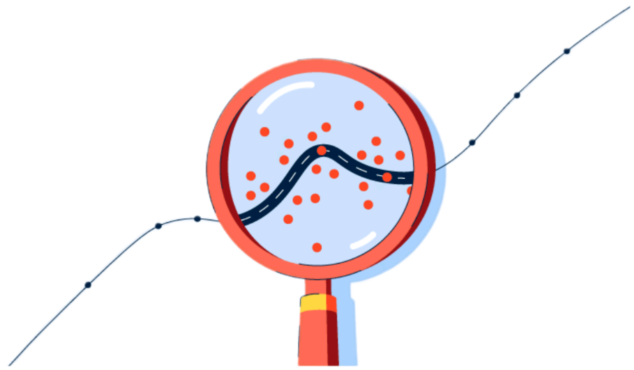


Binary Classification with a Bank Churn – Competition Notebook (Prediction Submission)

Sawandi Kirby

2024-01-25

Contents

1 Ask 5

1.1 About the Data 5

2 Prepare 5

2.1 Get Enviornment ready 5

2.2 Trim rows 9

2.3 Check for missing values 9

3 Analyze 10

3.1 Train Data 10

3.2 Train logistic regression model 13

3.3 Train XGBoost model 14

3.4 Create matrix with features and labels 15

4 Predictions **17**

4.1 XGBoost 17

4.2 Logistical Model 18

5 Share **19**

5.1 Visual 19

5.2 Archive 20

6 Logistic Regrssion Submission Entry Score = 0.80891 **20**

7 XGBoost Submission Entry Score = .88418 **21**

Ask

About the Data

The bank customer churn dataset is a commonly used dataset for predicting customer churn in the banking industry. It contains information on bank customers who either left the bank or continue to be a customer.

Our task is to predict whether a customer continues with their account or closes it (e.g., churns).

Prepare

Get Environment ready

Load required libraries

```
library(tidyverse)

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

library(ggplot2)
library(corrplot)

## corrplot 0.92 loaded

library(caret)

## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(glmnet)

## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8

library(pROC)

## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

library(xgboost)

##
## Attaching package: 'xgboost'
##
## The following object is masked from 'package:dplyr':
##
##     slice
```

Create function to apply trimws to character columns of a data frame

```r
trimws_df <- function(df) {
  char_cols <- sapply(df, is.character)
  df[char_cols] <- lapply(df[char_cols], trimws)
  return(df)
}
```

Load data

```r
test <- read_csv("playground-series-s4e1\\test.csv")

## Rows: 110023 Columns: 13
## ── Column specification
────────────────────────────────────────────────────────────────────────────────
────
## Delimiter: ","
## chr  (3): Surname, Geography, Gender
## dbl (10): id, CustomerId, CreditScore, Age, Tenure, Balance,
NumOfProducts, HasCrCa...
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

train <- read_csv("playground-series-s4e1\\train.csv")

## Rows: 165034 Columns: 14
## ── Column specification ──────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): Surname, Geography, Gender
## dbl (11): id, CustomerId, CreditScore, Age, Tenure, Balance, NumOfProducts, HasCrCa...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

sample_submission <- read_csv("playground-series-s4e1\\sample_submission.csv")

## Rows: 110023 Columns: 2
## ── Column specification ──────────────────────────────────────────────────────
## Delimiter: ","
## dbl (2): id, Exited
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Exploratory Data Analysis (EDA)

Get a glimpse of the datasets

```
head(train)

## # A tibble: 6 × 14
##      id CustomerId Surname        CreditScore Geography Gender   Age Tenure Balance
##   <dbl>      <dbl> <chr>                <dbl> <chr>     <chr>  <dbl>  <dbl>   <dbl>
## 1     0   15674932 Okwudilichukwu         668 France    Male      33 3      0
## 2     1   15749177 Okwudiliolisa          627 France    Male      33 1      0
## 3     2   15694510 Hsueh                  678 France    Male      40 10      0
## 4     3   15741417 Kao                    581 France    Male      34 2 148883.
```

```
## 5     4   15766172 Chiemenam              716 Spain     Male      33
5      0
## 6     5   15771669 Genovese               588 Germany   Male      36
4 131779.
## # ℹ 5 more variables: NumOfProducts <dbl>, HasCrCard <dbl>, IsActiveMember
<dbl>,
## #   EstimatedSalary <dbl>, Exited <dbl>
```

