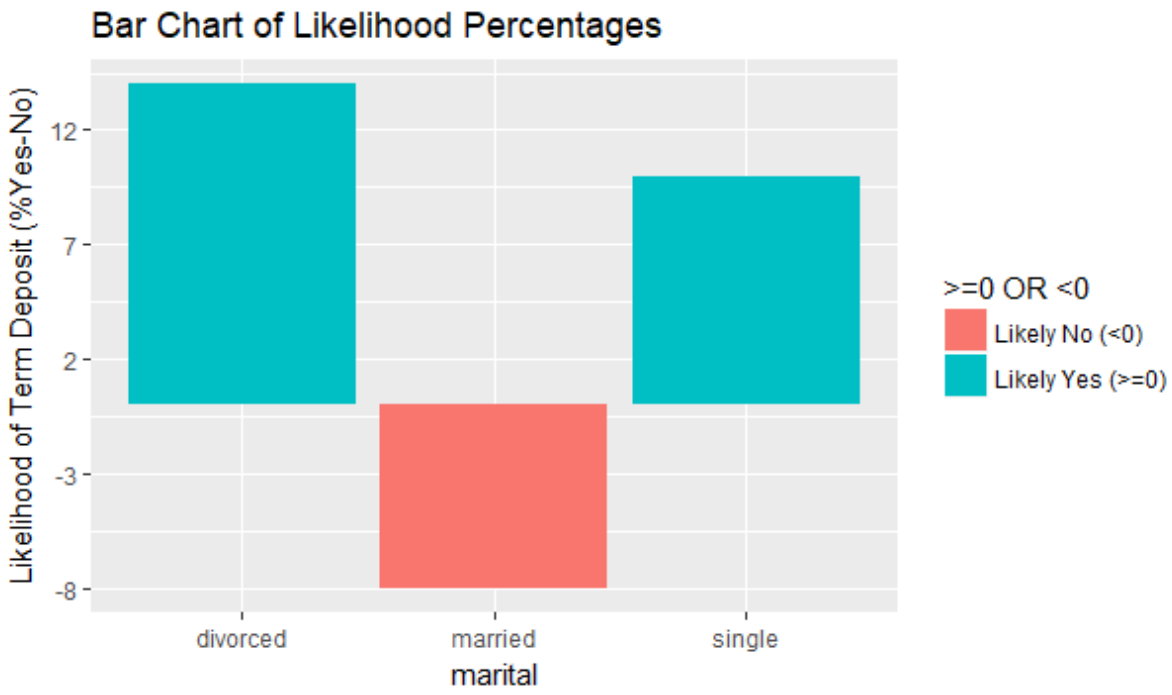


Figure 2

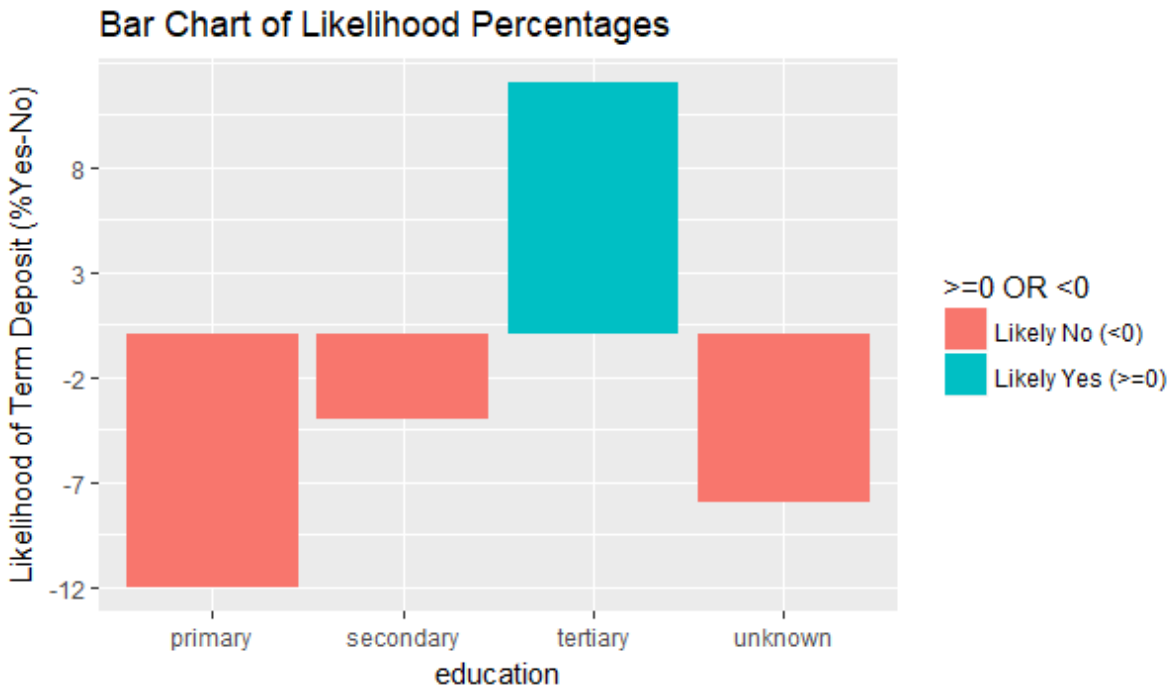


Figure 3 (bar is 1 percentage point high)

