PREDICT IF TRANSACTION IS BY LOYAL CUSTOMERS IN MACHINE LEARNING USING R

BY BRIAN ESTVANDER DATE: 10 MAY 2024

```
# Load libraries and read in data----

library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'tibble' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3

## Warning: package 'forcats' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
df1 <- read_rds('logistic1.rds')
```

```
# Build a Matrix Function to be used later----

my_confusion_matrix <- function(cf_table) {
  true_positive <- cf_table[4]
  true_negative <- cf_table[1]
  false_positive <- cf_table[2]
  false_negative <- cf_table[3]
  accuracy <- (true_positive + true_negative) / (true_positive + true_negative + false_positive + false_
  sensitivity_recall <- true_positive / (true_positive + false_negative)
  specificity_selectivity <- true_negative / (true_negative + false_positive)
  precision <- true_positive / (true_positive + false_positive)
  neg_pred_value <- true_negative/(true_negative + false_negative)
  print(cf_table)
  my_list <- list(sprintf("%1.0f = True Positive (TP), Hit", true_positive),
                  sprintf("%1.0f = True Negative (TN), Rejection", true_negative),
                  sprintf("%1.0f = False Positive (FP), Type 1 Error", false_positive),
                  sprintf("%1.0f = False Negative (FN), Type 2 Error", false_negative),
                  sprintf("%1.4f = Accuracy (TP+TN/(TP+TN+FP+FN))", accuracy),
                  sprintf("%1.4f = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives
                  sprintf("%1.4f = Specificity, Selectivity, True Negative Rate (How many negatives did
                  sprintf("%1.4f = Precision, Positive Predictive Value (How good are the model's positi
```

```
                   sprintf("%1.4f = Negative Predictive Value (How good are the model's negative predicti
  )
  return(my_list)
}
```

Start with linear regression (sigmoidial) as first classification model then try RELU—-

```
# View the data----
slice_sample(df1, n=10)
```

```
##           loyalty                category quarter    state
## 52401       loyal              Cigarettes       2 Colorado
## 3624161 not loyal         Smokeless (951)       2     Iowa
## 879883  not loyal            Salty Snacks       2 Oklahoma
## 846524  not loyal                   Pizza       4 Oklahoma
## 661161  not loyal    Breakfast Sandwiches       3 Missouri
## 2795431 not loyal Cold Dispensed Beverage       2     Iowa
## 158135      loyal               Candy/Gum       2 Arkansas
## 711594  not loyal                 Lottery       1 Missouri
## 53409       loyal              Cigarettes       1 Missouri
## 1385414 not loyal                 Lottery       4     Iowa
```

Baseline Occurance of 'loyality'

```
loyal_table <- table(df1$loyalty)
print(loyal_table)
```

```
##
## not loyal     loyal
##    520000    519744
```

```
print(loyal_table[2]/(loyal_table[1]+loyal_table[2]))
```

```
##     loyal
## 0.4998769
```

Use contrast() to check order of 'loyality'—-

```
contrasts(df1$loyalty)
```

```
##           loyal
## not loyal     0
## loyal         1
```

Order is good so now split the data to get ~ 75% as training data—-

Also, load caret library—-

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(77)
partition <- caret::createDataPartition(y=df1$loyalty, p=.75, list=FALSE)
data_train <- df1[partition,]
data_test <- df1[-partition,]
print(nrow(data_train)/(nrow(data_test)+nrow(data_train)))
```

```
## [1] 0.75
```

Train the model

```
model_train <- glm(loyalty ~ category, family=binomial, data=data_train)
summary(model_train)
```