```
head(test)
```

```
## # A tibble: 6 × 13
##       id CustomerId Surname   CreditScore Geography Gender   Age Tenure
Balance
##    <dbl>      <dbl> <chr>           <dbl> <chr>     <chr>  <dbl>  <dbl>
<dbl>
## 1 165034   15773898 Lucchese          586 France    Female    23      2
0
## 2 165035   15782418 Nott              683 France    Female    46      2
0
## 3 165036   15807120 K?                656 France    Female    34      7
0
## 4 165037   15808905 O'Donnell         681 France    Male      36      8
0
## 5 165038   15607314 Higgins           752 Germany   Male      38     10
121264.
## 6 165039   15672704 Pearson           593 France    Female    22      9
0
## # ℹ 4 more variables: NumOfProducts <dbl>, HasCrCard <dbl>, IsActiveMember
<dbl>,
## #   EstimatedSalary <dbl>
```

```
head(sample_submission)
```

```
## # A tibble: 6 × 2
##       id Exited
##    <dbl>  <dbl>
## 1 165034    0.5
## 2 165035    0.5
## 3 165036    0.5
## 4 165037    0.5
## 5 165038    0.5
## 6 165039    0.5
```

Review the structure of the train data

```
str(train)
```

```
## spc_tbl_ [165,034 × 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id             : num [1:165034] 0 1 2 3 4 5 6 7 8 9 ...
##  $ CustomerId     : num [1:165034] 15674932 15749177 15694510 15741417
15766172 ...
##  $ Surname        : chr [1:165034] "Okwudilichukwu" "Okwudiliolisa"
```

```
"Hsueh" "Kao" ...
##  $ CreditScore    : num [1:165034] 668 627 678 581 716 588 593 678 676 583
...
##  $ Geography      : chr [1:165034] "France" "France" "France" "France" ...
##  $ Gender         : chr [1:165034] "Male" "Male" "Male" "Male" ...
##  $ Age            : num [1:165034] 33 33 40 34 33 36 30 37 43 40 ...
##  $ Tenure         : num [1:165034] 3 1 10 2 5 4 8 1 4 4 ...
##  $ Balance        : num [1:165034] 0 0 0 148883 0 ...
##  $ NumOfProducts  : num [1:165034] 2 2 2 1 2 1 1 1 2 1 ...
##  $ HasCrCard      : num [1:165034] 1 1 1 1 1 1 1 1 1 1 ...
##  $ IsActiveMember : num [1:165034] 0 1 0 1 1 0 0 0 0 1 ...
##  $ EstimatedSalary: num [1:165034] 181450 49504 184867 84561 15069 ...
##  $ Exited         : num [1:165034] 0 0 0 0 0 1 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   CustomerId = col_double(),
##   ..   Surname = col_character(),
##   ..   CreditScore = col_double(),
##   ..   Geography = col_character(),
##   ..   Gender = col_character(),
##   ..   Age = col_double(),
##   ..   Tenure = col_double(),
##   ..   Balance = col_double(),
##   ..   NumOfProducts = col_double(),
##   ..   HasCrCard = col_double(),
##   ..   IsActiveMember = col_double(),
##   ..   EstimatedSalary = col_double(),
##   ..   Exited = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The dataset includes the following attributes:

Customer ID: A unique identifier for each customer

Surname: The customer's surname or last name

Credit Score: A numerical value representing the customer's credit score

Geography: The country where the customer resides (France, Spain or Germany)

Gender: The customer's gender (Male or Female)

Age: The customer's age.

