Churn Case

Mette

2024-03-07

Table of contents

Figure 1: Churn rates

1 Introduction

This case focuses on customer churn (also known as customer attrition or customer turnover). Customer churn is interesting because it is usually much cheaper to retain existing customers than to acquire new ones. Instead of focusing on each individual customer, we will attempt to build a predictive model that can help us decide which customers we should focus our retention efforts on.

2 Analysis

I follow the CRISP-DM (Cross-Industry Standard Process for Data Mining) framework in my data mining projects, guiding me through six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. This structured approach ensures I effectively extract insights and apply data science.

2.1 Business Understanding

Understanding customer churn is critical for banks as it aids in cost reduction by prioritizing the retention of existing customers over acquiring new ones. This not only helps in stabilizing revenue but also enhances customer satisfaction by addressing their specific needs and concerns. By gaining insights into churn patterns, banks can develop targeted strategies, optimize resource allocation, and gain a competitive edge in the market.

2.2 Data Understanding

2.2.1 Reading libraries

We will start by loading the nessescary libraries

```r
pacman::p_load("tidyverse", "magrittr", "nycflights13", "gapminder",
               "Lahman", "maps", "lubridate", "pryr", "hms", "hexbin",
               "feather", "htmlwidgets", "broom", "pander", "modelr",
               "XML", "httr", "jsonlite", "lubridate", "microbenchmark",
               "splines", "ISLR2", "MASS", "testthat",  "caret",
               "RSQLite", "class", "babynames", "nasaweather", "pls",
               "fueleconomy", "viridis", "boot", "devtools", "tree", "leaps",
               "glmnet", "gam", "akima", "factoextra", "randomForest", "gbm",
               "ggrepel", "GGally", "fmsb", "sjPlot", "rcompanion", "DT")
```

2.2.2 Importing data

The dataset we are going to work with will be imported and investigated.

```r
#bank_churn <- read.csv("Churn_Modelling.csv")
bank_churn <- read.csv("C:/Users/mette/OneDrive/Skrivebord/PB dataanalyse/Programmering og s


#tjekker data og klasser
str(bank_churn)

#vi skaber en interaktiv tabel

datatable(bank_churn, caption = htmltools::tags$caption(
    style = 'caption-side: top; text-align: center;',
    'Table 1: ', htmltools::em('Bank data '))
    )
```

Show 10 ∨ entries

Search: [   ]

Table 1: *Bank data*

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProduc
1	1	15634602	Hargrave	619	France	Female	42	2	0	
2	2	15647311	Hill	608	Spain	Female	41	1	83807.86	
3	3	15619304	Onio	502	France	Female	42	8	159660.8	
4	4	15701354	Boni	699	France	Female	39	1	0	
5	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	
6	6	15574012	Chu	645	Spain	Male	44	8	113755.78	
7	7	15592531	Bartlett	822	France	Male	50	7	0	
8	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	
9	9	15792365	He	501	France	Male	44	4	142051.07	
10	10	15592389	H?	684	France	Male	27	2	134603.88	

Showing 1 to 10 of 10,000 entries

Previous | 1 | 2 | 3 | 4 | 5 | … | 1,000 | Next

4

2.3 Data Preparation

2.3.1 Cleaning data

First we are checking for missing values. There are no missing values in the TotalCharges variable. We are imputing thesevalues with the meanvalue, since the quantity is low. After this we are changes all the character class variables to factors for later statistical analysis. The data is normalised and finally the churndataset is again changes to a dataframe and the CustomerID variable is removed.

```r
# Beregn antallet af missing værdier i hver kolonne. No missing
bank_churn %>% purrr::map(~ sum(is.na(.)))

summary(bank_churn)

# There are no missin values, so we can proceed.

#the relevant variables are converted into factors.

# Type = factor and integers---------------------------------------------------------

bank_churn_fact <- bank_churn %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.integer, as.factor)

bank_churn_fact$CreditScore <- as.integer(bank_churn_fact$CreditScore)
bank_churn_fact$Age <- as.integer(bank_churn_fact$Age)
bank_churn_fact$Tenure <- as.integer(bank_churn_fact$Tenure)
str(bank_churn_fact)

# Normalisering -----------------------------------------------------

# Det er ikke nødvendigt at normalisere data i forbindelse med de statistiske
# modeller, som vi skal køre her. Der er forskellige typer af normalisering.
# Vi ser her på følgende:

normalize <- function(x) {
  ((x-min(x))/(max(x)-min(x)))
}

bank_churn_fact <- bank_churn_fact %>%
  mutate_if(is.numeric, normalize)
```

```r
bank_churn_fact <- bank_churn_fact %>%
  dplyr::select(Exited, everything())

glimpse(bank_churn_fact)

numeric_columns <- sapply(bank_churn_fact, is.numeric)

# Konverter numeriske variable fra dbl til int
#bank_churn_fact[numeric_columns] <- lapply(bank_churn_fact[numeric_columns], as.integer)

# Fravalg af customerID, Efternavn og ID -------------------------------------------------

bank_churn_fact <- bank_churn_fact %>%
  dplyr::select(-RowNumber, -CustomerId, -Surname)

names(bank_churn_fact)

#Ændre variablen Exited til Churn

bank_churn_fact <- bank_churn_fact %>%
  rename(Churn = Exited)

bank_churn_fact$Churn <- ifelse(bank_churn_fact$Churn == 1, "Yes", "No")
bank_churn_fact$Churn <- as.factor(bank_churn_fact$Churn)
str((bank_churn_fact))
```

