Predicting the Effectiveness of Bank Marketing

Nicholas Porrone 250918147

April 19th, 2019

Introduction

The purpose of this study is to predict if the client of a bank will subscribe a term deposit based on data obtained by a direct marketing campaigns (phone calls) conducted by a Portuguese banking institution. This data was initially collected by Sérgio Moro and Paulo Rita of ISCTE-IUL, Business Research Unit. Joining them in the study is Paulo Cortez of ALGORITMI, Research Centre. They too had a goal to propose a data mining approach to predict the success of telemarketing calls for selling bank long-term deposits. Their analysis is referenced in the appendix at the bottom of the document.

Data Summary

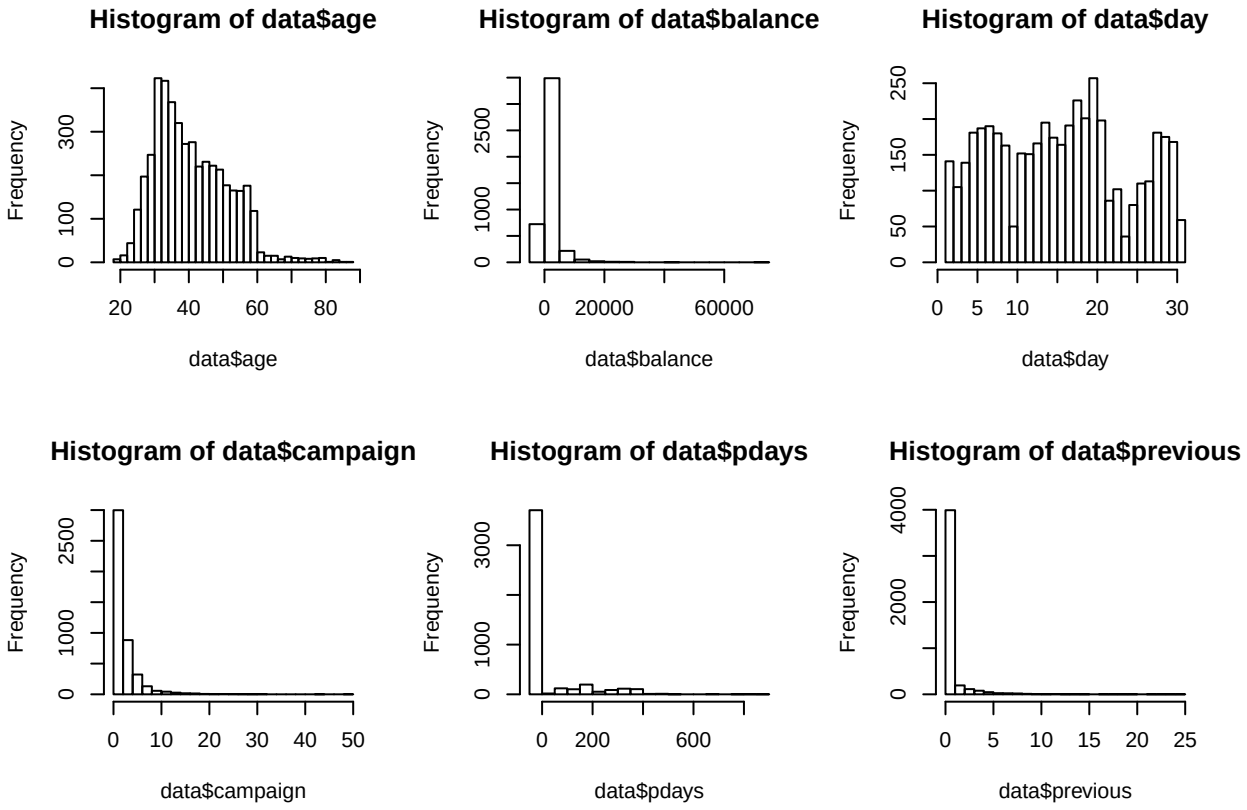
The data set was randomly sampled from a larger data set containing 45211 observations with 16 input variables. The current data set is only a subset of the original containing 4521 observations (10% of the original data set). As suggested by the previous study, we will exclude the duration input variable because based on its result (if duration=0) we can say with certainty that some clients will not participate in the term deposit. This would not be useful for predictive models to be construct and thus we will omit this variable. The other 15 input variables are described as follows:

Variable	Definition
`age`	The age of the client (Continous)
`job`	The type of job the client works at (Categorical)
`marital`	The marital status of the client (Categorical)
`education`	The maxiumum level of education achieved by the client (Categorical)
`default`	Does the client currently have credit in default (Categorical)
`balance`	Average yearly balance in Euros (Continous)
`housing`	Does the client have a housing loan (Categorical)
`loan`	Does the client have a personal loan (Categorical)
`contact`	How did we get in contact with the client (Categorical)
`month`	Which month did we contact the client last (Categorical)
`day`	Which day of the month did we contact the client last (Continous)
`campaign`	Number of contacts made with the client during the campaign (Continous)
`pdays`	The number of days that have passed by since the client was contacted for a previous campaign (Continous)
`previous`	The number of contacts made with client before this campaign (Continous)
`poutcome`	The outcome of the previous campaign (Categorical)
`y` (Response)	Does the client accept a term deposit (Categorical)

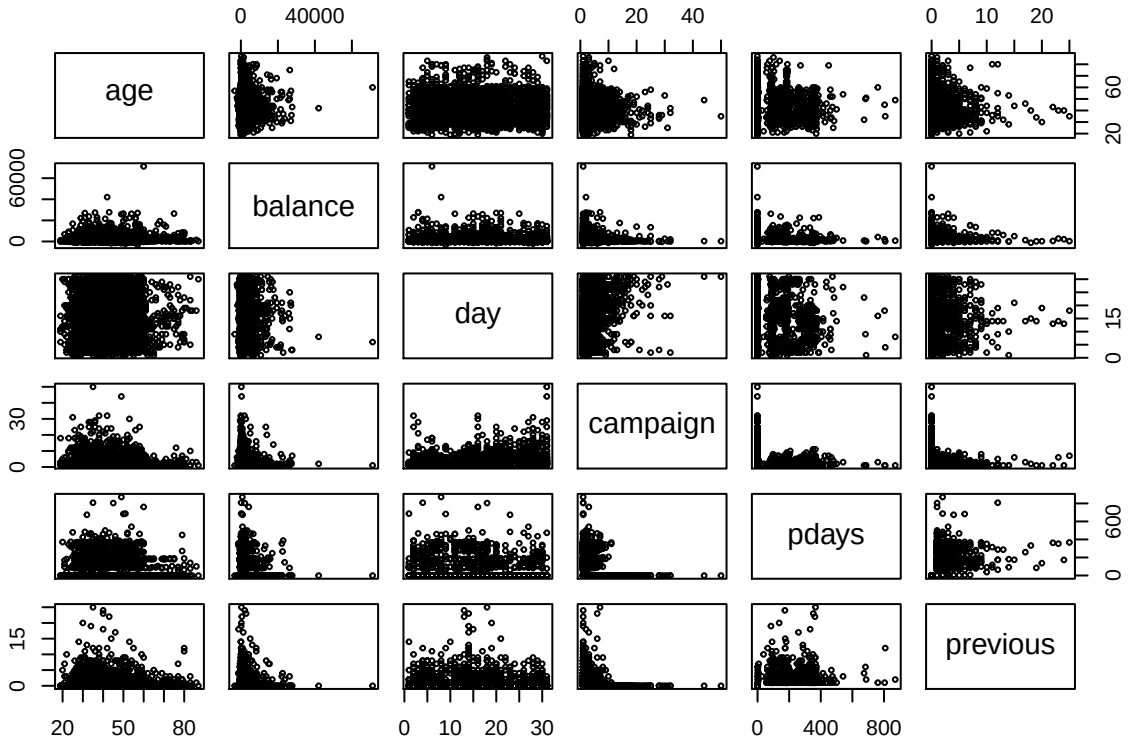
We note that of 16 input variables, 9 of them are categorical variables while the other 6 are continuous variables. We look to see the data summary of the continuous variables:

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
age	19.00	33.00	39.00	41.17	49.00	87.00
balance	-3313.00	69.00	444.00	1422.66	1480.00	71188.00
day	1.00	9.00	16.00	15.92	21.00	31.00
campaign	1.00	1.00	2.00	2.79	3.00	50.00
pdays	-1.00	-1.00	-1.00	39.77	-1.00	871.00
previous	0.00	0.00	0.00	0.54	0.00	25.00

Table 1: The Summaries of the Continous variables.



