finalProject\_yavuz\_karabiyik

Yavuz Numan Karabiyik

2/15/2021

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

## Part One - Project Description

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed.

There is dataset bank-full.csv with all examples, ordered by date (from May 2008 to November 2010). The classification goal is to predict if the client will subscribe a term deposit (variable y).

ROC (Receiver Operating Characteristic) Curve is a way to visualize the performance of a binary classifier.

TPR or True Positive Rate answers the question — When the actual classification is positive, how often does the classifier predict positive? FPR or False Positive Rate answers the qestion — When the actual classification is negative, how often does the classifier incorrectly predict positive?

The questions that we want to answer is,

Can we predict loan status which is described as loanyes using other features considering age, job, martial, education, balance, housing, contact, loan, day, month, duration, campaign, pdays, previous, poutcome, and others? If yes, which classifier is the best fit for the case using ROC curve.

## Part One -Loading and Prepare Data

###1.a Reading Data

bank\_df<- read.csv("./bank-full.csv", sep = ";")

###1.b Label Decoding

In simple terms, label encoding is the process of replacing the different levels of a categorical variable with dummy numbers. For instance, the variable loanyes has two levels, “yes” and “no”. These can be encoded to 1 and 0, respectively.

bank\_df <- bank\_df %>%  
 mutate(housing = ifelse(housing == "no",0,1))

###1.c One Hot Decoder

In this technique, one-hot (dummy) encoding is applied to the features, creating a binary column for each category level and returning a sparse matrix. In each dummy variable, the label “1” will represent the existence of the level in the variable, while the label “0” will represent its non-existence.

We will apply this technique to all the remaining categorical variables. The decoding comes from caret package, while the first line uses the dummyVars() function to create a full set of dummy variables. The dummyVars() method works on the categorical variables. It is to be noted that the first line contains the argument fullrank=T, which will create n-1 columns for a categorical variable with n unique levels.

The second line uses the output of the dummyVars() function and transforms the dataset, dat, where all the categorical variables are encoded to numerical variables. The fourth line of code prints the structure of the resulting data, dat-transfored, which confirms that one-hot encoding is completed.

dmy <-dummyVars("~ .", data = bank\_df, fullRank = T)  
bank\_df\_transformed<- data.frame(predict(dmy, newdata = bank\_df))  
  
bank\_df\_transformed<-na.omit(bank\_df\_transformed)

###1.d Checking Data Types

Using sapply method to see the data types of the column values. It is important because, models usually uses numeric values and it requries all data to be in same type. We can see that all the data is converted to numeric data type.

bank\_df\_transformed<-na.omit(bank\_df\_transformed)  
sapply(bank\_df\_transformed, class)

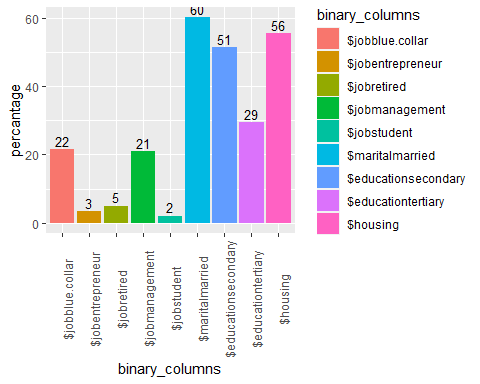
## age jobblue.collar jobentrepreneur jobhousemaid   
## "numeric" "numeric" "numeric" "numeric"   
## jobmanagement jobretired jobself.employed jobservices   
## "numeric" "numeric" "numeric" "numeric"   
## jobstudent jobtechnician jobunemployed jobunknown   
## "numeric" "numeric" "numeric" "numeric"   
## maritalmarried maritalsingle educationsecondary educationtertiary   
## "numeric" "numeric" "numeric" "numeric"   
## educationunknown defaultyes balance housing   
## "numeric" "numeric" "numeric" "numeric"   
## loanyes contacttelephone contactunknown day   
## "numeric" "numeric" "numeric" "numeric"   
## monthaug monthdec monthfeb monthjan   
## "numeric" "numeric" "numeric" "numeric"   
## monthjul monthjun monthmar monthmay   
## "numeric" "numeric" "numeric" "numeric"   
## monthnov monthoct monthsep duration   
## "numeric" "numeric" "numeric" "numeric"   
## campaign pdays previous poutcomeother   
## "numeric" "numeric" "numeric" "numeric"   
## poutcomesuccess poutcomeunknown yyes   
## "numeric" "numeric" "numeric"

###1.e Representing Binary Features

In order to represent the features in graph, it is better to categorize them with their value type considering if they are binary values or not. I have visualize the binary features with their percentages, so we can see what are the percentages of the customer’s job, their martial status, education and so on.