Tenure: The number of years the customer has been with the bank

Balance: The customer's account balance

NumOfProducts: The number of bank products the customer uses (e.g., savings ### account, credit card)

HasCrCard: Whether the customer has a credit card (1 = yes, 0 = no)

IsActiveMember: Whether the customer is an active member (1 = yes, 0 = no)

EstimatedSalary: The estimated salary of the customer

Exited: Whether the customer has churned (1 = yes, 0 = no)

Trim rows

Use trimrows function created previously

This will go through the data frame find the character columns and trim whitespaces around the values

```
train <- trimws_df(train)
```

Check for missing values

```
missing_values <- colSums(is.na(train))
head(missing_values)

##          id  CustomerId     Surname CreditScore   Geography      Gender
##           0           0           0           0           0           0
```

Analyze

Train Data

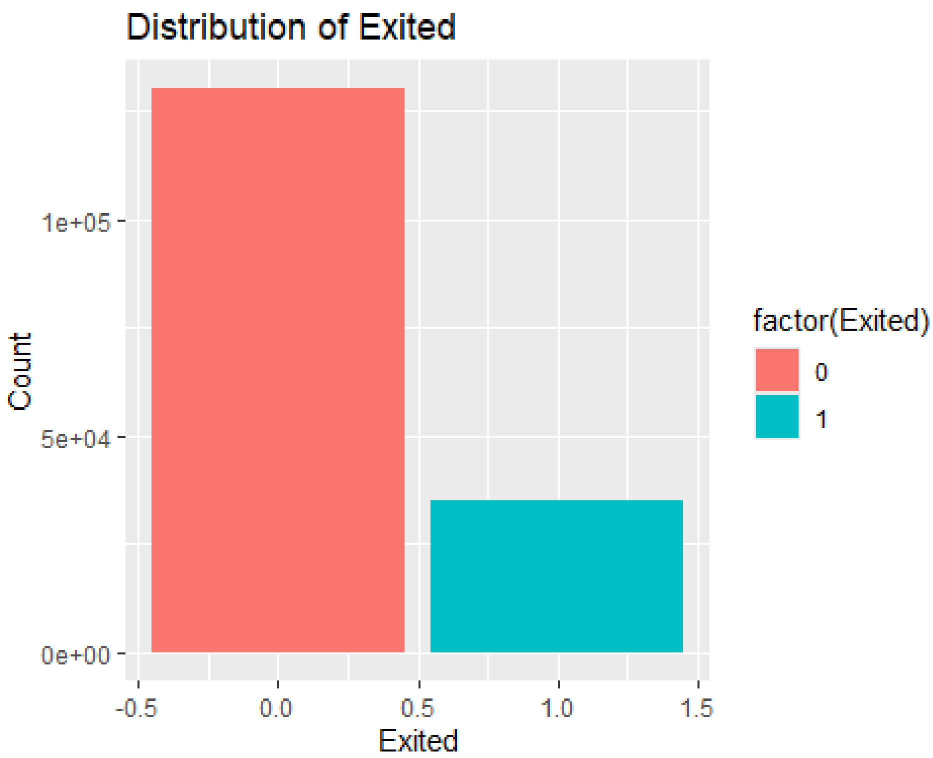
View summary statistics of train data

```
summary(train)

##        id           CustomerId           Surname           CreditScore   
##  Min.   :     0   Min.   :15565701   Length:165034      Min.   :350.0  
##  1st Qu.: 41258   1st Qu.:15633141   Class :character   1st Qu.:597.0  
##  Median : 82517   Median :15690169   Mode  :character   Median :659.0  
##  Mean   : 82517   Mean   :15692005                      Mean   :656.5  
##  3rd Qu.:123775   3rd Qu.:15756824                      3rd Qu.:710.0  
##  Max.   :165033   Max.   :15815690                      Max.   :850.0  
##   Geography            Gender               Age            Tenure     
##  Length:165034      Length:165034      Min.   :18.00   Min.   : 0.00  
##  Class :character   Class :character   1st Qu.:32.00   1st Qu.: 3.00  
##  Mode  :character   Mode  :character   Median :37.00   Median : 5.00  
##                                        Mean   :38.13   Mean   : 5.02  
##                                        3rd Qu.:42.00   3rd Qu.: 7.00  
##                                        Max.   :92.00   Max.   :10.00  
##     Balance       NumOfProducts     HasCrCard     IsActiveMember   
EstimatedSalary   
##  Min.   :     0   Min.   :1.000   Min.   :0.000   Min.   :0.0000   Min.   
:    11.58  
##  1st Qu.:     0   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.0000   1st 
Qu.: 74637.57  
##  Median :     0   Median :2.000   Median :1.000   Median :0.0000   Median 
:117948.00  
##  Mean   : 55478   Mean   :1.554   Mean   :0.754   Mean   :0.4978   Mean   
:112574.82  
##  3rd Qu.:119940   3rd Qu.:2.000   3rd Qu.:1.000   3rd Qu.:1.0000   3rd 
Qu.:155152.47  
##  Max.   :250898   Max.   :4.000   Max.   :1.000   Max.   :1.0000   Max.   
:199992.48  
##      Exited      
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.2116  
##  3rd Qu.:0.0000  
##  Max.   :1.0000
```