```
##
## Call:
## glm(formula = loyalty ~ category, family = binomial, data = data_train)
##
## Coefficients:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        0.30575    0.01539  19.870  < 2e-16 ***
## categoryBeer                      -1.15590    0.02101 -55.009  < 2e-16 ***
## categoryBread & Cakes             -0.11000    0.02173  -5.063 4.14e-07 ***
## categoryBreakfast Sandwiches       0.22179    0.02156  10.288  < 2e-16 ***
## categoryCandy/Gum                 -0.33491    0.01816 -18.442  < 2e-16 ***
## categoryChips                     -0.20758    0.02392  -8.677  < 2e-16 ***
## categoryCigarettes                -0.48665    0.01769 -27.517  < 2e-16 ***
## categoryCigars                    -0.95134    0.02544 -37.395  < 2e-16 ***
## categoryCold Dispensed Beverage    0.46045    0.01706  26.991  < 2e-16 ***
## categoryEnergy                     0.02773    0.01797   1.544  0.12269
## categoryFuel                      -0.63600    0.01622 -39.211  < 2e-16 ***
## categoryHot Dispensed Beverage    -0.01130    0.01866  -0.605  0.54487
## categoryHot Sandwiches & Chicken -0.06145    0.02298  -2.674  0.00749 **
## categoryJuice/tonics              -0.40630    0.01752 -23.196  < 2e-16 ***
## categoryLottery                   -0.91991    0.01865 -49.313  < 2e-16 ***
## categoryPizza                      0.10643    0.02130   4.996 5.84e-07 ***
## categoryPop (587)                 -0.31478    0.01706 -18.448  < 2e-16 ***
## categoryRoller Grill              -0.18406    0.02183  -8.430  < 2e-16 ***
## categorySalty Snacks              -0.28467    0.01997 -14.252  < 2e-16 ***
```

```
## categorySmokeless (951)          -0.67115    0.02315 -28.989  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1081043  on 779807  degrees of freedom
## Residual deviance: 1050668  on 779788  degrees of freedom
## AIC: 1050708
##
## Number of Fisher Scoring iterations: 4
```

The intercept lists the effect of bakery items on whether a transaction is from a loyalty customer or not. The coefficient is positive, which means that selling a bakery item increases the likelihood that a transaction is a loyalty transaction. More precisely, purchasing this item increases the log odds of a loyalty purchase happening by 0.30575.We know now that several categories, such as fountain sodas, bakery items, and breakfast sandwiches help increase the chance a customer will be a loyalty customer. The company could promote these products to try to draw in more loyalty customers——

Examine accuracy with the train model first to see loyal vs non-loyal customers within regression model—-

```
# Predict the probabilities for each row, using a small sample of the first 10 rows to get a visual ide

predict_train <- predict(model_train, newdata=data_train, type='response')
print(summary(predict_train))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2994  0.4182  0.4927  0.4999  0.5731  0.6827
```

```
data_train$prediction <- predict_train
head(data_train, n=10)
```

```
##    loyalty      category quarter    state prediction
## 1    loyal Bread & Cakes       3 Colorado  0.5487819
## 2    loyal      Candy/Gum       1  Wyoming  0.4927105
## 3    loyal      Pop (587)       3  Wyoming  0.4977430
## 4    loyal      Pop (587)       4 Colorado  0.4977430
## 5    loyal          Chips       1 Oklahoma  0.5245229
## 6    loyal   Juice/tonics       2 Colorado  0.4748839
## 7    loyal   Juice/tonics       3  Wyoming  0.4748839
## 9    loyal           Beer       3  Alabama  0.2994030
## 10   loyal          Chips       2 Oklahoma  0.5245229
## 11   loyal     Cigarettes       1 Nebraska  0.4548983
```

Determine accuracy of model using loyality and non-loyalty data—-

```r
# Put prediction on left and truth on top
table1 <- table(predict_train>0.5, data_train$loyalty)
my_confusion_matrix(table1)
```

```
##
##         not loyal  loyal
##   FALSE    265829 204677
##   TRUE     124171 185131
```

```
## [[1]]
## [1] "185131 = True Positive (TP), Hit"
##
## [[2]]
## [1] "265829 = True Negative (TN), Rejection"
##
## [[3]]
## [1] "124171 = False Positive (FP), Type 1 Error"
##
## [[4]]
## [1] "204677 = False Negative (FN), Type 2 Error"
##
## [[5]]
## [1] "0.5783 = Accuracy (TP+TN/(TP+TN+FP+FN))"
##
## [[6]]
## [1] "0.4749 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model ge
##
## [[7]]
## [1] "0.6816 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get rig
##
## [[8]]
## [1] "0.5985 = Precision, Positive Predictive Value (How good are the model's positive predictions? T
##
## [[9]]
## [1] "0.5650 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN)"
```

In general, the model predicts better than 50% without implementing a model, and is way better for predicting non-loyal vs loyal customers at 58%—-

Train on the testing data by replacing data_train$loyality with data_test$loyality—-