We can see some binary features, %22 of the customers have a blue collar job %3 of the customers are entrepreneur %5 of the customers are retired %21 of the customers are manager %2 of the customers are student %60 of the customers are married %51 of the customers have secondary degree %29 of the customers have tertiary %56 of the customers have house

binary\_features <- data.frame(  
 binary\_columns = factor(c(  
 "$jobblue.collar",  
 "$jobentrepreneur",  
 "$jobretired",  
 "$jobmanagement",  
 "$jobstudent",  
 "$maritalmarried",  
 "$educationsecondary",  
 "$educationtertiary",  
 "$housing"),  
 levels = c(  
 "$jobblue.collar",  
 "$jobentrepreneur",  
 "$jobretired",  
 "$jobmanagement",  
 "$jobstudent",  
 "$maritalmarried",  
 "$educationsecondary",  
 "$educationtertiary",  
 "$housing")),  
 percantage = c(  
 mean(bank\_df\_transformed$jobblue.collar)\*100,  
 mean(bank\_df\_transformed$jobentrepreneur)\*100,  
 mean(bank\_df\_transformed$jobretired\*100),  
 mean(bank\_df\_transformed$jobmanagement\*100),  
 mean(bank\_df\_transformed$jobstudent\*100),  
 mean(bank\_df\_transformed$maritalmarried\*100),  
 mean(bank\_df\_transformed$educationsecondary\*100),  
 mean(bank\_df\_transformed$educationtertiary\*100),  
 mean(bank\_df\_transformed$housing\*100)  
 ))  
  
  
b1 <- ggplot(data = binary\_features, aes(x = binary\_columns, y = percantage, fill = binary\_columns)) +  
 geom\_bar(stat = "identity") +  
 geom\_text(aes(label= round(percantage)), vjust=-0.3, color="black", size=3.5) +  
 theme(axis.text.x = element\_text(angle = 90))   
  
b1

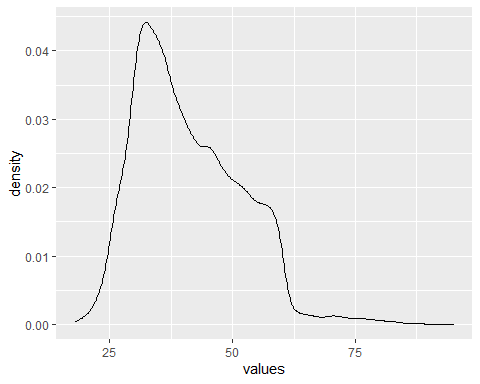
 ###1.f Representing Non-Binary Features

I have checked for missing values, but did not go into further pre -processing of data since my objective is to demonstrate ROC curves here and not model fine tuning.There are no missing values. We can find more details about the column stats using summary function for non-binary features.

summary(bank\_df\_transformed[c("age", "balance")])

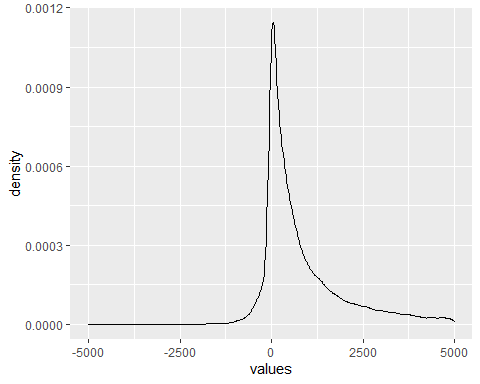
## age balance   
## Min. :18.00 Min. : -8019   
## 1st Qu.:33.00 1st Qu.: 72   
## Median :39.00 Median : 448   
## Mean :40.94 Mean : 1362   
## 3rd Qu.:48.00 3rd Qu.: 1428   
## Max. :95.00 Max. :102127

dfs<- stack(bank\_df\_transformed)  
dfs<- filter(dfs, ind == "age")  
ggplot(dfs, aes(x=values)) + geom\_density()



dfs<- stack(bank\_df\_transformed)  
dfs<- filter(dfs, ind == "balance")  
ggplot(dfs, aes(x=values)) + geom\_density() + xlim(-5000,5000)

## Warning: Removed 2847 rows containing non-finite values (stat\_density).

 ## Part Two - Splitting Data

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

Train Dataset: Used to fit the machine learning model. Test Dataset: Used to evaluate the fit machine learning model.