This creates a dataset with 10.000 observations, that can be investigated and is ready for analysis.

```r
# Install DT package
install.packages("DT")

# Load DT package
library(DT)

library(DT)

datatable(bank_churn_fact,
          options = list(pageLength = 5, autoWidth = TRUE),
          caption = 'Table 1: Telecommunication data')
```

Show 5 ∨ entries   Search: [   ]

Table 1: Telecommunication data

	Churn	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfPr
1	Yes	0.4967320261437909	France	Female	0.3478260869565217	0.2	0	1
2	No	0.4727668845315904	Spain	Female	0.3333333333333333	0.1	0.334031478677259	1
3	Yes	0.2418300653594771	France	Female	0.3478260869565217	0.8	0.6363571759354565	3
4	No	0.6710239651416122	France	Female	0.3043478260869565	0.1	0	2
5	No	1	Spain	Female	0.3623188405797101	0.2	0.5002462155052675	1

Showing 1 to 5 of 10,000 entries   Previous 1 2 3 4 5 … 2,000 Next

7

2.4 Modeling

Training and test data

```
#træningsdata og testdata


# Vi bruger funktionen set.seed, så vi kan reproducere vores resultater
set.seed(5)
# træningsdel og testdel:
intrain <- createDataPartition(y=bank_churn_fact$Churn,
                               p=0.80, list=FALSE)

# list=FALSE betyder at outputtet bliver en matrix og denne kan bruges
# i koden nedenfor:

train <- bank_churn_fact[intrain,]
test <- bank_churn_fact[-intrain,]
```

2.4.1 Cost Assessment

The cost assessment is significant when deciding on the threshold in connection with, for example, logistic regression. The relative cost of the different errors that can be made affects where it is optimal to place the threshold. Optimal in terms of reducing costs. When we do not have any a priori knowledge about the relative costs, we use a 50/50 split. But in this example, the situation is different. It does not cost the same to commit the different errors.

```
#                       |  Vil churne  | vil ikke churne
# predikte churne       |      TP      |      FP
# predikte ikke churne  |      FN      |      TN
```

Customer acquisition $200 Se documentation here){.uri} - the cost for a false negative (FN) prediction That is, predicting that a customer is satisfied when in reality they churn. Customer retention $40 (Source: Bain & Company, "The Value of Online Customer Loyalty in Retail Banking," 2016.) - the cost of a false positive (FP) That is, predicting that a customer will churn when in reality the customer was satisfied, and a true positive (TP) that is, correctly predicting dissatisfied customers. Correctly predicted true negatives (TN) cost nothing. That is, correctly predicting that a customer is satisfied.

The situation until now is that the company assumes that no customers churn. Thus the model is that nobody churns. In other words, the company predicts a NO for every customer

in the test dataset ,i.e., TP=FP=0. We only need to calculate FN (since TN does not cost anything) ourselves.

The total savings from the new model based on a customer base of 10.000 customers:

```
FN_omk <- 200

TP_omk <- 40

 FP_omk <- TP_omk

 TN_omk <- 0
```

Calculation of the current model's costs based on the above information. Since TN is free, we only need to calculate FN on the test dataset. Remember, that all are expected not to churn, but a certain portion will churn.

```
test$no_churn <- "No"
FN_simple <- table(test$Churn, test$no_churn)[2]/(table(test$Churn, test$no_churn)[1]+
                                        table(test$Churn, test$no_churn)[2])

omkostninger_mavefornemmelse <- FN_omk*FN_simple # per kunde


test <- dplyr::select(test, -no_churn) # We dont need no churn anymore.
```

