Classification

Matthew McCoy, Dmitrii Obideiko

2023-02-18

Load data and look at it's characteristics. There are 45k rows and 17 columns.

```
library(readr)

bank_full <- read.csv("bank-full.csv", sep = ";" , header = TRUE )
#bank_full <- bank_full[, -c(9 , 10, 11, 12, 13, 14, 15, 16)]

dim(bank_full)
```

```
## [1] 45211    17
```

```
head(bank_full)
```

```
##   age          job marital education default balance housing loan contact day
## 1  58   management married  tertiary      no    2143     yes   no unknown   5
## 2  44   technician  single secondary      no      29     yes   no unknown   5
## 3  33 entrepreneur married secondary      no       2     yes  yes unknown   5
## 4  47  blue-collar married   unknown      no    1506     yes   no unknown   5
## 5  33      unknown  single   unknown      no       1      no   no unknown   5
## 6  35   management married  tertiary      no     231     yes   no unknown   5
##   month duration campaign pdays previous poutcome  y
## 1   may      261        1    -1        0  unknown no
## 2   may      151        1    -1        0  unknown no
## 3   may       76        1    -1        0  unknown no
## 4   may       92        1    -1        0  unknown no
## 5   may      198        1    -1        0  unknown no
## 6   may      139        1    -1        0  unknown no
```

```
str(bank_full)
```

```
## 'data.frame':    45211 obs. of  17 variables:
##  $ age      : int  58 44 33 47 33 35 28 42 58 43 ...
##  $ job      : chr  "management" "technician" "entrepreneur" "blue-collar" ...
##  $ marital  : chr  "married" "single" "married" "married" ...
##  $ education: chr  "tertiary" "secondary" "secondary" "unknown" ...
##  $ default  : chr  "no" "no" "no" "no" ...
##  $ balance  : int  2143 29 2 1506 1 231 447 2 121 593 ...
##  $ housing  : chr  "yes" "yes" "yes" "yes" ...
##  $ loan     : chr  "no" "no" "yes" "no" ...
##  $ contact  : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ day      : int  5 5 5 5 5 5 5 5 5 5 ...
##  $ month    : chr  "may" "may" "may" "may" ...
##  $ duration : int  261 151 76 92 198 139 217 380 50 55 ...
##  $ campaign : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ pdays    : int  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
##  $ previous : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ poutcome : chr  "unknown" "unknown" "unknown" "unknown" ...
##  $ y        : chr  "no" "no" "no" "no" ...
```

Change the char columns to work with our models.

```
bank_full$y <- as.factor(bank_full$y)
bank_full$job <- as.factor(bank_full$job)
bank_full$marital <- as.factor(bank_full$marital)
bank_full$education <- as.factor(bank_full$education)
bank_full$default <- as.factor(bank_full$default)
bank_full$housing <- as.factor(bank_full$housing)
bank_full$loan <- as.factor(bank_full$loan)
bank_full$contact <- as.factor(bank_full$contact)
bank_full$month <- as.factor(bank_full$month)
bank_full$poutcome <- as.factor(bank_full$poutcome)
```

Set seed and separate our data set into 80% train data and 20% test data.

```
set.seed(1234)
i <- sample(1:nrow(bank_full), 0.80*nrow(bank_full), replace=FALSE)
train <- bank_full[i,]
test <- bank_full[-i,]
summary(train)
```

```
##       age                 job           marital          education    
##  Min.   :18.00   blue-collar:7765   divorced: 4169   primary  : 5436  
##  1st Qu.:33.00   management :7615   married :21776   secondary:18609  
##  Median :39.00   technician :6085   single  :10223   tertiary :10660  
##  Mean   :40.92   admin.     :4167                    unknown  : 1463  
##  3rd Qu.:48.00   services   :3306                                     
##  Max.   :95.00   retired    :1818                                     
##                  (Other)    :5412                                     
##  default        balance       housing      loan            contact     
##  no :35512   Min.   : -6847   no :16107   no :30377   cellular :23497  
##  yes:  656   1st Qu.:    74   yes:20061   yes: 5791   telephone: 2315  
##              Median :   451                           unknown  :10356  
##              Mean   :  1358                                            
##              3rd Qu.:  1428                                            
##              Max.   :102127                                            
##                                                                        
##       day           month          duration       campaign     
##  Min.   : 1.0   may    :10976   Min.   :   0   Min.   : 1.000  
##  1st Qu.: 8.0   jul    : 5561   1st Qu.: 103   1st Qu.: 1.000  
##  Median :16.0   aug    : 4998   Median : 181   Median : 2.000  
##  Mean   :15.8   jun    : 4257   Mean   : 258   Mean   : 2.755  
##  3rd Qu.:21.0   nov    : 3151   3rd Qu.: 319   3rd Qu.: 3.000  
##  Max.   :31.0   apr    : 2354   Max.   :4918   Max.   :63.000  
##                 (Other): 4871                                  
##      pdays           previous          poutcome       y        
##  Min.   : -1.00   Min.   : 0.0000   failure: 3950   no :31918  
##  1st Qu.: -1.00   1st Qu.: 0.0000   other  : 1452   yes: 4250  
##  Median : -1.00   Median : 0.0000   success: 1242              
##  Mean   : 40.35   Mean   : 0.5765   unknown:29524              
##  3rd Qu.: -1.00   3rd Qu.: 0.0000                              
##  Max.   :854.00   Max.   :58.0000                              
## 
```