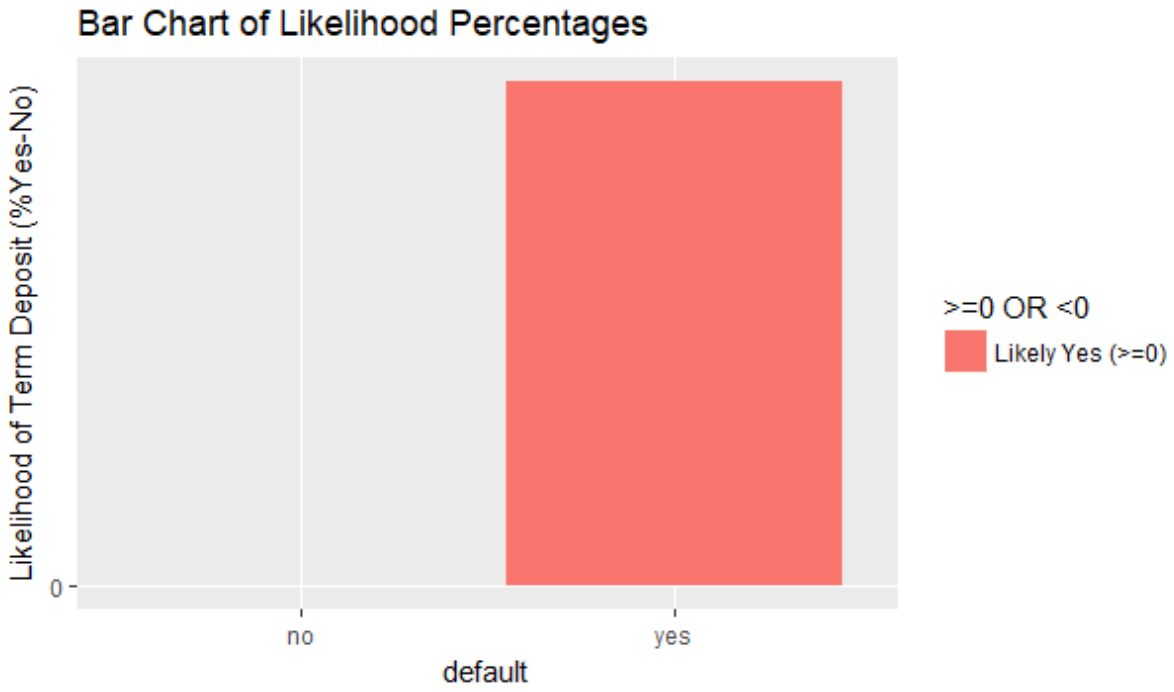


Figure 4

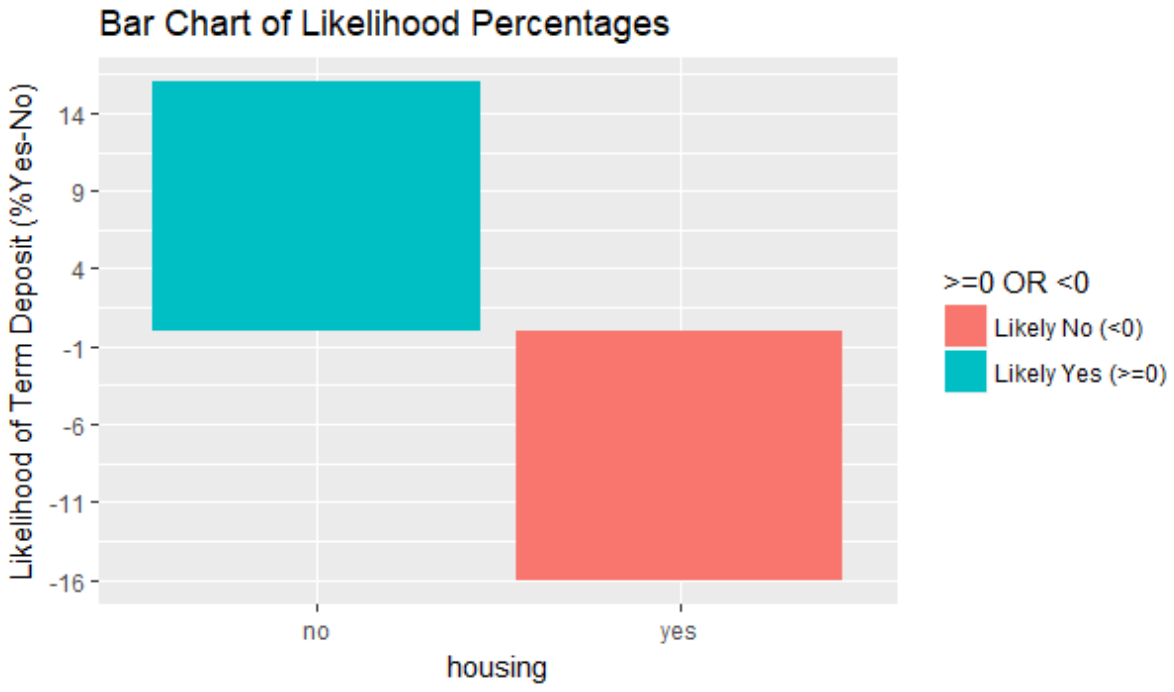


Figure 5