###2.a Splittig data to train and test data set

We can see that 33908 examples (75 percent) were allocated to the training set and 11303 examples (25 percent) were allocated to the test set, as we specified.

sample = sample.split(bank\_df\_transformed, SplitRatio = .75)  
  
train = subset(bank\_df\_transformed, sample == TRUE)  
test = subset(bank\_df\_transformed, sample == FALSE)  
  
dim(train)

## [1] 33645 43

dim(test)

## [1] 11566 43

## Part Three - ROC

ROC (Receiver Operating Characteristic) Curve is a way to visualize the performance of a binary classifier.

TPR or True Positive Rate answers the question — When the actual classification is positive, how often does the classifier predict positive? FPR or False Positive Rate answers the qestion — When the actual classification is negative, how often does the classifier incorrectly predict positive?

###3.a Logistic Regression

The code below estimates a logistic regression model using the glm (generalized linear model) function. Since we gave our model a name (mylogit), R will not produce any output from our regression. In order to get the results we use the summary command.

lr\_fit <- glm(loanyes~., data = train, family = binomial())  
lr\_predict <- predict(lr\_fit, newdata = test, type = "response")  
  
summary(lr\_fit)

##   
## Call:  
## glm(formula = loanyes ~ ., family = binomial(), data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.5407 -0.6282 -0.5124 -0.3358 3.5795   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.464e+00 1.780e-01 -8.227 < 2e-16 \*\*\*  
## age -1.676e-03 1.901e-03 -0.882 0.377832   
## jobblue.collar -1.147e-01 5.518e-02 -2.079 0.037630 \*   
## jobentrepreneur 2.213e-01 8.733e-02 2.534 0.011288 \*   
## jobhousemaid -4.720e-01 1.125e-01 -4.197 2.71e-05 \*\*\*  
## jobmanagement -1.637e-01 6.514e-02 -2.514 0.011953 \*   
## jobretired -1.294e-01 9.306e-02 -1.390 0.164443   
## jobself.employed -1.966e-01 9.711e-02 -2.024 0.042949 \*   
## jobservices -5.038e-02 6.265e-02 -0.804 0.421325   
## jobstudent -2.175e+00 3.096e-01 -7.024 2.15e-12 \*\*\*  
## jobtechnician -1.059e-02 5.600e-02 -0.189 0.850044   
## jobunemployed -8.423e-01 1.238e-01 -6.804 1.02e-11 \*\*\*  
## jobunknown -2.238e+00 5.872e-01 -3.811 0.000138 \*\*\*  
## maritalmarried 4.902e-03 4.767e-02 0.103 0.918099   
## maritalsingle -2.540e-01 5.668e-02 -4.481 7.42e-06 \*\*\*  
## educationsecondary 2.093e-01 4.912e-02 4.261 2.04e-05 \*\*\*  
## educationtertiary -3.798e-02 6.357e-02 -0.597 0.550186   
## educationunknown -7.104e-01 1.143e-01 -6.216 5.09e-10 \*\*\*  
## defaultyes 8.792e-01 8.763e-02 10.033 < 2e-16 \*\*\*  
## balance -1.402e-04 1.021e-05 -13.730 < 2e-16 \*\*\*  
## housing 3.198e-02 3.646e-02 0.877 0.380400   
## contacttelephone -2.004e-01 6.783e-02 -2.954 0.003133 \*\*   
## contactunknown 1.741e-01 5.401e-02 3.224 0.001264 \*\*   
## day -4.848e-03 2.143e-03 -2.262 0.023694 \*   
## monthaug -2.590e-01 8.914e-02 -2.905 0.003670 \*\*   
## monthdec -7.050e-01 3.967e-01 -1.777 0.075547 .   
## monthfeb 1.867e-01 9.748e-02 1.915 0.055470 .   
## monthjan 3.213e-01 1.132e-01 2.838 0.004536 \*\*   
## monthjul 1.083e+00 7.655e-02 14.154 < 2e-16 \*\*\*  
## monthjun 3.207e-02 9.505e-02 0.337 0.735800   
## monthmar -3.632e-01 2.387e-01 -1.522 0.128107   
## monthmay -1.733e-02 7.879e-02 -0.220 0.825954   
## monthnov 5.315e-01 8.467e-02 6.277 3.44e-10 \*\*\*  
## monthoct -1.347e-01 1.767e-01 -0.762 0.445858   
## monthsep -5.817e-01 2.334e-01 -2.492 0.012707 \*   
## duration 1.264e-04 6.647e-05 1.901 0.057309 .   
## campaign -1.308e-03 5.052e-03 -0.259 0.795666   
## pdays -1.568e-04 3.696e-04 -0.424 0.671461   
## previous 3.148e-02 1.005e-02 3.132 0.001734 \*\*   
## poutcomeother -9.770e-02 9.250e-02 -1.056 0.290889   
## poutcomesuccess -6.722e-01 1.442e-01 -4.661 3.15e-06 \*\*\*  
## poutcomeunknown -1.100e-01 1.089e-01 -1.009 0.312868   
## yyes -4.925e-01 6.775e-02 -7.269 3.61e-13 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 29740 on 33644 degrees of freedom  
## Residual deviance: 27607 on 33602 degrees of freedom  
## AIC: 27693  
##   
## Number of Fisher Scoring iterations: 6