The first few rows of the data.

```
head(train)
```

```
##       age         job  marital education default balance housing loan  contact
## 40784  45  management  married  tertiary      no    3857     yes   no cellular
## 40854  60 blue-collar  married   primary      no     631      no   no cellular
## 41964  77  management  married   unknown      no    1780     yes   no cellular
## 15241  50  technician divorced secondary      no    8016      no   no cellular
## 33702  37      admin.  married secondary      no     749      no   no cellular
## 35716  48 blue-collar  married secondary      no    1794     yes  yes cellular
##       day month duration campaign pdays previous poutcome   y
## 40784  11   aug      425        2   190        1  failure  no
## 40854  12   aug      429        2    -1        0  unknown yes
## 41964  23   oct      221        2   183        3  success yes
## 15241  17   jul      903        4    -1        0  unknown  no
## 33702  21   apr      219        1    -1        0  unknown  no
## 35716   8   may       97        1   343        2  failure  no
```

Dim on the training data shows us that it still contains 17 columns but now only 36168 rows, 80% of our original amount.

```
dim(train)
```

```
## [1] 36168    17
```

The structure of our training data set shows how the char variables have been changed to factors and how many levels each has. For example our job column has 12 different identifiable jobs separated into 12 levels.

```
str(train)
```

```
## 'data.frame':    36168 obs. of  17 variables:
##  $ age      : int  45 60 77 50 37 48 56 50 37 32 ...
##  $ job      : Factor w/ 12 levels "admin.","blue-collar",..: 5 2 5 10 1 2 5 10 5 2 ...
##  $ marital  : Factor w/ 3 levels "divorced","married",..: 2 2 2 1 2 2 1 3 2 3 ...
##  $ education: Factor w/ 4 levels "primary","secondary",..: 3 1 4 2 2 2 3 2 3 2 ...
##  $ default  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ balance  : int  3857 631 1780 8016 749 1794 -59 246 578 1940 ...
##  $ housing  : Factor w/ 2 levels "no","yes": 2 1 2 1 1 2 1 2 2 2 ...
##  $ loan     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 2 2 1 1 2 ...
##  $ contact  : Factor w/ 3 levels "cellular","telephone",..: 1 1 1 1 1 1 1 1 1 3 ...
##  $ day      : int  11 12 23 17 21 8 28 17 8 13 ...
##  $ month    : Factor w/ 12 levels "apr","aug","dec",..: 2 2 11 6 1 9 6 6 2 9 ...
##  $ duration : int  425 429 221 903 219 97 127 174 401 299 ...
##  $ campaign : int  2 2 2 4 1 1 6 2 4 3 ...
##  $ pdays    : int  190 -1 183 -1 -1 343 -1 -1 -1 -1 ...
##  $ previous : int  1 0 3 0 0 2 0 0 0 0 ...
##  $ poutcome : Factor w/ 4 levels "failure","other",..: 1 4 3 4 4 1 4 4 4 4 ...
##  $ y        : Factor w/ 2 levels "no","yes": 1 2 2 1 1 1 1 1 1 1 ...
```

The skim function is an alternative to summary() , quickly providing a broad overview of a data frame. It handles data of all types, dispatching a different set of summary functions based on the types of columns in the data frame.

```
library(skimr)
skim(train)
```