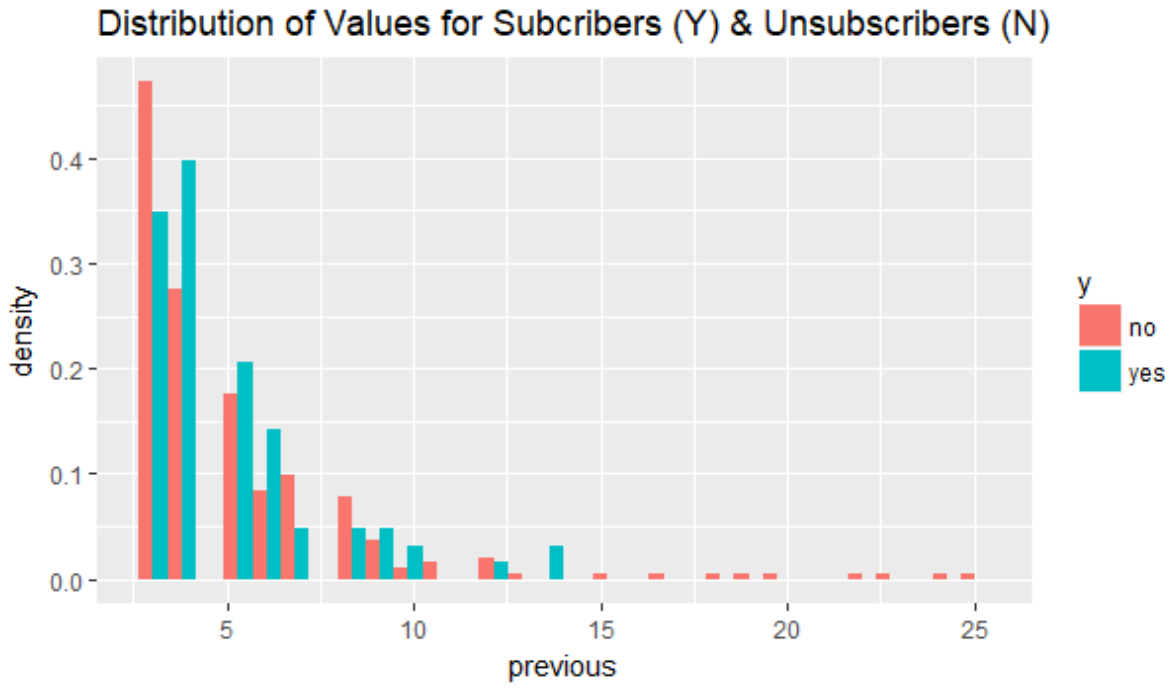


Figure 6

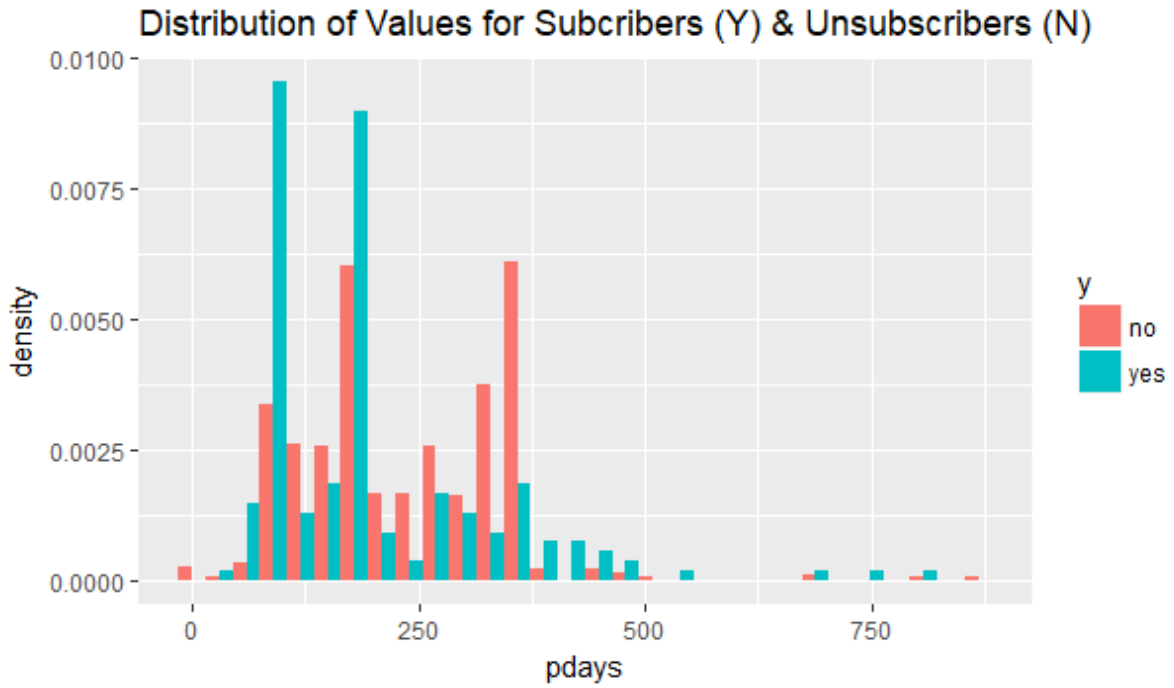


Figure 7

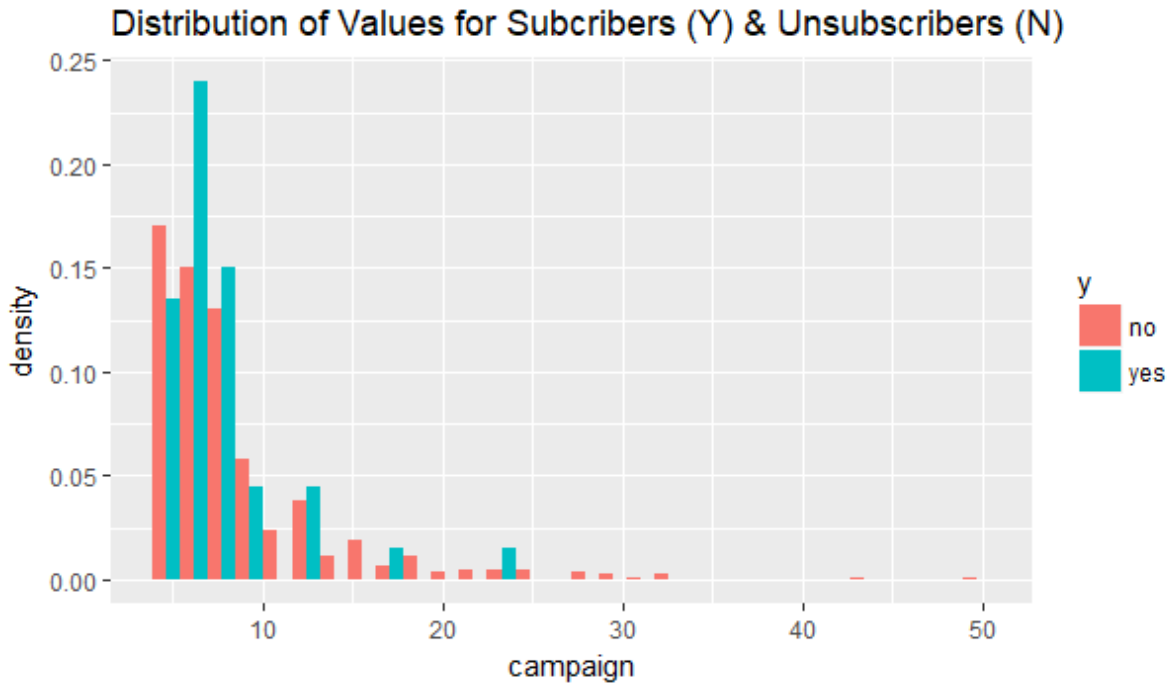