Visualize relationships using ggplot2

```
ggplot(train, aes(x = Exited, fill = factor(Exited))) +
  geom_bar() +
  labs(title = "Distribution of Exited",
       x = "Exited",
       y = "Count")
```



Explore relationships with 'Exited' for numerical variables

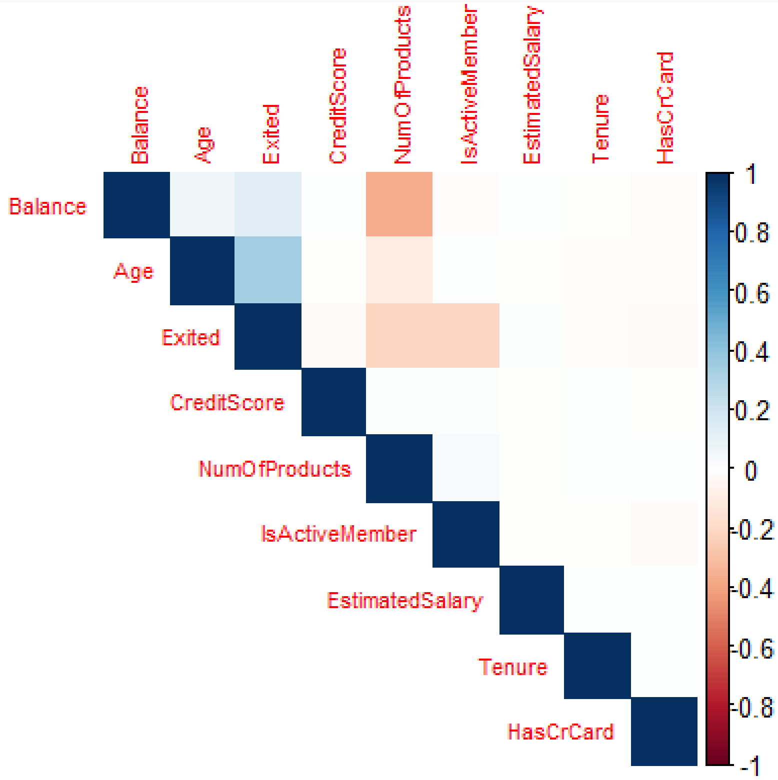
```
cor(train[, c("CreditScore", "Age", "Tenure", "Balance", "NumOfProducts",
"HasCrCard", "IsActiveMember", "EstimatedSalary", "Exited")])

##                   CreditScore          Age        Tenure      Balance
NumOfProducts
## CreditScore      1.0000000000 -0.008918146  0.0009424799  0.006973053
0.011360808
## Age             -0.0089181458  1.000000000 -0.0108303451  0.064318287
-0.102194912
## Tenure           0.0009424799 -0.010830345  1.0000000000 -0.009481186
0.007334828
## Balance          0.0069730535  0.064318287 -0.0094811862  1.000000000
-0.361032521
## NumOfProducts    0.0113608082 -0.102194912  0.0073348275 -0.361032521
1.000000000
## HasCrCard       -0.0028277567 -0.012111332  0.0053266159 -0.018584007
0.005482281
## IsActiveMember   0.0147902638  0.003319563 -0.0055322590 -0.015073487
0.039736070
## EstimatedSalary -0.0018203035 -0.005398663  0.0009705869  0.008586201
-0.004285089
## Exited          -0.0273826001  0.340768163 -0.0195648445  0.129742860
-0.214554232
##                    HasCrCard IsActiveMember EstimatedSalary      Exited
## CreditScore     -0.002827757    0.014790264   -0.0018203035 -0.02738260
```

```
## Age              -0.012111332    0.003319563   -0.0053986634  0.34076816
## Tenure            0.005326616   -0.005532259    0.0009705869 -0.01956484
## Balance          -0.018584007   -0.015073487    0.0085862012  0.12974286
## NumOfProducts     0.005482281    0.039736070   -0.0042850891 -0.21455423
## HasCrCard         1.000000000   -0.021033789    0.0044382187 -0.02214133
## IsActiveMember   -0.021033789    1.000000000   -0.0080800461 -0.21023703
## EstimatedSalary   0.004438219   -0.008080046    1.0000000000  0.01882681
## Exited           -0.022141333   -0.210237026    0.0188268057  1.00000000
```