Data summary

Name	train
Number of rows	36168
Number of columns	17
_______________________	
Column type frequency:	

factor	10
numeric	7
________________________	
Group variables	None

Variable type: factor

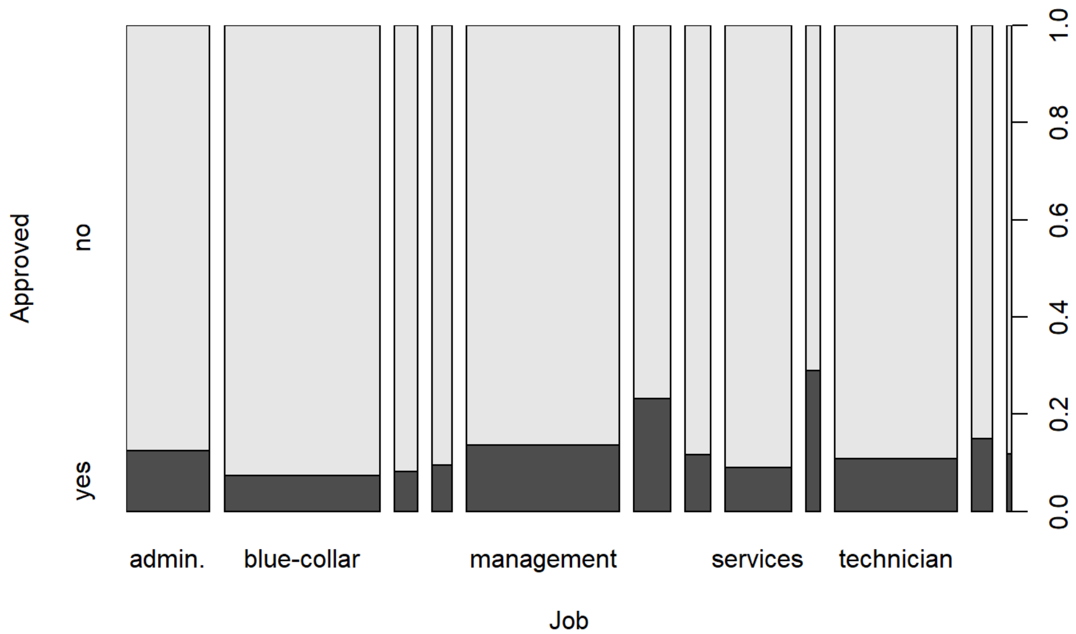
skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
job	0	1	FALSE	12	blu: 7765, man: 7615, tec: 6085, adm: 4167
marital	0	1	FALSE	3	mar: 21776, sin: 10223, div: 4169
education	0	1	FALSE	4	sec: 18609, ter: 10660, pri: 5436, unk: 1463
default	0	1	FALSE	2	no: 35512, yes: 656
housing	0	1	FALSE	2	yes: 20061, no: 16107
loan	0	1	FALSE	2	no: 30377, yes: 5791
contact	0	1	FALSE	3	cel: 23497, unk: 10356, tel: 2315
month	0	1	FALSE	12	may: 10976, jul: 5561, aug: 4998, jun: 4257
poutcome	0	1	FALSE	4	unk: 29524, fai: 3950, oth: 1452, suc: 1242
y	0	1	FALSE	2	no: 31918, yes: 4250

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	40.92	10.62	18	33	39	48	95	
balance	0	1	1357.94	3012.30	-6847	74	451	1428	102127	
day	0	1	15.80	8.33	1	8	16	21	31	
duration	0	1	258.01	256.88	0	103	181	319	4918	
campaign	0	1	2.76	3.08	1	1	2	3	63	
pdays	0	1	40.35	100.21	-1	-1	-1	-1	854	
previous	0	1	0.58	1.92	0	0	0	0	58	