In order to evalutate the relevant variables we will perform Lasso modelling

```
bank_churn_lasso <- bank_churn

bank_churn_lasso <- bank_churn_lasso %>%
  dplyr::select(-RowNumber, -CustomerId, -Surname)

names(bank_churn_fact)

#Ændre variablen Exited til Churn

bank_churn_lasso <- bank_churn_lasso %>%
  dplyr::rename(Churn = Exited)
```

```r
str(bank_churn_lasso)


x <- model.matrix(Churn ~ ., bank_churn_lasso)[, -1]
y <- bank_churn_lasso$Churn

grid <- 10^seq(10, -2, length = 100)
lasso.mod <- glmnet(x, y, alpha = 1, lambda = grid)

coef(lasso.mod)
dim(coef(lasso.mod))

names(bank_churn_lasso)

set.seed(5)
train <- sample(1:nrow(x), nrow(x)*2/3)
test <- (-train)
y.test <- y[test]


set.seed(5)
cv.out <- cv.glmnet(x[train, ], y[train], alpha = 1)
plot(cv.out)
```

```r
bestlam <- cv.out$lambda.min
bestlam # optimale
cv.out$lambda
cv.out$lambda.1se

log(bestlam)

lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[test, ])
mse_lasso <- mean((lasso.pred - y.test)^2)

mse_lasso

out <- glmnet(x, y, alpha = 1)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:8, ]

#print koeficienterne
lasso.coef
```

Based on the lassomodel, the following variables are relevant

```
lasso.coef[lasso.coef != 0]
```

```
     (Intercept)       CreditScore GeographyGermany          GenderMale
   -1.033746e-01      -6.073331e-05     1.191796e-01        -6.837273e-02
             Age            Tenure          Balance
    1.079818e-02      -7.884361e-04     2.934616e-07
```

Now we will perform logistic regression

```
#training and test data

train <- bank_churn_fact[intrain,]
test <- bank_churn_fact[-intrain,]

names(test)

outcome <- "Churn"

#Vi så i lasso regression, hvilke variabler der ikke havde relevans, disse eksluderes

variables <- c( ".", "NumOfProducts", "HasCrCard", "IsActiveMember",
                        "EstimatedSalary")



f <- as.formula(paste(outcome,
                    paste(variables, collapse = " - "), sep = " ~ "))



# Vi fitter en logistisk regressionsmodel:

fit_logit <- glm(f, data=train, family = "binomial")


# Foretage prædiktioner på testsættet og gemmer dem i et objekt, churn_probs:

churn_probs <- predict(fit_logit, test, type = "response")

head(churn_probs)
```

```
# Kan vi gøre det bedre end den simple model (og modellen med 50/50 split)
# Loop:

thresh <- seq(0.01, 1.0, length = 100)
omk <- rep(0, length(thresh))

for (i in 1:length(thresh)) {
  glm.pred <- rep("No", length(churn_probs))
  glm.pred[churn_probs>thresh[i]] <- "Yes"
  glm.pred <- as.factor(glm.pred)
  x <- confusionMatrix(glm.pred, test$Churn, positive = "Yes")
  total <- x$table[1] + x$table[2] + x$table[3] + x$table[4]
  TN <- x$table[1]/total
  FP <- x$table[2]/total
  FN <- x$table[3]/total
  TP <- x$table[4]/total
  omk[i] <- FN*FN_omk + TP*TP_omk + FP*FP_omk + TN*0
}
```

We repeat the above procedure, but only for a model where threshold = 0.5. We call this model simple_model, and we compare it with different thresholds.

```
glm.pred <- rep("No", length(churn_probs))
glm.pred[churn_probs>0.5] <- "Yes"
glm.pred <- as.factor(glm.pred)
x <- confusionMatrix(glm.pred, test$Churn, positive = "Yes")
total <- x$table[1] + x$table[2] + x$table[3] + x$table[4]
TN <- x$table[1]/total
FP <- x$table[2]/total
FN <- x$table[3]/total
TP <- x$table[4]/total
omk_simple <- FN*FN_omk + TP*TP_omk + FP*FP_omk + TN*0
```

We visualize the results to be able to see in a single way how much the costs depend on the thresholds.

```
model <- c(rep("optimized", 100), "simple")
cost_thresh <- c(omk, omk_simple)
thresh_plot <- c(thresh, 0.5)
```