Build a heatmap for better visualization of correlations

```r
cor_data <- train[, c("CreditScore", "Age", "Tenure", "Balance",
"NumOfProducts", "HasCrCard", "IsActiveMember", "EstimatedSalary", "Exited")]
cor_matrix <- cor(cor_data)
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
tl.cex = 0.7)
```



View character categorical variables

```r
table(train$Geography, train$Exited)
```

```
##
##              0     1
##   France  78643 15572
##   Germany 21492 13114
##   Spain   29978  6235
```

```r
table(train$Gender, train$Exited)
```

```
##
##              0     1
##   Female 51779 20105
##   Male   78334 14816
```

Convert character categorical variables to factors

```
train_encoded <- train %>%
  select(-c(id, CustomerId, Surname)) %>%
  mutate(Geography = as.factor(Geography),
         Gender = as.factor(Gender),
         Exited = as.factor(Exited))
```

Split data into features (X) and target variable (y)

```
X <- select(train_encoded, -Exited)
y <- train_encoded$Exited
```

Split data into training and testing sets

```
set.seed(123)
split_index <- createDataPartition(y, p = 0.7, list = FALSE)
train_data <- train_encoded[split_index, ]
test_data <- train_encoded[-split_index, ]

Log_train_data <- train_encoded[split_index, ]
Log_test_data <- train_encoded[-split_index, ]

XGB_train_data <- train_encoded[split_index, ]
XGB_test_data <- train_encoded[-split_index, ]
```

Turn Exited column into factor

```
Log_train_data$Exited <- as.factor(Log_train_data$Exited)
XGB_train_data$Exited <- as.factor(XGB_train_data$Exited)
```

Train logistic regression model

```
logistic_model <- glm(Exited ~ ., data = Log_train_data, family = "binomial")
summary(logistic_model)

##
## Call:
## glm(formula = Exited ~ ., family = "binomial", data = Log_train_data)
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -2.430e+00  8.784e-02 -27.662  < 2e-16 ***
## CreditScore      -7.890e-04  1.039e-04  -7.596 3.04e-14 ***
## GeographyGermany  1.154e+00  2.360e-02  48.882  < 2e-16 ***
## GeographySpain    4.251e-02  2.198e-02   1.934   0.0531 .
## GenderMale       -6.822e-01  1.675e-02 -40.723  < 2e-16 ***
## Age               9.430e-02  9.440e-04  99.894  < 2e-16 ***
## Tenure           -1.794e-02  2.965e-03  -6.051 1.44e-09 ***
```

```
## Balance           -1.971e-06  1.701e-07 -11.583  < 2e-16 ***
## NumOfProducts     -9.154e-01  1.650e-02 -55.490  < 2e-16 ***
## HasCrCard         -1.549e-01  1.914e-02  -8.093 5.83e-16 ***
## IsActiveMember    -1.293e+00  1.799e-02 -71.871  < 2e-16 ***
## EstimatedSalary    9.597e-07  1.663e-07   5.771 7.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 119237  on 115524  degrees of freedom
## Residual deviance:  91122  on 115513  degrees of freedom
## AIC: 91146
##
## Number of Fisher Scoring iterations: 5
```

Predictions on train_data using logistic regression

```
predictions_train_logistic <- predict(logistic_model, newdata =
Log_train_data, type = "response")
binary_predictions_train_logistic <- ifelse(predictions_train_logistic > 0.5,
1, 0)
train_data$PredictedExited_Logistic <- binary_predictions_train_logistic
```

Train XGBoost model

```
xgb_features <- setdiff(names(XGB_train_data), c("id", "Surname", "Exited",
"PredictedExited_Logistic"))
```