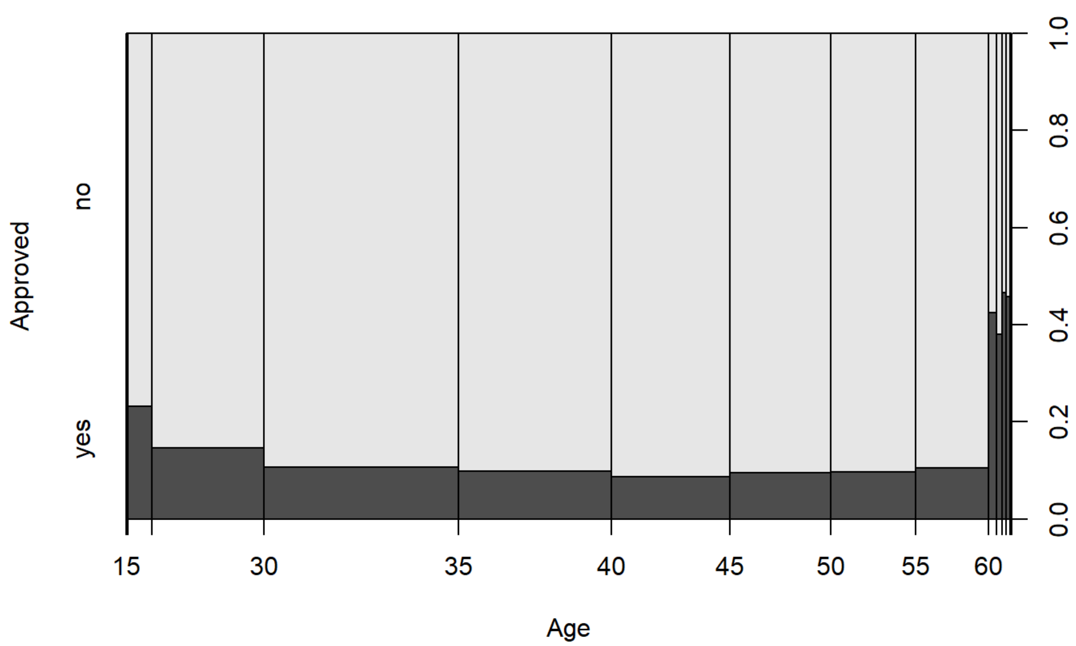
Our first graph plots the amount of approvals were given to each job. The width of the columns shows the amount of those jobs in the data set in comparison to the other jobs in the set.

```
plot(train$y~train$job , xlab="Job", ylab="Approved")
```



Our second graph shows the amount of approvals based on age. We can see that more approvals come to the people in the data set 60 years of age and older.

```
plot(train$y~train$age , xlab="Age", ylab="Approved")
```