The visualization shows that the optimum threshold is 0,17.

```
dataII <- data.frame(
  model,
  cost_thresh,
  thresh_plot
)

ggplot(dataII, aes(x = thresh_plot, y = cost_thresh, group = model, colour = model)) +
  geom_line() +
  geom_point() +
  geom_text(data = subset(dataII, cost_thresh == min(cost_thresh)),
            aes(label = paste("(", round(thresh_plot, 2), ",", round(cost_thresh, 2), ")"),
                x = thresh_plot + 0.05, y = cost_thresh),
            vjust = -0.5, hjust = 0)
```
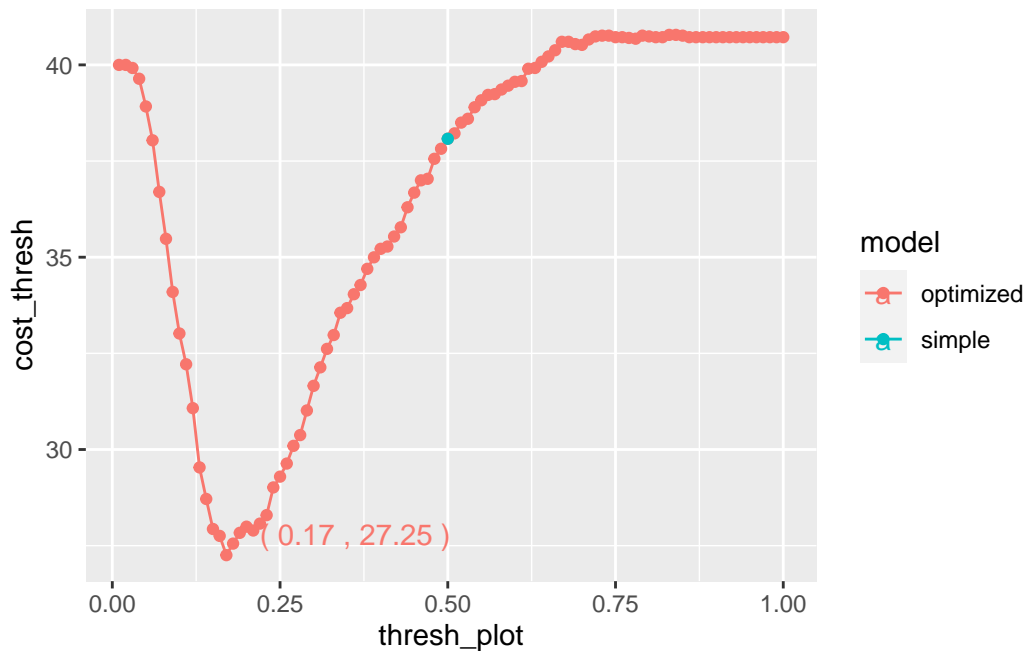

Note that we can find an optimum with a different threshold than 0.50.

Calculates the saved costs of the optimized model (threshold=0.17) compared to the baseline model (threshold=0.5)."

```
besparelse_pr_kunde <- omk_simple - min(omk)

besparelse_pr_kunde*10000
```

```
[1] 108254.1
```

Calculates the saved costs of the optimized model (threshold=0.17) compared to one's own calculations."

```
besparelse_mavefornemmelse <- omkostninger_mavefornemmelse - min(omk)

besparelse_mavefornemmelse*10000
```

```
[1] 134667.3
```

We can clearly outperform the existing model that the company uses (which assumes no churn). We can also clearly outperform the common 50/50 split.

Can we do even better with an alternative model: Let's use linear and quadratic discriminant analysis and qudratic discriminant analysis, and we calculate again the total cost savings, and compare them with the best logistic regression:

2.4.1.1 Linear Discriminant Analysis (LDA)

LDA is a method used for classification and dimensionality reduction. It finds the best linear combination of features to separate different classes in the dataset. LDA works well when classes are distinct and follows normal distributions with equal covariance. It's useful for understanding which features are most important for classification. However, it may not perform well with overlapping classes or outliers. Overall, LDA is effective for classification tasks with well-separated classes and can provide valuable insights into the data structure.

```
# lda
lda.fit <- lda(f, data=train)

lda.pred <- data.frame(predict(lda.fit, test))
omk_lda <- rep(0,length(thresh))
thresh <- seq(0.01, 1.0, length = 100)

for (i in 1:length(thresh)) {
  glm.pred <- rep("No", length(lda.pred$posterior.Yes))  # laver en vektor med No
  glm.pred[lda.pred$posterior.Yes > thresh[i]] <- "Yes" # ændrer værdien af glm.pred, hvis p
  glm.pred <- as.factor(glm.pred) # sikrer, at variablen er en faktor
  x <- confusionMatrix(glm.pred, test$Churn, positive = "Yes")
  total <- x$table[1] + x$table[2] + x$table[3] + x$table[4]
  TN <- x$table[1]/total
```

```
  FP <- x$table[2]/total
  FN <- x$table[3]/total
  TP <- x$table[4]/total
  omk_lda[i] <- FN*FN_omk + TP*TP_omk + FP*FP_omk + TN*0


}
```