Ensure relevant columns are numeric

```
XGB_train_data[, xgb_features] <- lapply(XGB_train_data[, xgb_features],
as.numeric)
```

Convert Exited to integer

```
XGB_train_data$Exited <- as.integer(XGB_train_data$Exited) - 1
```

Create matrix with features and labels

```
xgb_matrix <- xgb.DMatrix(as.matrix(XGB_train_data[, xgb_features]), label =
XGB_train_data$Exited)
```

Create xgboost model

```
xgb_model <- xgboost(
  data = xgb_matrix,
  nrounds = 100,
  objective = "binary:logistic",
  eval_metric = "logloss",
  verbose = 1
)
```

```
## [1]  train-logloss:0.543701
## [2]  train-logloss:0.462336
## [3]  train-logloss:0.412461
## [4]  train-logloss:0.380961
## [5]  train-logloss:0.359584
## [6]  train-logloss:0.345226
## [7]  train-logloss:0.335974
## [8]  train-logloss:0.329861
## [9]  train-logloss:0.325658
## [10] train-logloss:0.322307
## [11] train-logloss:0.320131
## [12] train-logloss:0.318390
## [13] train-logloss:0.316737
## [14] train-logloss:0.315470
## [15] train-logloss:0.314520
## [16] train-logloss:0.313604
## [17] train-logloss:0.312482
## [18] train-logloss:0.311714
## [19] train-logloss:0.311183
## [20] train-logloss:0.310581
## [21] train-logloss:0.309813
## [22] train-logloss:0.309084
## [23] train-logloss:0.308712
## [24] train-logloss:0.308201
## [25] train-logloss:0.307895
## [26] train-logloss:0.307645
## [27] train-logloss:0.307268
## [28] train-logloss:0.306740
## [29] train-logloss:0.306295
## [30] train-logloss:0.305839
## [31] train-logloss:0.305605
## [32] train-logloss:0.305331
## [33] train-logloss:0.304996
## [34] train-logloss:0.304396
## [35] train-logloss:0.303979
## [36] train-logloss:0.303414
## [37] train-logloss:0.303012
## [38] train-logloss:0.302704
## [39] train-logloss:0.302152
## [40] train-logloss:0.301955
## [41] train-logloss:0.301361
## [42] train-logloss:0.301265
## [43] train-logloss:0.300832
## [44] train-logloss:0.300524
## [45] train-logloss:0.300114
## [46] train-logloss:0.299660
## [47] train-logloss:0.299616
## [48] train-logloss:0.299285
## [49] train-logloss:0.298938
## [50] train-logloss:0.298681
```

```
## [51] train-logloss:0.298142
## [52] train-logloss:0.298068
## [53] train-logloss:0.297926
## [54] train-logloss:0.297558
## [55] train-logloss:0.297218
## [56] train-logloss:0.297096
## [57] train-logloss:0.296917
## [58] train-logloss:0.296738
## [59] train-logloss:0.296551
## [60] train-logloss:0.296179
## [61] train-logloss:0.295673
## [62] train-logloss:0.295116
## [63] train-logloss:0.294991
## [64] train-logloss:0.294726
## [65] train-logloss:0.294525
## [66] train-logloss:0.294153
## [67] train-logloss:0.293980
## [68] train-logloss:0.293652
## [69] train-logloss:0.293333
## [70] train-logloss:0.293068
## [71] train-logloss:0.292707
## [72] train-logloss:0.292461
## [73] train-logloss:0.292430
## [74] train-logloss:0.292045
## [75] train-logloss:0.291600
## [76] train-logloss:0.291247
## [77] train-logloss:0.291174
## [78] train-logloss:0.290648
## [79] train-logloss:0.290502
## [80] train-logloss:0.290215
## [81] train-logloss:0.289993
## [82] train-logloss:0.289767
## [83] train-logloss:0.289315
## [84] train-logloss:0.289098
## [85] train-logloss:0.288972
## [86] train-logloss:0.288591
## [87] train-logloss:0.288351
## [88] train-logloss:0.287893
## [89] train-logloss:0.287826
## [90] train-logloss:0.287453
## [91] train-logloss:0.286964
## [92] train-logloss:0.286841
## [93] train-logloss:0.286783
## [94] train-logloss:0.286579
## [95] train-logloss:0.286352
## [96] train-logloss:0.286126
## [97] train-logloss:0.286106
## [98] train-logloss:0.285915
## [99] train-logloss:0.285555
## [100]    train-logloss:0.285188
```