#Building Log Reg Model

The summary of our model shows that many factors of each of our variables is providing significant data in predicting whether the person will be approved or not. Our residual deviance is lower than our null deviance by nearly 40% showing that this may be a good model for our data set. However our AIC is high and may show that this is not a good fit for our data.

```
glm1 <- glm(y~., data = train, family = binomial)
summary(glm1)
```

```
##
## Call:
## glm(formula = y ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.7458  -0.3748  -0.2536  -0.1491   3.4437
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -2.714e+00  2.061e-01 -13.172  < 2e-16 ***
## age                 1.598e-04  2.458e-03   0.065 0.948159
## jobblue-collar     -3.274e-01  8.027e-02  -4.079 4.52e-05 ***
## jobentrepreneur    -4.196e-01  1.422e-01  -2.951 0.003166 **
## jobhousemaid       -4.260e-01  1.481e-01  -2.876 0.004023 **
## jobmanagement      -2.052e-01  8.110e-02  -2.530 0.011414 *
## jobretired          2.695e-01  1.077e-01   2.502 0.012361 *
## jobself-employed   -3.272e-01  1.243e-01  -2.633 0.008475 **
## jobservices        -2.938e-01  9.373e-02  -3.135 0.001721 **
## jobstudent          4.081e-01  1.220e-01   3.345 0.000823 ***
## jobtechnician      -2.420e-01  7.658e-02  -3.160 0.001580 **
## jobunemployed      -2.372e-01  1.254e-01  -1.891 0.058560 .
## jobunknown         -4.276e-01  2.639e-01  -1.620 0.105214
## maritalmarried     -2.014e-01  6.539e-02  -3.080 0.002067 **
## maritalsingle       4.573e-02  7.477e-02   0.612 0.540755
## educationsecondary  2.512e-01  7.280e-02   3.450 0.000560 ***
## educationtertiary   4.140e-01  8.457e-02   4.895 9.81e-07 ***
## educationunknown    3.401e-01  1.163e-01   2.924 0.003455 **
## defaultyes          6.891e-02  1.750e-01   0.394 0.693843
## balance             1.179e-05  5.829e-06   2.022 0.043176 *
## housingyes         -6.552e-01  4.890e-02 -13.400  < 2e-16 ***
## loanyes            -4.014e-01  6.649e-02  -6.037 1.57e-09 ***
## contacttelephone   -1.977e-01  8.460e-02  -2.336 0.019469 *
## contactunknown     -1.668e+00  8.154e-02 -20.458  < 2e-16 ***
## day                 1.064e-02  2.786e-03   3.817 0.000135 ***
## monthaug           -6.444e-01  8.782e-02  -7.338 2.17e-13 ***
## monthdec            6.883e-01  1.956e-01   3.519 0.000433 ***
## monthfeb           -1.390e-01  1.003e-01  -1.385 0.166038
## monthjan           -1.255e+00  1.370e-01  -9.160  < 2e-16 ***
## monthjul           -7.983e-01  8.642e-02  -9.238  < 2e-16 ***
## monthjun            5.654e-01  1.044e-01   5.418 6.04e-08 ***
## monthmar            1.626e+00  1.350e-01  12.038  < 2e-16 ***
## monthmay           -3.582e-01  8.093e-02  -4.426 9.60e-06 ***
## monthnov           -8.033e-01  9.420e-02  -8.527  < 2e-16 ***
## monthoct            9.877e-01  1.217e-01   8.115 4.87e-16 ***
## monthsep            9.919e-01  1.321e-01   7.506 6.10e-14 ***
## duration            4.219e-03  7.268e-05  58.050  < 2e-16 ***
## campaign           -1.034e-01  1.178e-02  -8.783  < 2e-16 ***
## pdays               1.322e-04  3.380e-04   0.391 0.695628
## previous            3.284e-02  1.066e-02   3.080 0.002070 **
## poutcomeother       1.557e-01  1.026e-01   1.517 0.129303
## poutcomesuccess     2.307e+00  9.173e-02  25.145  < 2e-16 ***
## poutcomeunknown     6.703e-02  1.078e-01   0.622 0.534080
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 26180  on 36167  degrees of freedom
## Residual deviance: 17282  on 36125  degrees of freedom
## AIC: 17368
##
## Number of Fisher Scoring iterations: 6
```

We evaluate our model to see if we can use it to predict our test data. We are using all the columns in the prediction and see that our model was able to predict the outcome of our test data with an over 90% accuracy. While this may seem good we need to explore our data more to find out if this is truly the case.

```
probs <- predict(glm1,  newdata=test, type="response")
pred <- ifelse(probs> 0.5, "yes", "no")
acc <- mean(pred == test$y)
print(paste("accuracy = ", acc))
```

```
## [1] "accuracy =  0.902687161340263"
```

Our table shows a matrix that will be the same as our confusion matrix but with less information. Let’s try the confusion matrix to be sure.

```
table(pred, test$y)
```

```
##
## pred    no  yes
##   no  7796  672
##   yes  208  367
```

Yes our confusion matrix is equal to the table we created with the table() function. The model predicted no more than it predicted yes. It predicted no 7796 times and was correct and 672 times it predicted no and was wrong. We can see here that our data is heavily slated to “no”s. The model also predicted yes correctly 367 times and incorrectly 208 times. This gives us some confidence that it is correct on both no and yes more times than it is wrong. Our 90+% accuracy looks like it might be that high due to the amount of times no is the correct answer though.

```
library(caret)
```

```
## Loading required package: ggplot2
```