###3.b Conditional Inference Trees

The party package provides non-parametric regression trees for nominal, ordinal, numeric, censored, and multivariate responses, which is called Conditional Inference Trees.

ct\_fit<-ctree(loanyes~., data = train)  
ct\_predict<- predict(ct\_fit, newdata = test)

###3.c Decision Tree

Decision tree is a type of supervised learning algorithm that can be used in both regression and classification problems. It works for both categorical and continuous input and output variables.

tr\_fit<-tree(loanyes~., data = train)  
tr\_predict<- predict(tr\_fit, newdata = test)  
  
summary(tr\_fit)

##   
## Regression tree:  
## tree(formula = loanyes ~ ., data = train)  
## Variables actually used in tree construction:  
## [1] "monthjul" "balance"   
## Number of terminal nodes: 3   
## Residual mean deviance: 0.1301 = 4376 / 33640   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.3050 -0.1237 -0.1237 0.0000 -0.1237 0.8763

###3.d Visualizing ROC Curve

Now, if I plot this data on a graph, I will get a ROC curve. The ROC curve is the graph plotted with TPR on y-axis and FPR on x-axis for all possible threshold. Both TPR and FPR vary from 0 to 1. Therefore, a good classifier will have an arc/ curve and will be further away from the random classifier line. To quantify a good classifier from a bad one using a ROC curve, is done by AUC (Area under Curve). From the graph it is quite clear that a good classifier will have AUC higher than a bad classifier as the area under curve will be higher for the former.

AUC is also visible making Conditional Inference Trees a slightly better than other classifiers Logistic Regression, and Decision Tree Classifier.

lr\_roc <- roc(test$loanyes ~ lr\_predict,plot=TRUE,print.auc=TRUE,col="green",lwd =4,legacy.axes=TRUE,main="ROC Curves")

## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

ct\_roc <- roc(test$loanyes ~ ct\_predict,plot=TRUE,print.auc=TRUE,col="blue",lwd = 4,print.auc.y=0.4,legacy.axes=TRUE,add = TRUE)

## Setting levels: control = 0, case = 1

## Warning in roc.default(response, predictors[, 1], ...): Deprecated use a matrix  
## as predictor. Unexpected results may be produced, please pass a numeric vector.

## Setting direction: controls < cases

tr\_roc <- roc(test$loanyes ~ tr\_predict,plot=TRUE,print.auc=TRUE,col="red",lwd = 4,print.auc.y=0.3,legacy.axes=TRUE,add = TRUE)

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

legend("bottomright",legend=c("Logistic Regression","CT", "Decision Tree"),col=c("green","blue", "red"),lwd=4)