```r
predict_test <- predict(model_train, newdata=data_test, type='response')
print(summary(predict_test))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2994  0.4182  0.4927  0.5003  0.5731  0.6827
```

```r
data_test$prediction <- predict_test
head(data_test, n=10)
```

```
##    loyalty                category quarter    state prediction
## 8    loyal           Bread & Cakes       2  Alabama  0.5487819
## 13   loyal               Pop (587)       4     Iowa  0.4977430
## 16   loyal            Salty Snacks       3 Colorado  0.5052708
## 20   loyal Cold Dispensed Beverage       1 Missouri  0.6826999
## 21   loyal                    Fuel       2  Alabama  0.4181812
## 24   loyal         Smokeless (951)       1  Wyoming  0.4096534
## 26   loyal            Juice/tonics       4  Wyoming  0.4748839
## 34   loyal               Candy/Gum       1 Oklahoma  0.4927105
## 36   loyal            Juice/tonics       1  Alabama  0.4748839
## 37   loyal                  Energy       4 Oklahoma  0.5826080
```

```r
table2 <- table(predict_test>.5, data_test$loyalty) #prediction on left and truth on top
my_confusion_matrix(table2)
```

```
##
##         not loyal loyal
##   FALSE     88248 67847
##   TRUE      41752 62089
```

```
## [[1]]
## [1] "62089 = True Positive (TP), Hit"
##
## [[2]]
## [1] "88248 = True Negative (TN), Rejection"
##
## [[3]]
## [1] "41752 = False Positive (FP), Type 1 Error"
##
## [[4]]
## [1] "67847 = False Negative (FN), Type 2 Error"
##
## [[5]]
## [1] "0.5784 = Accuracy (TP+TN/(TP+TN+FP+FN))"
##
## [[6]]
## [1] "0.4778 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model get
##
## [[7]]
## [1] "0.6788 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get rig
##
## [[8]]
## [1] "0.5979 = Precision, Positive Predictive Value (How good are the model's positive predictions? T
##
## [[9]]
## [1] "0.5653 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN)"
```

Training on the test_data is comparable to train_data in accuracy, sensitivity, and precision—-

Improve sensitivity by performing a multivariate regression to include seasons—-

```
model_train <- glm(loyalty ~ category + factor(quarter) + state, family=binomial, data=data_train)
summary(model_train)
```