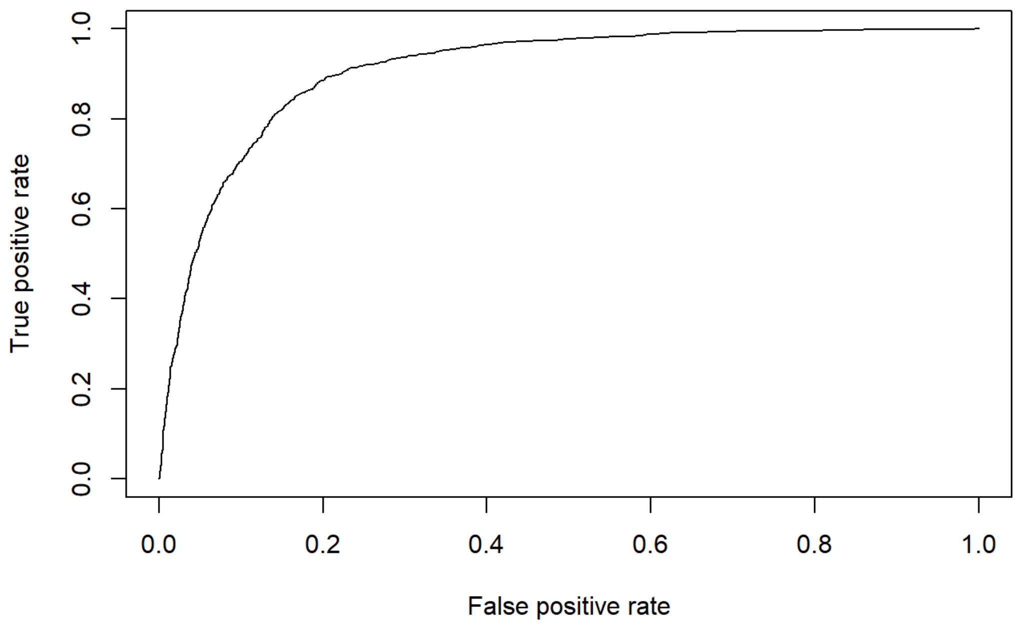
```
## Loading required package: lattice
```

```
confusionMatrix(as.factor(pred), reference = test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  7796  672
##        yes  208  367
##
##                Accuracy : 0.9027
##                  95% CI : (0.8964, 0.9087)
##     No Information Rate : 0.8851
##     P-Value [Acc > NIR] : 4.487e-08
##
##                   Kappa : 0.4062
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9740
##             Specificity : 0.3532
##          Pos Pred Value : 0.9206
##          Neg Pred Value : 0.6383
##              Prevalence : 0.8851
##          Detection Rate : 0.8621
##    Detection Prevalence : 0.9364
##       Balanced Accuracy : 0.6636
##
##        'Positive' Class : no
##
```

Let’s build our ROCR. The ROC is a curve that plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. Also we like to see the ROC shoot up rather quickly. As we see in the graph we do have a quick curve to 1.

```
library(ROCR)
p <- predict(glm1, newdata = test, type = "response")
pr <- prediction(p, test$y)
prf <- performance (pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```



Let’s check the area under the curve. We will be looking for something close to accuracy to confirm the results. The AUC is the area under the ROC curve. A good AUC is close to 1 than 0.5.

```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.9094118
```

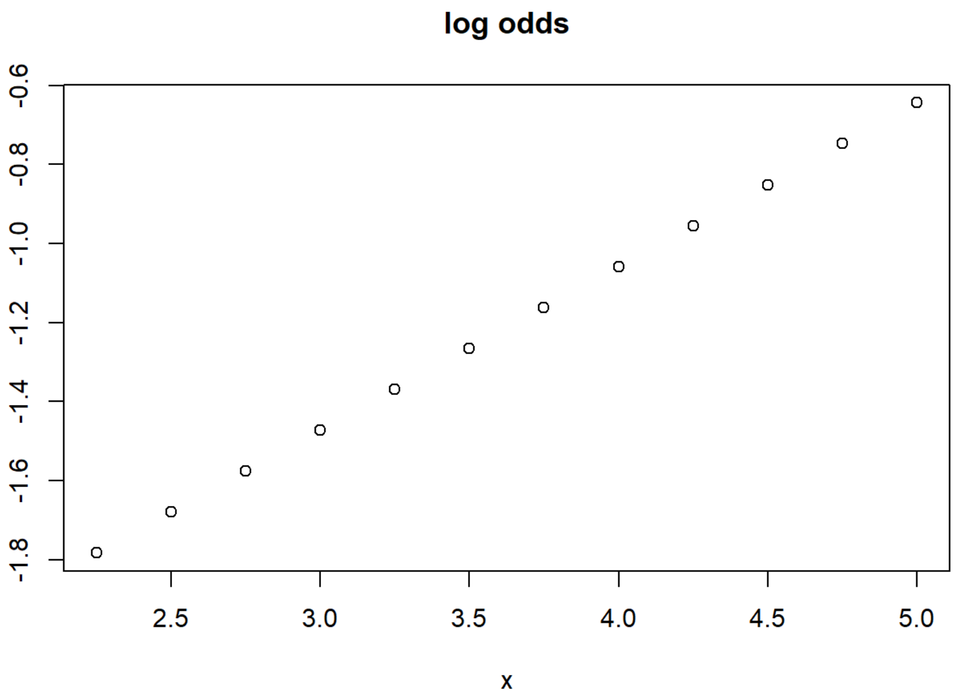
Let’s check the log odds our model. Log odds are a bit more difficult with the amount of variables we have being used to predict our target. When we build and plot the model we see a visualization of our log odds as a line. A separate graph plots these as possibilities. This line seems to sag a bit in the middle showing a slower rate to the top.