With these histograms, we can see that aside from predictor "day", all the continuous variables are heavily skewed to the right. Looking deeper we can see if any of these variables are correlated with each other.



Through the correlation chart we can see none of the variables share a true positive linear relationship.

In order to proceed with this classification problem we must split the data into a training and a testing set. It was decided to split that data such that 80% of the observations would be classified as training data and the remaining 20% would be classified as testing data. This corresponds to a training set of length 3617 and a test set of length 904.

Methods of Classification

Logistic Regression

With a binary classification problem we apply a logistic regression model to the data to predict input variables as either being 0 or 1. Looking at the full model first, we have a confusion matrix of:

Y-hat	No	Yes
0	795	84
1	7	18

With this matrix we calculated the classification error of the model to be 0.1006637. We can use this value for reference after we preform some variable selection methods in order to see if some reduced models have better error rates.

Variable Selection

Starting with the full model we can preform a variable selection method in order to find the best model. A stepwise selection with the Bayes Information Criterion(BIC) was used to find the best model. It is noted that BIC was chosen over AIC (Akaike Information Criterion) because the BIC has a larger penalty to larger models and our full model is very large. An AIC model was tested however it was quickly disregarded because a model that would purely suggest no (for the response) had a better mean squared error. The new model (using BIC) is composed of the following variables:

Variable	Definition
`marital`	The marital status of the client (Categorical)
`housing`	Does the client have a housing loan (Categorical)
`loan`	Does the client have a personal loan (Categorical)
`contact`	How did we get in contact with the client (Categorical)
`poutcome`	The outcome of the previous campaign (Categorical)

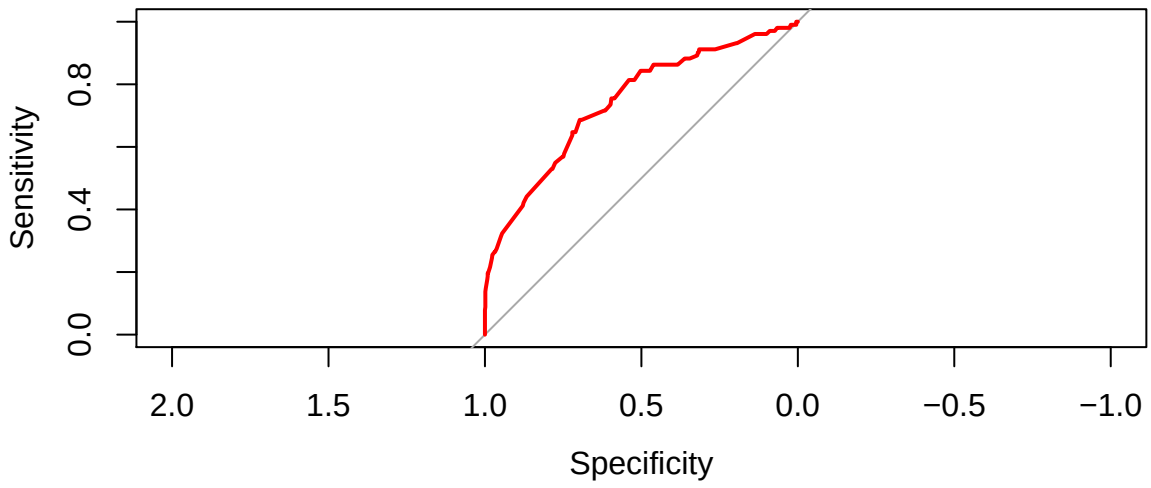
It is important to note that our new model does not contain any of the continous variables which we orginally started with. The confusion matrix to this model is listed below:

Y-hat	No	Yes
0	795	83
1	7	19

With this matrix we calculated the classification error of the model to be 0.0995575. Referencing the full model, we can see that our reduced model appears predicts as well as or perhaps better than the full model. Taking a deeper approach, a 5-fold cross validation on each model was applied to see if the reduced model is better in general. The results showed that the reduced model had an accuracy of 89.25% (.1075 classification error) while the full model had an accuracy of 89.12% (0.1088). Again we can see that the error rates are very comparable between the two despite the reduced model having much fewer predictors.