Figure 8

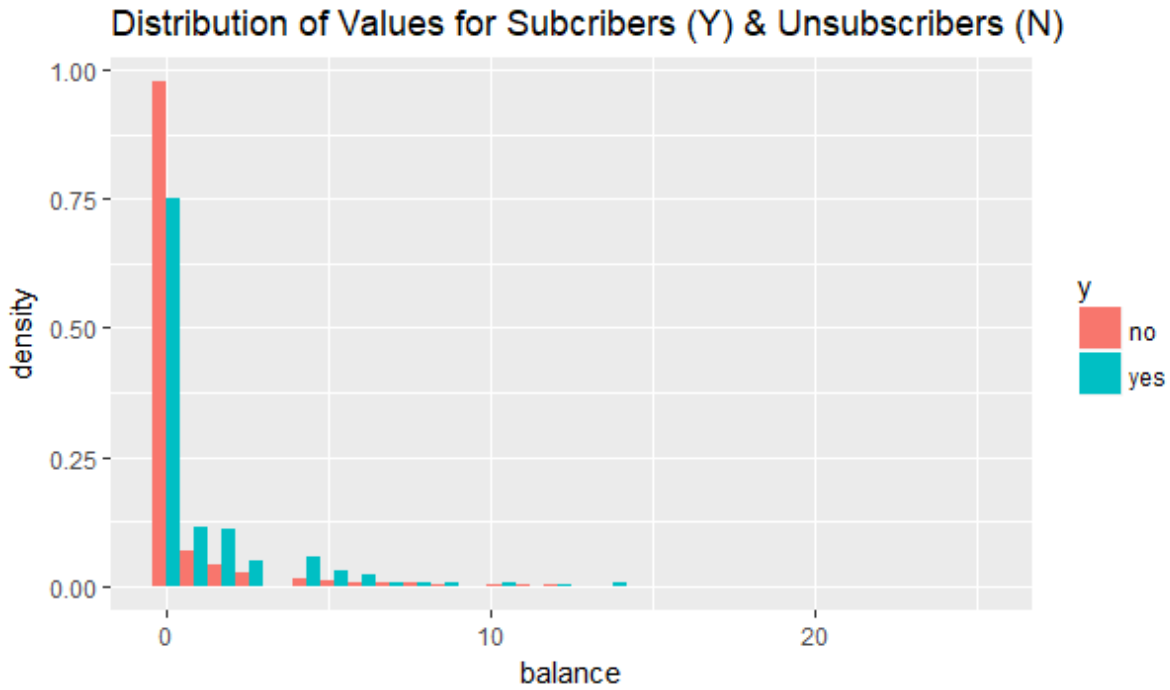
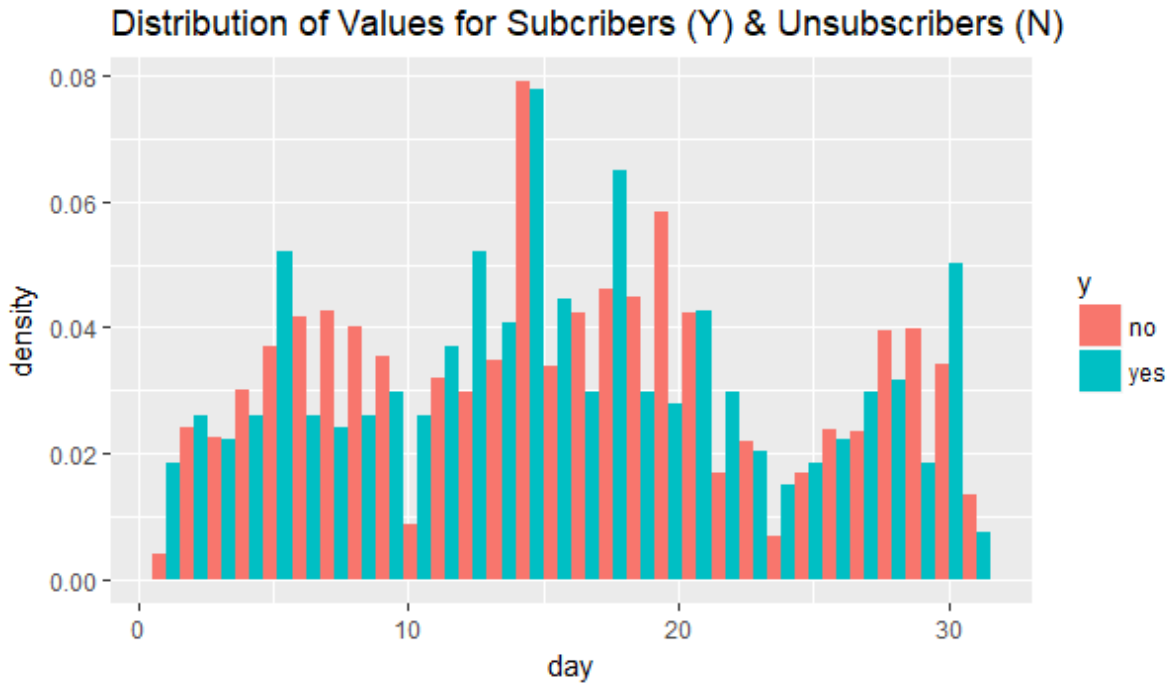