```
y_prime <- glm1$coefficients[17]
intercept <- glm1$coefficients[1]

log_odds <- function(x, y_prime, intercept){
  intercept + y_prime * x
}

x <- seq(from=2.25, to=5.0, by=0.25)
y <- log_odds(x, y_prime, intercept)
par(mfrow=c(~.))
```

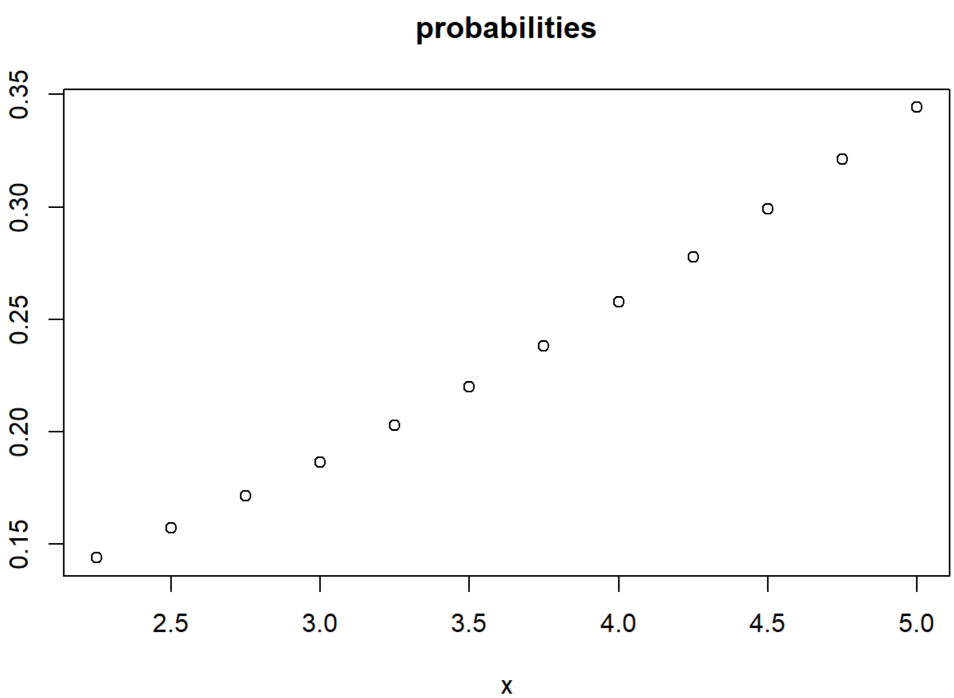
```
## Warning in par(mfrow = c(~.)): argument 1 does not name a graphical parameter
```

```
## [[1]]
## NULL
```

```
plot(x,y, main="log odds", ylab="")
```




```
prob <- exp(y) / (1+ exp(y))

plot(x, prob, main="probabilities", ylab="")
```



Let’s build our naive Bayes model. The naiveBayes function is used to fit a naive Bayes classifier to our bank-full data set. Naive Bayes is a probabilistic algorithm that is commonly used for classification tasks in machine learning. When you run naiveBayes in R Studio, it produces a trained classifier object that can be used to make predictions on new data. We can see that some of our predictors almost always lead to a particular value for our target column. One example is “default”, it states that a person has defaulted on a loan in the past. 98+% of people who defaulted on a loan in the past will be declined for the loan in the data set.