RoC Curve

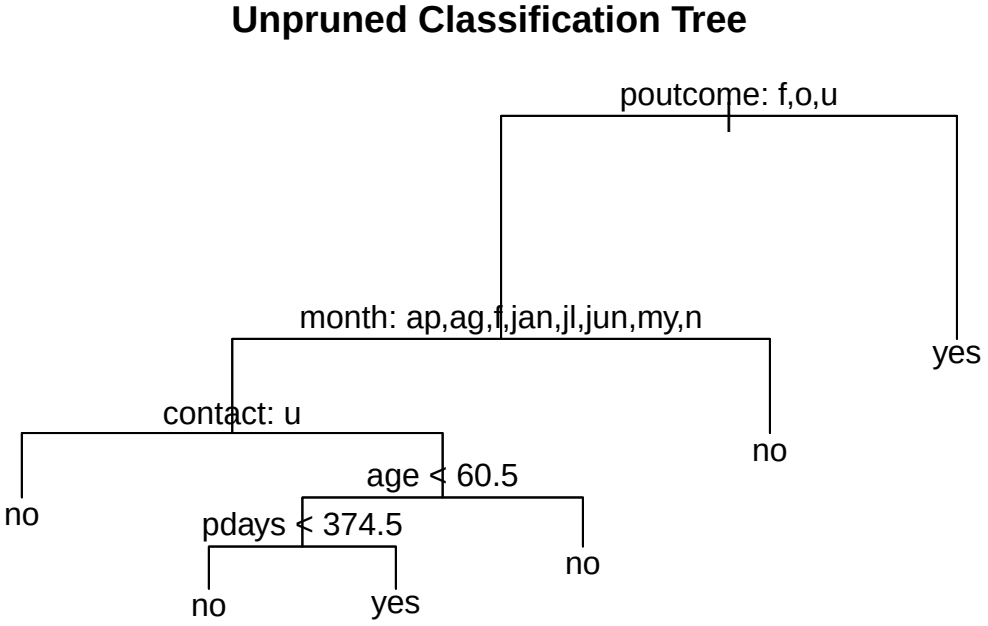
All of our models were created using a cutoff value of 0.5. To examine the change of the cutoff value an RoC curve was built and is shown below. It is noted that this RoC Curve has an AUC of 0.7421 :



Decsion Trees

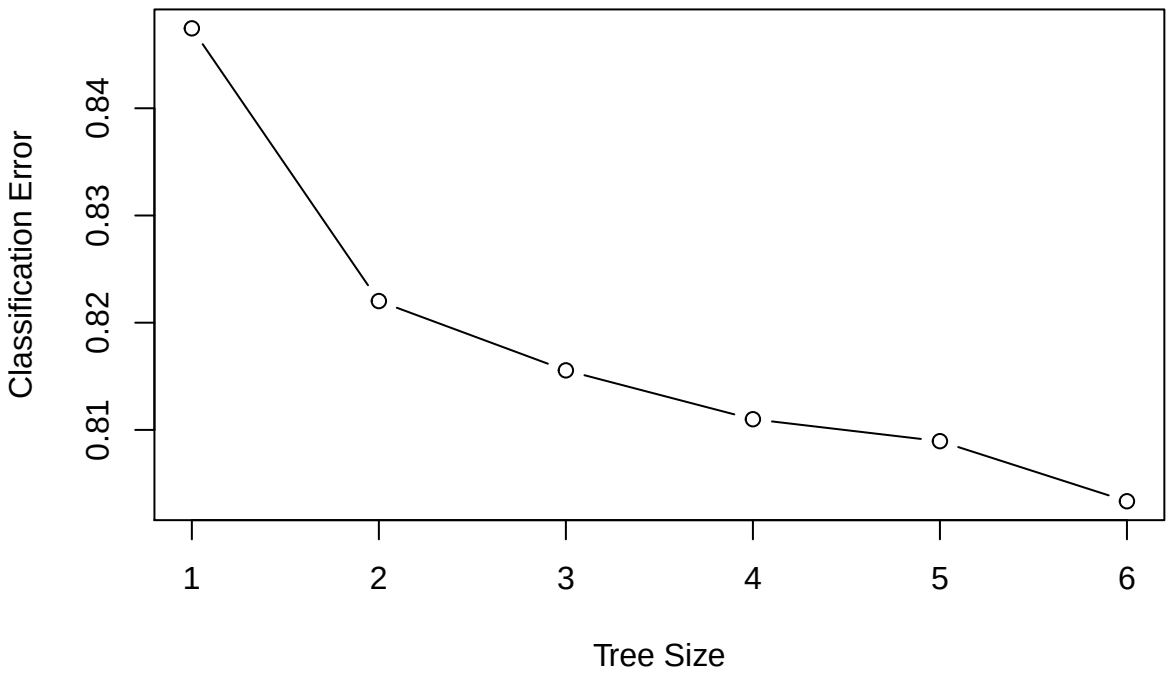
Single Tree

A decision tree is a method of making several binary splits or "rules" in order to attempt to classify an observation. In this case, we can apply a decision tree on our data set to classify bank clients as investors or non-investors (response variable y). The tree is listed as follows:



From this tree we can see that of the 6 terminal nodes only 2 of them result in the bank client taking the loan. We can see that the initial split is very important as one of the outcomes of it is a terminal node resulting in yes. The initial split is on if the previous outcome of a past study ended as failure, unknown result, or other. In other words, if the client had a success with a previous marketing campaign, then they are predicted to be investors in the current campaign.

It should be noted that the tree above is an unpruned classification tree. When attempting to prune this tree, through cross-validation it was concluded that the tree did not need to be pruned. Below depicts the relationship between classification error and the size of our tree:



The most optimal tree size is 6, the length of our unpruned tree. Since it is known that the unpruned tree is the best, further analysis can now be conducted. Below is the confusion matrix for our unpruned tree:

Y-hat	No	Yes
0	792	81
1	10	21

The classification error for this method is 0.1006637. Coincidently, this is the same error as our full logistic model.

Bagging

Bagging, also known as bootstrap aggregating is another method that may be applied to a binary classification problem. In bagging, a bootstrap sample is taken from your data set and then all the different results are combined and averaged. In other words, bagging builds multiple decision trees and merges them together to get a more accurate and stable prediction. Applying bagging to our data we get a confusion matrix of:

Y-hat	No	Yes
0	780	80
1	22	22

With this matrix we calculated the classification error of the model to be 0.1128319. This has yielded the highest error of all methods tested this far.

Random Forest

Random Forest is another form of bagging which is applicable in a binary classification problem. In Random forests, only a subset of input variables are selected at random out of the total and the best split feature from the subset is used to split each node in a tree, unlike in bagging where all input variables are considered for splitting a node. In this problem we choose a subset of size 4 to apply the Random Forest method. Below is the confusion matrix:

Y-hat	No	Yes
0	788	86
1	14	16

The associated error with this confusion matrix is 0.1106195. Looking back at bagging, we can see that Random Forest has a better accuracy as its error rate is smaller. However, both techniques have not been better than a regular decision tree. This may be because our model may be underfitting the data. Our next method, boosting, is used to fit underfit models.

Boosting

As stated previously, boosting can be used to accurately predict underfit models. In boosting, the starting model should be a simple "weak" model (high training error). Then by fitting more simple weak models and combining them the error will gradually be reduced each time a new model is fit. Apply boosting to the data, we achieve a confusion matrix of:

Y-hat	No	Yes
0	791	84
1	11	18

The error associated with this model is 0.1050885. This is better than bagging and random forest however it was worse than the single decision tree. Thus, we can conclude that our model was neither overfitted or underfitted. Continuing to the next section of you can see a summary of all the results in this study.

Method Summary

The table below shows a summary of all methods tested in this study. It is noted that the reduced logistic model yields the best Classification Error and Sensitivity out of all methods tested. Another interesting fact is, of all decision tree methods, the Single Tree method yielded the highest Classification Error, Sensitivity, and Specificity.

Method	Classification Error	Sensitivity	Specificity
Logistic Regression (Full Model)	0.1006637	0.72	0.904437
Logistic Regression (Reduced Model)	0.0995575	0.730769	0.905467
Single Tree	0.1006637	0.677419	0.907216
Bagging	0.1128319	0.5	0.906977
Random Forest	0.1106195	0.533333	.9016018
Boosting	0.1050885.	.6206897	.904

Limitations

As stated previously, the current data is a subset of the original (10% of the size). This is since the computer used in this study did not have enough CPU power to test efficiently on the full data set. In the previous study (referenced in the appendix) the results to their logistic regression and decision trees were better than the ones done in this study. The techniques they used were not stated and thus limited this study to the results that were made.