Make predictions using XGBoost model

Convert probabilities to binary predictions

```
xgb_predictions_train <- predict(xgb_model, newdata = xgb_matrix)
binary_xgb_predictions_train <- ifelse(xgb_predictions_train > 0.5, 1, 0)
train_data$PredictedExited_XGBoost <- binary_xgb_predictions_train
```

Compare the 'Exited' columns for accuracy

```
accuracy_logistic <- sum(train_data$Exited ==
train_data$PredictedExited_Logistic) / nrow(train_data)
accuracy_xgb <- sum(train_data$Exited == binary_xgb_predictions_train) /
nrow(train_data)

cat("Logistic Regression Accuracy on train_data:", round(accuracy_logistic *
100, 2), "%\n")

## Logistic Regression Accuracy on train_data: 83.44 %

cat("XGBoost Accuracy on train_data:", round(accuracy_xgb * 100, 2), "%\n")

## XGBoost Accuracy on train_data: 88.07 %
```

Predictions

XGBoost

Convert character categorical variables to factors in test data

```
test_XGB <- test %>%
  select(-c(id, CustomerId, Surname)) %>%
  mutate(Geography = as.factor(Geography),
         Gender = as.factor(Gender))
```

Ensure relevant columns are numeric

```
test_XGB[, xgb_features] <- lapply(test_XGB[, xgb_features], as.numeric)
```

Create a matrix for our predictions

Run the XGBoost prediction model on the `test_XGB` data set

```
xgb_matrix_test <- xgb.DMatrix(as.matrix(test_XGB[, xgb_features]))
xgb_predictions_test <- predict(xgb_model, newdata = xgb_matrix_test)
binary_xgb_predictions_test <- ifelse(xgb_predictions_test > 0.5, 1, 0)
test_XGB$PredictedExited_XGBoost <- binary_xgb_predictions_test
```

Evaluate the model on test data using ROC curve

```
roc_curve_test <- roc(test_XGB$PredictedExited_XGBoost, xgb_predictions_test)

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

## Setting direction: controls < cases
```

Logistical Model

Create a new data frame for our logistical model predictions

```
test_Logistic <- test %>%
  select(-c(id, CustomerId, Surname)) %>%
  mutate(Geography = as.factor(Geography),
         Gender = as.factor(Gender))
```

Run the logistical predictions on the `test_Logistic` data set

```
predictions_test_logistic <- predict(logistic_model, newdata = test_Logistic,
type = "response")
binary_predictions_test_logistic <- ifelse(predictions_test_logistic > 0.5,
1, 0)
test_Logistic$PredictedExited_Logistic <- binary_predictions_test_logistic
```

Evaluate the model on test data using ROC curve for logistic regression

```
roc_curve_logistic_test <- roc(test_Logistic$PredictedExited_Logistic,
predictions_test_logistic)

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

## Setting direction: controls < cases
```

Share

Visual

Summary of ROC curves

```
summary(roc_curve_test)

##                    Length Class  Mode
## percent                 1 -none- logical
## sensitivities      108248 -none- numeric
## specificities      108248 -none- numeric
## thresholds         108248 -none- numeric
## direction               1 -none- character
## cases               17466 -none- numeric
## controls            92557 -none- numeric
## fun.sesp                1 -none- function
## auc                     1 auc    numeric
## call                    3 -none- call
## original.predictor 110023 -none- numeric
## original.response  110023 -none- numeric
## predictor          110023 -none- numeric
## response           110023 -none- numeric
## levels                  2 -none- character

summary(roc_curve_logistic_test)
```

```
##                    Length Class  Mode
## percent                 1 -none- logical
## sensitivities      109952 -none- numeric
## specificities      109952 -none- numeric
## thresholds         109952 -none- numeric
## direction               1 -none- character
## cases               12970 -none- numeric
## controls            97053 -none- numeric
## fun.sesp                1 -none- function
## auc                     1 auc    numeric
## call                    3 -none- call
## original.predictor 110023 -none- numeric
## original.response  110023 -none- numeric
## predictor          110023 -none- numeric
## response           110023 -none- numeric
## levels                  2 -none- character
```