```
library(e1071)
nb1 <- naiveBayes(y~., data=train)
nb1
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##        no       yes
## 0.8824928 0.1175072
##
## Conditional probabilities:
##      age
## Y         [,1]     [,2]
##   no  40.81647 10.16867
##   yes 41.69859 13.54081
##
##      job
## Y          admin. blue-collar entrepreneur   housemaid  management     retired
##   no  0.114261545 0.225327401  0.033147440 0.028071934 0.205902625 0.043799737
##   yes 0.122352941 0.134823529  0.022352941 0.022117647 0.245411765 0.098823529
##      job
## Y     self-employed    services     student  technician  unemployed     unknown
##   no    0.035152578 0.094210164 0.016385738 0.169872799 0.027507989 0.006360048
##   yes   0.034823529 0.070352941 0.050117647 0.156000000 0.036470588 0.006352941
##
##      marital
## Y      divorced   married    single
##   no  0.1146375 0.6129457 0.2724168
##   yes 0.1200000 0.5204706 0.3595294
##
##      education
## Y        primary  secondary   tertiary    unknown
##   no  0.15583683 0.52049001 0.28419700 0.03947616
##   yes 0.10870588 0.46964706 0.37388235 0.04776471
##
##      default
## Y             no        yes
##   no  0.98088853 0.01911147
##   yes 0.98917647 0.01082353
##
##      balance
## Y         [,1]     [,2]
##   no  1299.318 2916.282
##   yes 1798.174 3623.894
##
##      housing
## Y            no       yes
##   no  0.4203584 0.5796416
##   yes 0.6329412 0.3670588
##
##      loan
## Y             no        yes
##   no  0.83100445 0.16899555
##   yes 0.90658824 0.09341176
##
##      contact
## Y       cellular  telephone    unknown
##   no  0.62601040 0.06297387 0.31101573
##   yes 0.82729412 0.07176471 0.10094118
##
##      day
## Y         [,1]     [,2]
##   no  15.89426 8.296194
##   yes 15.12306 8.524005
```

```
##
##      month
## Y             apr         aug         dec         feb         jan         jul
##   no  0.059214236 0.139325772 0.003007707 0.056206529 0.031800238 0.158186603
##   yes 0.109176471 0.129647059 0.019058824 0.081647059 0.026117647 0.120470588
##      month
## Y             jun         mar         may         nov         oct         sep
##   no  0.119211730 0.005796103 0.320759446 0.088727364 0.009837709 0.007926562
##   yes 0.106352941 0.044941176 0.173647059 0.075058824 0.061882353 0.052000000
##
##      duration
## Y         [,1]     [,2]
##   no  221.1191 206.7678
##   yes 535.0727 391.8009
##
##      campaign
## Y         [,1]     [,2]
##   no  2.840623 3.198375
##   yes 2.112941 1.807504
##
##      pdays
## Y         [,1]     [,2]
##   no  36.53757  96.6514
##   yes 68.96494 119.9090
##
##      previous
## Y          [,1]     [,2]
##   no  0.4979009 1.794267
##   yes 1.1665882 2.588425
##
##      poutcome
## Y        failure      other    success    unknown
##   no  0.10824613 0.03816029 0.01387932 0.83971427
##   yes 0.11647059 0.05505882 0.18800000 0.64047059
```

Let's use the predict function to make predictions on our model. The predict function will take a trained classifier object and our test data as input, and produces a vector of predicted class labels. We can see at 87.3% our naive bayes model did not predict the outcome as well as our logistic regression model at 90+%.

```
p1 <- predict(nb1,  newdata=test, type="class")
table(p1, test$y)
```

```
##
## p1      no  yes
##   no  7342  483
##   yes  662  556
```

```
mean(p1==test$y)
```

```
## [1] 0.8733827
```

```
mean(p1==test$y)
```

```
## [1] 0.8733827
```

Our confusion matrix on the naive Bayes model furthers this observation.

```
confusionMatrix(as.factor(p1), reference = test$y)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   no  yes
##        no  7342  483
##        yes  662  556
##                                           
##                Accuracy : 0.8734          
##                  95% CI : (0.8664, 0.8802)
##     No Information Rate : 0.8851          
##     P-Value [Acc > NIR] : 0.9997          
##                                           
##                   Kappa : 0.4209          
##                                           
##  Mcnemar's Test P-Value : 1.438e-07       
##                                           
##             Sensitivity : 0.9173          
##             Specificity : 0.5351          
##          Pos Pred Value : 0.9383          
##          Neg Pred Value : 0.4565          
##              Prevalence : 0.8851          
##          Detection Rate : 0.8119          
##    Detection Prevalence : 0.8653          
##       Balanced Accuracy : 0.7262          
##                                           
##        'Positive' Class : no              
## 
```