Figure 9



R-Code

```
library(reshape2)
library(ggplot2)
library(plotly)
library(rpart)
library(partykit)
library(nnet)
library(adabag)
library(randomForest)
library(mvtnorm)
library(ROCR)
library(pROC)

bank <- read.csv(file="C:/Users/Luke/Documents/UCD/Data Mining/Labs & Assignments/bank.csv",
header=TRUE, sep=",")
head(bank)
nrow(bank)
table(bank[,15])*100/sum(table(bank[,15]))

# check blanks
length(bank[,1])
for (i in 1:15){
  cat(i,":",length(which(is.na(bank[,i])==1)),"\n")
}
```

```
# Categorical Variables
cat_vars = c(2,3,4,5,7,8,10,14)
for (i in 1:8){
  bank[,cat_vars[i]] = as.factor(bank[,cat_vars[i]])
}

for (i in 1:8){
  bank_col = bank[,cat_vars[i]]
  full_table = table(bank_col,bank$y)
  row_no = full_table[1:length(table(bank_col))]
  row_yes = full_table[(length(table(bank_col))+1):(2*length(table(bank_col)))]
  full_table[1:length(table(bank_col))] = row_no/sum(row_no)
  full_table[(length(table(bank_col))+1):(2*length(table(bank_col)))] = row_yes/sum(row_yes)
  tab <-
round((100*full_table)/(full_table[1:length(table(bank_col))]+full_table[(length(table(bank_col))+1):
(2*length(table(bank_col)))]),digits=0)
  print(tab)
  print(colnames(bank)[cat_vars[i]])

  dat <- as.data.frame(tab)
  colnames(dat) = c(colnames(bank)[cat_vars[i]],"Subscribed?","Adjusted Percentage")
  if (max(dat[,3])-min(dat[,3])>50){
    scale = 10
  }
  else scale = 5
  # print(ggplot(dat, aes_q(x=as.name(names(dat)[1]), y=as.name(names(dat)[3]),
fill=as.name(names(dat)[2]))) +
  #   stat_summary(fun.y="mean", geom="bar",position="dodge") +
  #   scale_y_continuous(breaks = round(seq(0, max(dat[,3]), by = scale),1)))

  adjdat <- data.frame(dat[which(dat[,2]=="no"),c(1,3)])
  adjdat[,2] = dat[which(dat[,2]=="yes"),3]-dat[which(dat[,2]=="no"),3]
  adjdat[,3] = "Likely Yes (>=0)"
  adjdat[which(adjdat[,2]<0),3] = "Likely No (<0)"
  colnames(adjdat) <- c(colnames(adjdat)[1],"Likelihood of Term Deposit (%Yes-No)",">=0 OR <0")
  if (max(adjdat[,2])-min(adjdat[,2])>50){
    scale = 10
  }
  else scale = 5
  print(ggplot(adjdat, aes_q(x=as.name(names(adjdat)[1]),
y=as.name(names(adjdat)[2]),fill=as.name(names(adjdat)[3]))) +
    stat_summary(fun.y="mean", geom="bar") +
      labs(title="Bar Chart of Likelihood Percentages") +
    scale_y_continuous(breaks = round(seq(min(adjdat[,2]), max(adjdat[,2]), by = scale),1)))

  if (i==1){
    print(ggplot(adjdat, aes_q(x=as.name(names(adjdat)[1]),
y=as.name(names(adjdat)[2]),fill=as.name(names(adjdat)[3]))) +
    stat_summary(fun.y="mean", geom="bar") +
    labs(title="Bar Chart of Likelihood Percentages") +
```

```
   scale_x_discrete(labels = abbreviate) +
   scale_y_continuous(breaks = round(seq(min(adjdat[,2]), max(adjdat[,2]), by = scale),1)))
  }
}


# Numerical Variables
num_vars = c(1,6,9,11,12,13)

for (i in 1:6){
  cat("\n\n",colnames(bank)[num_vars[i]], ": ","\n","Yes: ","\n",
          summary(bank[which(bank[,15]=="yes"),num_vars[i]])[4],"\n",
          "No: ","\n" ,
          summary(bank[which(bank[,15]=="no"),num_vars[i]],"\n")[4])
}
summary(bank$age)
summary(bank$balance)
summary(bank$campaign)
summary(bank$day)
summary(bank$pdays)
summary(bank$previous)

for (i in 1:6){
  print(ggplot(data=bank,
aes_q(x=as.name(names(bank)[num_vars[i]]),fill=as.name(names(bank)[15]))) +
        labs(title="Distribution of Values for Subcribers (Y) & Unsubscribers (N)") +
        geom_histogram(aes_q(x=as.name(names(bank)[num_vars[i]]),y=as.name("..density..")),
position="dodge"))
}

ggplot(data=bank[which(bank$campaign>4),], aes(x=campaign,fill=y)) + labs(title="Distribution of
Values for Subcribers (Y) & Unsubscribers (N)") + geom_histogram(aes(x=campaign,y=..density..),