AUC of ROC curves

```
auc_test <- auc(roc_curve_test)
auc_logistic_test <- auc(roc_curve_logistic_test)

cat("XGBoost AUC on test data:", auc_test, "\n")

## XGBoost AUC on test data: 1

cat("Logistic Regression AUC on test data:", auc_logistic_test, "\n")

## Logistic Regression AUC on test data: 1
```

Archive

Create submission data frames

```
Logistic_submission <- data.frame(id = test$id, Exited =
predictions_test_logistic)

XGBoost_submission <- data.frame(id = test$id, Exited = xgb_predictions_test)
```

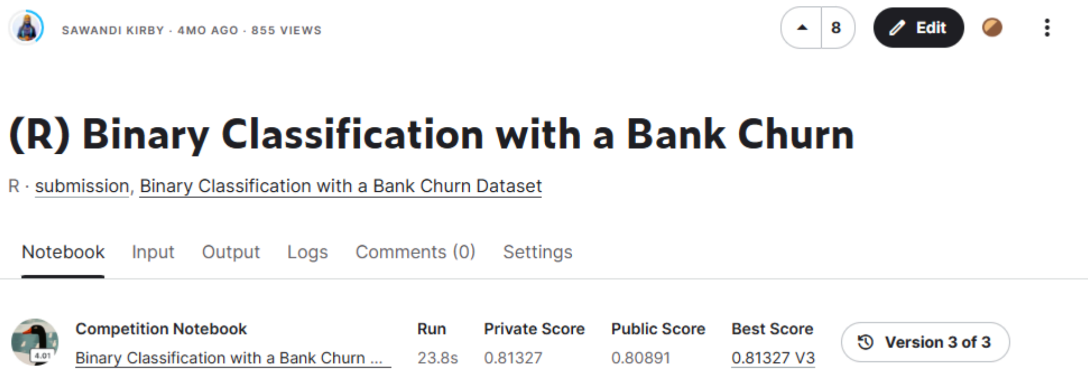
Logistic Regression Submission Entry Score = 0.80891

Review the submission data set

```
head(Logistic_submission)

##       id     Exited
## 1 165034 0.02346466
## 2 165035 0.58237170
## 3 165036 0.15197213
## 4 165037 0.20440605
```

```
## 5 165038 0.41989777
## 6 165039 0.05921289
```



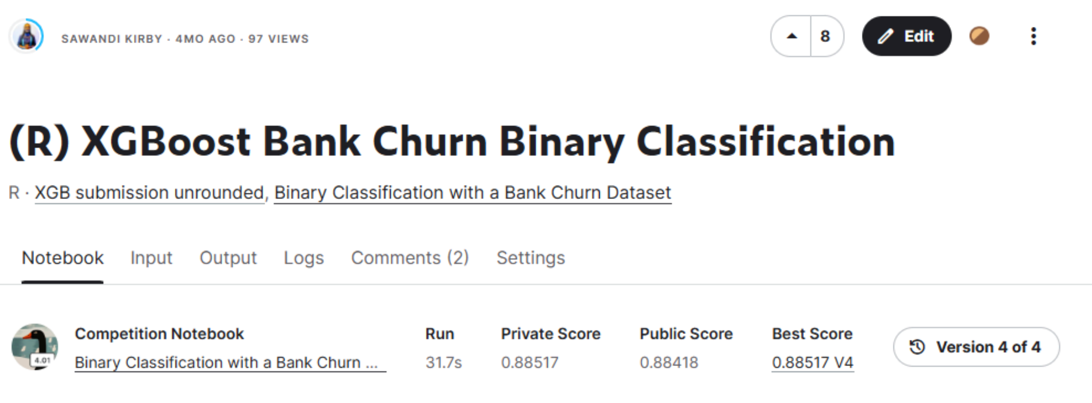
Logistic Submission

XGBoost Submission Entry Score = .88418

Review the submission data set

```
head(XGBoost_submission)

##        id      Exited
## 1 165034 0.04157397
## 2 165035 0.84078336
## 3 165036 0.02030335
## 4 165037 0.23674323
## 5 165038 0.37523997
## 6 165039 0.03150999
```



XGBoost Submission