After running the LDA analysis we can compare the savings to the prior models

```
#LDA analysis minimum costs minus logistic analysis minimum costs
10000*(min(omk_lda) - min(omk))
```

```
[1] -400.2001
```

```
#LDA analysis minimum costs minus simple model minimum costs
10000*(min(omk_lda) - omk_simple)
```

```
[1] -108654.3
```

```
#LDA analysis minimum costs minus own calculation costs
10000*(min(omk_lda) - omkostninger_mavefornemmelse)
```

```
[1] -135067.5
```

2.4.1.2 Quadratic Discriminant Analysis (QDA)

QDA is similar to LDA but allows for different covariance matrices for each class, making it more flexible in capturing complex relationships between features. It works by estimating separate covariance matrices for each class, which can better capture non-linear decision boundaries. QDA is beneficial when classes have different variances or when the decision boundary is non-linear. However, it requires more parameters to estimate compared to LDA and may overfit with small datasets. Overall, QDA is useful for classification tasks with non-linear decision boundaries and varying variances between classes.

```
#qda

qda.fit <- qda(f, data=train)
qda.pred <- data.frame(predict(qda.fit, test))
omk_qda <- rep(0,length(thresh))
```

```
thresh <- seq(0.01, 1.0, length = 100)

for (i in 1:length(thresh)) {
  glm.pred <- rep("No", length(qda.pred$posterior.Yes))  # laver en vektor med No
  glm.pred[qda.pred$posterior.Yes > thresh[i]] <- "Yes" # ændrer værdien af glm-pred, hvis p
  glm.pred <- as.factor(glm.pred) # sikrer at variablen er en faktor
  x <- confusionMatrix(glm.pred, test$Churn, positive = "Yes")
  total <- x$table[1] + x$table[2] + x$table[3] + x$table[4]
  TN <- x$table[1]/total
  FP <- x$table[2]/total
  FN <- x$table[3]/total
  TP <- x$table[4]/total
  omk_qda[i] <- FN*FN_omk + TP*TP_omk + FP*FP_omk + TN*0

}
```

After running the QDA analysis we can compare the savings to the prior models

```
#QDA analysis minimum costs minus logistic analysis minimum costs
10000*(min(omk_qda) - min(omk))
```

```
[1] 15007.5
```

```
#QDA analysis minimum costs minus LDA minimums costs
10000*(min(omk_qda) - min(omk_lda))
```

```
[1] 15407.7
```

```
#QDA analysis minimum costs minus simple model minimum costs
10000*(min(omk_qda) - omk_simple)
```

```
[1] -93246.62
```

```
#QDA analysis minimum costs minus own calculation costs
10000*(min(omk_qda) - omkostninger_mavefornemmelse)
```

```
[1] -119659.8
```

2.5 Evaluation

We can now asses the performance of the models. As we can see we are able to save money by focusing on data driven decisions.

2.6 Deployment

The code is scalable since it can be adapted to changes in costs. If we assume that retention cost is double, we only have to change one parameter.

```
TP_omk <- 80
```

And run the code again.

Furthermore once models are trained and evaluated, they're deployed for real-world use. Here's how:

1. **Packaging**: Bundle the trained model with preprocessing steps for easy deployment.

2. **Integration**: Integrate the model into existing systems, ensuring compatibility.

3. **Optimization**: Optimize for scalability and performance to handle real-time requests.

4. **Monitoring**: Monitor model performance over time and update regularly.

5. **Security**: Ensure compliance with security regulations and protect data privacy.

6. **Documentation**: Provide user-friendly documentation and support resources.

7. **Feedback**: Gather feedback for continuous improvement and iteration.

3 Conslusion

Based on the analysis, it is recommended that the bank focuses its retention efforts on customers identified as high-risk churners by the predictive models. This targeted approach can help optimize resource allocation and improve overall customer retention strategies.

In conclusion, the analysis highlights the importance of leveraging data-driven approaches to understand and address customer churn effectively. By implementing the recommended strategies, the bank can enhance customer satisfaction, reduce churn rates, and ultimately, drive long-term business growth and profitability.