Appendix

References

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

R Code

```
# Data Summary

data=read.csv("C:/Users/nporr/OneDrive/Desktop/Bank_Marketing/Bank.csv",sep=";")
data = data[,-12]  # get rid of duration
data
table = as.data.frame(apply(FUN=summary,X=data[,c(1,6,10,12,13,14)],MARGIN = 2)) # Create a table summa
print(xtable(table, caption= "The Summaries of the Continous variables."))

# plot histogram for the continous variables
par(mfrow= c(2,3))
hist(data$age,breaks = 25)
hist(data$balance,breaks = 25)
hist(data$day,breaks = 25)
hist(data$campaign,breaks = 25)
hist(data$pdays,breaks = 25)
hist(data$previous,breaks = 25)

pairs(data[,c(1,6,10,12,13,14)]) # check for correlation between continous

# Data Split

split = sample(length(data$y), round(length(data$y)*0.8), rep=FALSE) # 80% training, 20% test
data_trn = data[split, ]
data_tst = data[-split, ]

# Methods

## Logistic Regression

### BIC Aprroach

logmodelf = glm(y ~ ., data = data_trn, family = binomial(link = "logit"))
logbicfit = step(logmodelf, direction = "both",k=log(nrow(data_trn)),scope = list(upper = ~.,lower= ~1)
```

```r
summary(logmodelf)
summary(logbicfit)

# full model

cutoff = 0.5
y_hatf = rep(0,nrow(data_tst))
idxf = which(predict(logmodelf,data_tst,type = "response") > cutoff )
y_hatf[idxf] = 1

confful = table(y_hatf, data_tst$y)
confful

ErrorRtf = 1 - sum(diag(confful)/sum(confful))
ErrorRtf

## CV on full

control = trainControl(method="cv", number=5)
logmodelfcv= train(y ~ .,data = data,trControl = control, method = "glm", family=binomial())

# reduced model

y_hat = rep(0,nrow(data_tst))
idx = which(predict(logbicfit,data_tst,type = "response") > cutoff )
y_hat[idx] = 1

conf = table(y_hat, data_tst$y)
conf

ErrorRt = 1 - sum(diag(conf)/sum(conf))
ErrorRt

## CV on reduced

controlr = trainControl(method="cv", number=5)
logmodelcvr= train(y ~ housing + loan + contact + campaign + poutcome,data = data,trControl = controlr,

# RoC Curve

logpred = predict(logbicfit,data_tst,type="response")
RoC= roc(data_tst$y~logpred,plot=T,col="red")
pROC::auc(RoC)

## Decsion Trees

#unpruned tree

dtree = tree(y ~ ., data = data_trn)
plot(dtree)
text(dtree, pretty = 1)
title(main = "Unpruned Classification Tree")
```

```r
## cv to see the best tree size
dtree.cv=cv.tree(dtree)
plot(dtree.cv$size,sqrt(dtree.cv$dev/nrow(data_trn)),type="b",xlab="Tree Size",ylab="Classification Err

dtree_pred = predict(dtree,newdata = data_tst,type="class")

conf = table(dtree_pred, data_tst$y)
conf

ErrorRt = 1 - sum(diag(conf)/sum(conf))
ErrorRt


## Pruned tree did not help

## Bagging

bag = randomForest(y ~ ., data = data_trn, mtry = 15, importance = TRUE, ntree = 500)
bag

bag_pred = predict(bag, newdata = data_tst)

# test error (bagging)

conf = table(bag_pred, data_tst$y)
conf

ErrorRt = 1 - sum(diag(conf)/sum(conf))
ErrorRt


## Random Forest

rf = randomForest(y ~ ., data = data_trn, mtry = 4, importance = TRUE, ntree = 500)
rf

rf_pred = predict(rf, newdata = data_tst)
# test error (bagging)

conf = table(rf_pred, data_tst$y)
conf

ErrorRt = 1 - sum(diag(conf)/sum(conf))
ErrorRt

# Boosting

boost = gbm((unclass(y) - 1) ~ ., data = data_trn, distribution = "bernoulli", n.trees = 500, interacti
boost_pred = predict(boost, newdata = data_tst, n.trees = 500, type="response")
boost_pred[boost_pred >= 0.5] = "yes"
boost_pred[boost_pred < 0.5] = "no"

# test error(boosting)
```

```
conf = table(boost_pred, data_tst$y)
conf

ErrorRt = 1 - sum(diag(conf)/sum(conf))
ErrorRt
```