```
##
## Call:
## glm(formula = loyalty ~ category + factor(quarter) + state, family = binomial,
##     data = data_train)
##
## Coefficients:
##                                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)                          0.254291   0.016633  15.289  < 2e-16 ***
## categoryBeer                        -1.096064   0.021205 -51.689  < 2e-16 ***
## categoryBread & Cakes               -0.114865   0.021934  -5.237 1.63e-07 ***
## categoryBreakfast Sandwiches         0.220034   0.021724  10.129  < 2e-16 ***
## categoryCandy/Gum                   -0.341969   0.018316 -18.670  < 2e-16 ***
## categoryChips                       -0.204701   0.024131  -8.483  < 2e-16 ***
## categoryCigarettes                  -0.511199   0.017849 -28.641  < 2e-16 ***
## categoryCigars                      -0.966302   0.025690 -37.614  < 2e-16 ***
## categoryCold Dispensed Beverage      0.447609   0.017227  25.982  < 2e-16 ***
## categoryEnergy                       0.015697   0.018120   0.866  0.38633
## categoryFuel                        -0.638023   0.016359 -39.002  < 2e-16 ***
## categoryHot Dispensed Beverage      -0.009300   0.018821  -0.494  0.62122
## categoryHot Sandwiches & Chicken -0.059721   0.023156  -2.579  0.00991 **
## categoryJuice/tonics                -0.433122   0.017665 -24.519  < 2e-16 ***
## categoryLottery                     -0.991808   0.018851 -52.612  < 2e-16 ***
## categoryPizza                        0.096907   0.021468   4.514 6.36e-06 ***
## categoryPop (587)                   -0.279395   0.017215 -16.229  < 2e-16 ***
## categoryRoller Grill                -0.192345   0.022021  -8.735  < 2e-16 ***
## categorySalty Snacks                -0.281628   0.020151 -13.976  < 2e-16 ***
## categorySmokeless (951)             -0.691339   0.023350 -29.608  < 2e-16 ***
## factor(quarter)2                     0.194020   0.006887  28.170  < 2e-16 ***
## factor(quarter)3                     0.216750   0.006653  32.580  < 2e-16 ***
## factor(quarter)4                     0.228426   0.006624  34.486  < 2e-16 ***
## stateArkansas                       -0.341681   0.008364 -40.851  < 2e-16 ***
## stateColorado                        0.116604   0.007972  14.627  < 2e-16 ***
## stateIowa                           -0.285021   0.007312 -38.980  < 2e-16 ***
## stateMinnesota                      -0.285307   0.030645  -9.310  < 2e-16 ***
## stateMissouri                        0.276047   0.008295  33.281  < 2e-16 ***
## stateNebraska                       -0.182782   0.010315 -17.720  < 2e-16 ***
## stateOklahoma                       -0.592161   0.009407 -62.947  < 2e-16 ***
## stateSouth Dakota                   -0.165378   0.021925  -7.543 4.60e-14 ***
## stateWyoming                         0.085997   0.013440   6.399 1.57e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1081043  on 779807  degrees of freedom
## Residual deviance: 1037616  on 779776  degrees of freedom
## AIC: 1037680
##
## Number of Fisher Scoring iterations: 4
```

```r
predict_test <- predict(model_train, newdata=data_test, type='response')
summary(predict_test)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1925  0.4148  0.4903  0.5000  0.5788  0.7697
```

```r
data_test$prediction <- predict_test
head(data_test, n=10)
```

```
##    loyalty                category quarter    state prediction
## 8    loyal            Bread & Cakes       2  Alabama  0.5825976
## 13   loyal                Pop (587)       4     Iowa  0.4795867
## 16   loyal             Salty Snacks       3 Colorado  0.5759126
## 20   loyal Cold Dispensed Beverage       1 Missouri  0.7267008
## 21   loyal                     Fuel       2  Alabama  0.4527136
## 24   loyal          Smokeless (951)       1  Wyoming  0.4131277
## 26   loyal             Juice/tonics       4  Wyoming  0.5338463
## 34   loyal                Candy/Gum       1 Oklahoma  0.3362973
## 36   loyal             Juice/tonics       1  Alabama  0.4554110
## 37   loyal                   Energy       4 Oklahoma  0.4765806
```

```r
table2 <- table(predict_test>.5, data_test$loyalty)
my_confusion_matrix(table2)
```

```
##
##         not loyal loyal
##   FALSE     80501 55670
##   TRUE      49499 74266
```

```
## [[1]]
## [1] "74266 = True Positive (TP), Hit"
##
## [[2]]
## [1] "80501 = True Negative (TN), Rejection"
##
## [[3]]
## [1] "49499 = False Positive (FP), Type 1 Error"
##
## [[4]]
## [1] "55670 = False Negative (FN), Type 2 Error"
##
## [[5]]
## [1] "0.5954 = Accuracy (TP+TN/(TP+TN+FP+FN))"
##
## [[6]]
## [1] "0.5716 = Sensitivity, Recall, Hit Rate, True Positive Rate (How many positives did the model ge
##
## [[7]]
## [1] "0.6192 = Specificity, Selectivity, True Negative Rate (How many negatives did the model get rig
##
```

```
## [[8]]
## [1] "0.6001 = Precision, Positive Predictive Value (How good are the model's positive predictions? TI
##
## [[9]]
## [1] "0.5912 = Negative Predictive Value (How good are the model's negative predictions? TN/(TN+FN)"
```

Overall, the regression model is not that great. We can improve the iterations, increase the number data or try a different model like RELU. However, with the model, the accuracy is better than 50% at 60% and by including more variables into the regression model, the sensitivty was increased from 48% to 57%——