position="dodge")
ggplot(data=bank[which(bank$pdays>2),], aes(x=pdays,fill=y)) +labs(title="Distribution of Values for
Subcribers (Y) & Unsubscribers (N)") + geom_histogram(aes(x=pdays,y=..density..),
position="dodge")
ggplot(data=bank[which(bank$previous>2),], aes(x=previous,fill=y)) + labs(title="Distribution of
Values for Subcribers (Y) & Unsubscribers (N)") + geom_histogram(aes(x=previous,y=..density..),
position="dodge")
ggplot(data=bank, aes(x=balance,fill=y)) + labs(title="Distribution of Values for Subcribers (Y) &
Unsubscribers (N)") + geom_histogram(aes(x=previous,y=..density..), position="dodge")

round(cor(bank[,num_vars],method="pearson"),2)

for (i in 1:6){
  cat(colnames(bank[num_vars[i]]),":", t.test(bank[which(bank[,15]=="no"),num_vars[i]],
            bank[which(bank[,15]=="yes"),num_vars[i]])$p.value, "\n")
  i = i+1
}

# normalise all numerical variables
```

```
for (i in 1:6){
  bank[,num_vars[i]] = (bank[,num_vars[i]]-min(bank[,num_vars[i]]))/(max(bank[,num_vars[i]])-
min(bank[,num_vars[i]]))
}

set.seed(100)
N <- nrow(bank)
splits <- rep(1:3,ceiling(N/3))
splits <- sample(splits)
splits <- splits[1:N]
# Run through rest using splits==3 then 2 then 1
indtest <- (1:N)[(splits==3)]
indrest <- setdiff(1:N,indtest)
n <- length(indrest)
K <- 10
num = ceiling(n/K)
pred<-matrix(NA,length(indrest),8)
pred[,1] = bank[indrest,15]

for (iter in 1:K){
  ind = (iter*num-(num-1)):(min(iter*num,n))
  indvalid <- indrest[ind]
  indtrain <- setdiff(indrest,indvalid)

  fit.tree <- rpart(y~.,data=bank,subset=indtrain)
          pred[ind,2] <- predict(fit.tree,type="class",newdata=bank[indvalid,])

          fit.log <- multinom(y~.,data=bank,subset=indtrain)
          pred[ind,3] <- predict(fit.log,type="class",newdata=bank[indvalid,])

          fit.bag <- bagging(y~.,data=bank,subset=indtrain)
          pred[ind,4] <- predict(fit.bag,type="class",newdata=bank[indvalid,])$class

          fit.boost <- boosting(y~.,data=bank,boos=FALSE,coeflearn="Freund",subset=indtrain)
          pred[ind,5] <- predict(fit.boost,type="class",newdata=bank[indvalid,])$class

          fit.rf <- randomForest(y~.,data=bank,subset=indtrain)
          pred[ind,6] <- predict(fit.rf,type="class",newdata=bank[indvalid,])
}

pred[which(pred[,4]=="no"),4] = "1"
pred[which(pred[,5]=="no"),5] = "1"
pred[which(pred[,4]=="yes"),4] = "2"
pred[which(pred[,5]=="yes"),5] = "2"
pred[,8] = "1"
pred[,7] = "1"
rand = runif(length(indtest),0,1)
pred[which(rand<=0.1),7] = "2"
results <- matrix(NA,7,9)
results <- data.frame(results)
rownames(results)<-c("CA","No_Recall", "No_Precision", "No_F1",
```

```
                "Yes_Recall", "Yes_Precision", "Yes_F1")
colnames(results)<-c("Tree","Log", "Bag", "Boost","Forest","Random","ZeroR","Max","Margin %")
for (i in 1:6){
  tab <- table(pred[,1],pred[,i+1]) # rows=truth, cols=prediction
  results[1,i] = round(sum(diag(tab))*100/sum(tab),3) # Total Classification Accuracy
  # NO
  results[2,i] = round(tab[1]*100/sum(tab[c(1,3)]),3) # Recall
  results[3,i] = round(tab[1]*100/sum(tab[c(1,2)]),3) # Precision
  results[4,i] = round(2*results[3,i]*results[2,i]/(results[3,i]+results[2,i]),3) # f1
  # Yes
  results[5,i] = round(tab[4]*100/sum(tab[c(2,4)]),3) # Recall
  results[6,i] = round(tab[4]*100/sum(tab[c(3,4)]),3) # Precision
  results[7,i] = round(2*results[6,i]*results[5,i]/(results[6,i]+results[5,i]),3) # f1
}

tab <- table(pred[,1],pred[,8]) # rows=truth, cols=prediction
results[1,7] = round(sum(diag(tab))*100/sum(tab),3) # Total Classification Accuracy
# NO
results[2,7] = 100 # Recall
results[3,7] = round(tab[1]*100/sum(tab[c(1,2)]),3) # Precision
results[4,7] = round(2*results[3,7]*results[2,7]/(results[3,7]+results[2,7]),3) # f1
# Yes
results[5,7] = 0 # Recall
results[6,7] = 0 # Precision
results[7,7] = 0 # f1

for (r in 1:nrow(results)){
  results[r,8] = colnames(results)[which.is.max(results[r,1:7])]
  results[r,9] = round(100*(as.double(max(results[r,1:7]))-
as.double(sort(results[r,1:7])[4]))/(as.double(sort(results[r,1:7])[4])),2)
}

colnames(pred) = c("True","Tree","Log","Bag","Boost","RF","ZeroR","Random")
for (i in 2:7){
  for (j in (i+1):8){
    n00 = n11 = n10 = n01 = whoops = 0
    for (r in 1:nrow(pred)){
      if ((!(pred[r,1]==pred[r,i]))&&
          (pred[r,i]==pred[r,j])){
        n00 = n00 + 1
      } else if ((pred[r,1]==pred[r,j])&&
            (pred[r,i]==pred[r,j])) {
        n11 = n11 + 1
      } else if ((pred[r,1]==pred[r,j])&&(!(pred[r,i]==pred[r,j]))){
        n10 = n10 + 1
      } else if ((pred[r,1]==pred[r,i])&&(!(pred[r,i]==pred[r,j]))){
        n01 = n01 + 1
      } else {
        whoops = whoops + 1
      }
    }
```

```
  if(whoops>0|n01+n00+n10+n11!=nrow(pred)){print("WHOOPS")}
  mcnemar <- ((abs(n01-n10)-1)^2)/(n01+n10)
  cat(colnames(pred)[i],"&",colnames(pred)[j],": ",mcnemar,"\n")
  }
}

fitboost <- boosting(y~.,data=bank,boos=FALSE,coeflearn="Freund",subset=indrest)
boost_pred <- predict(fitboost,type="class",newdata=bank[indtest,])$class
boost_tab <- table(bank[indtest,15],boost_pred)
round(100*sum(diag(boost_tab))/sum(boost_tab),2)
# ZeroR
#round(100*table(bank[indtest,15])[1]/sum(table(bank[indtest,15])),2)
# Random
# rand = runif(length(indtest),0,1)
# rand_pred = c(rep("no",length(indtest)))
# rand_pred[which(rand<=0.1)] = "yes"
# rand_tab <- table(bank[indtest,15],rand_pred)
# round(100*sum(diag(rand_tab))/sum(rand_tab),2)


# ROC Curve & AUC for Test Set
# Boost
fitboost <- boosting(y~.,data=bank,boos=FALSE,coeflearn="Freund",subset=indrest)
boost_prob <- predict(fitboost,type="class",newdata=bank[indtest,])$prob
colnames(boost_prob) = c("Prob(No)","Prob(Yes)")
fitbag <- bagging(y~.,data=bank,subset=indrest)
bag_prob <- predict(fitbag,type="class",newdata=bank[indtest,])$prob
colnames(bag_prob) = c("Prob(No)","Prob(Yes)")

truth = matrix(0,length(indtest),2)
colnames(truth) = c("No=1","Yes=1")
truth[which(bank[indtest,15]=="no"),1]=1
truth[which(bank[indtest,15]=="yes"),2]=1
no_roc_pred <- prediction(boost_prob[,1], truth[,1])
no_perf <- performance(no_roc_pred, "tpr", "fpr")
yes_roc_pred <- prediction(boost_prob[,2], truth[,2])
yes_perf <- performance(yes_roc_pred, "tpr", "fpr")
no_roc_pred2 <- prediction(bag_prob[,1], truth[,1])
no_perf2 <- performance(no_roc_pred2, "tpr", "fpr")
yes_roc_pred2 <- prediction(bag_prob[,2], truth[,2])
yes_perf2 <- performance(yes_roc_pred2, "tpr", "fpr")

plot(no_perf, col="blue",main="ROC Curve for Predicting No to Term Deposit")
plot(no_perf2, add=TRUE, col="purple")
legend("bottomright",legend=c("Boosting", "Bagging"),
     col=c("blue", "purple"), lty=1:1, cex=0.8)
abline(0,1,col="red",lty=2)

plot(yes_perf, col="blue",main="ROC Curve for Predicting Yes to Term Deposit")
plot(yes_perf2, add=TRUE, col="purple")
legend("bottomright",legend=c("Boosting", "Bagging"),
```

```
     col=c("blue", "purple"), lty=1:1, cex=0.8)
abline(0,1,col="red",lty=2)

AUC = matrix(0,2,2)
colnames(AUC) = c("No","Yes")
rownames(AUC) = c("Boost","Bag")
AUC[1,1] = auc(truth[,1],boost_prob[,1])
AUC[1,2] = auc(truth[,2],boost_prob[,2])
AUC[2,1] = auc(truth[,1],bag_prob[,1])
AUC[2,2] = auc(truth[,2],bag_prob[,2])
print(AUC)

# variable importance plot (for both gain in Gini Index)
imp_boost = sort(fitboost$importance,decreasing = TRUE)
imp_bag = sort(fitbag$importance,decreasing = TRUE)
imp = data.frame(matrix(NA,length(colnames(bank[,1:14])),3))
imp[,1] = colnames(bank[,1:14])
for (i in 1:nrow(imp)){
  imp[i,2] = eval(parse(text=paste0("imp_boost['",imp[i,1],"']",sep="")))
  imp[i,3] = eval(parse(text=paste0("imp_bag['",imp[i,1],"']",sep="")))
}
colnames(imp) = c("Variables","Boost","Bag")

ggplot(imp, aes(x=reorder(Variables, -Boost), y=Boost),main="Bar Plot of variable Importance") +
  stat_summary(fun.y="mean", geom="bar") +
  scale_x_discrete(labels = abbreviate) +
  labs(title="Bar Chart of Variable Importance in Boosting Model",
       x ="Variables", y = "Mean Increase in Gini Index")

ggplot(imp, aes(x=reorder(Variables, -Bag), y=Bag),main="Bar Plot of Variable Importance") +
  stat_summary(fun.y="mean", geom="bar") +
  scale_x_discrete(labels = abbreviate) +
  labs(title="Bar Chart of Variable Importance in Bagging Model",
       x ="Variables", y = "Mean Increase in Gini Index")

nor_imp = imp
nor_imp[,2] = nor_imp[,2]/sum(nor_imp[,2])
nor_imp[,3] = nor_imp[,3]/sum(nor_imp[,3])
ggplot(melt(nor_imp, id=c("Variables")), aes(x=Variables, y=value, fill=variable),main="Bar Plot of
Variable Importance") +
  stat_summary(fun.y="mean", geom="bar", position="dodge") +
  scale_x_discrete(labels = abbreviate) +
  labs(title="Bar Chart of Relative Variable Importance in Both Models",
       x ="Variables", y = "Normalised Mean Increase in Gini Index")

# closer look at model
fit.tree <- rpart(y~.,data=bank,subset=indrest)
plot(fit.tree,margin=0.05,main="Decision Tree")
text(fit.tree, use.n=TRUE, all=TRUE, cex=.7,col="black")

fit.log <- multinom(y~.,data=bank,subset=indrest)
```

```
ncoef = length(summary(fit.log)$coefficients)
logcoef = data.frame(matrix(0,ncoef,6))
logcoef[,1] = names(summary(fit.log)$coefficients)
logcoef[,2] = summary(fit.log)$coefficients
logcoef[,4] = summary(fit.log)$coefficients/summary(fit.log)$standard.errors
for (i in 1:ncoef){
  if (logcoef[i,2]<0){
    logcoef[i,3]="Negative"
  }  else {
    logcoef[i,3]="Positive"
  }
  logcoef[i,5] = (1 - pnorm(abs(logcoef[i,4]), 0, 1)) * 2
  if (logcoef[i,5]<0.05){
    logcoef[i,6]="Yes"
  }  else {
    logcoef[i,6]="No"
  }
}
rankings = rank(logcoef[,5])
colnames(logcoef) = c("Variables","Coefficients","Effect on Subscribership",
                      "Z-Scores","PValues","Significant")
logcoef <- logcoef[order(logcoef$PValues),]
rownames(logcoef) <- NULL
logcoef

toplogcoef <- logcoef[which(logcoef[,6]=="Yes"),]
toplogcoef


# try best model on the test set
fitboost <- boosting(y~.,data=bank,boos=FALSE,coeflearn="Freund",subset=indrest)
boost_pred <- predict(fitboost,type="class",newdata=bank[indtest,])$class
boost_tab <- table(bank[indtest,15],boost_pred)
tab <- boost_tab
ca = round(sum(diag(tab))*100/sum(tab),3) # Total Classification Accuracy
# No
rn = round(tab[1]*100/sum(tab[c(1,3)]),3) # Recall
pn = round(tab[1]*100/sum(tab[c(1,2)]),3) # Precision
fn = round(2*rn*pn/(rn+pn),3) # f1
# Yes
ry = round(tab[4]*100/sum(tab[c(2,4)]),3) # Recall
py = round(tab[4]*100/sum(tab[c(3,4)]),3) # Precision
fy = round(2*ry*py/(ry+py),3) # f1
cat(ca,rn,pn,fn,ry,py,fy)

fitboost <- boosting(y~.,data=bank,boos=FALSE,coeflearn="Freund")
boost_pred <- predict(fitboost,type="class")$class
boost_tab <- table(bank[indtest,15],boost_pred)
tab <- boost_tab
ca = round(sum(diag(tab))*100/sum(tab),3) # Total Classification Accuracy
# No
```

```
rn = round(tab[1]*100/sum(tab[c(1,3)]),3) # Recall
pn = round(tab[1]*100/sum(tab[c(1,2)]),3) # Precision
fn = round(2*rn*pn/(rn+pn),3) # f1
# Yes
ry = round(tab[4]*100/sum(tab[c(2,4)]),3) # Recall
py = round(tab[4]*100/sum(tab[c(3,4)]),3) # Precision
fy = round(2*ry*py/(ry+py),3) # f1
cat(ca,rn,pn,fn,ry,py,fy)
```