10 Fallstudie: Churn Prediction

Martin Hanewald

2019-02-19

Packages

Überblick

Dataset

Zwei Datensätze:

- User-Stammdaten
- Aktivitäten pro User

```
activity_raw <-  read_csv('data/activity_info.csv')
user_raw <- read_csv('data/user_info.csv')
```

Pre-Processing

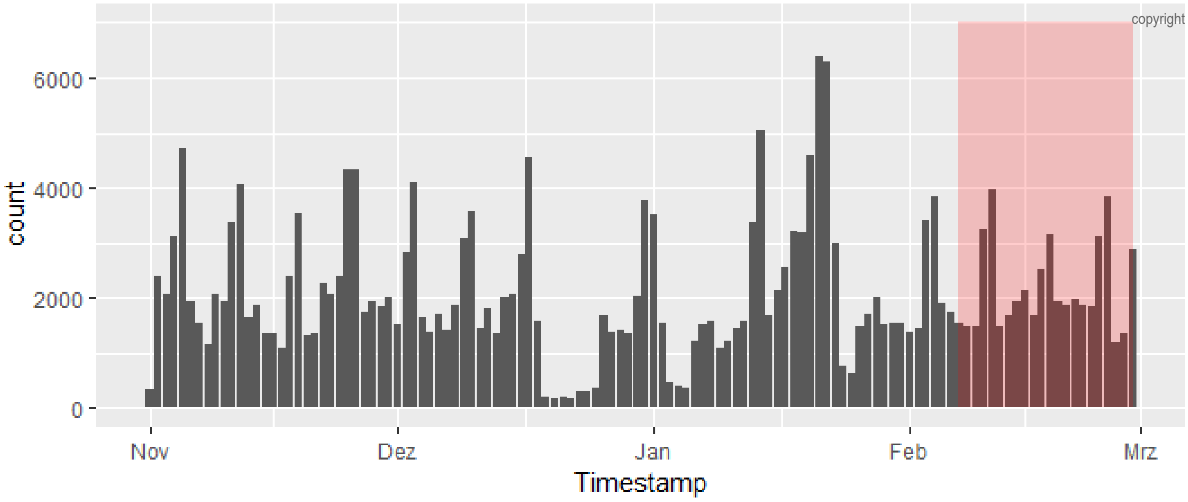
```
CHURN_PERIOD <- 21
CHURN_THRESHOLD <- 0
```

Cleanup

```
activity <- activity_raw %>%
    mutate(Timestamp = mdy_hm(Timestamp) %>% as_date() + years(17)) %>%
    mutate(ItemId = str_pad(ItemId, 12, pad='0')) %>%
    mutate_at(vars(TransactionId, UserId, ItemId), as.factor) %>%
    select(-Location, -ProductCategory, -X1)

user <- user_raw %>%
    select(-Gender, -UserType) %>%
    mutate_all(as.factor)
```

Create the `churn` label

```
last_date <- max(activity$Timestamp)

# Show Plot with churn period
activity %>%
    ggplot(aes(Timestamp)) + geom_bar() +
    annotate(geom='rect',
             xmin=last_date-CHURN_PERIOD,
             ymin = 0, ymax=7000, xmax=last_date,
             fill='red', alpha=.2)
```




```
# Approach one: For-loop, save results in list
churn <- list()

for(u in unique(activity$UserId)){
    churn[[u]] <- activity %>%
        filter(UserId == u) %>%
        filter(Timestamp > last_date - CHURN_PERIOD) %>%
        nrow() <= CHURN_THRESHOLD
}

churn_label <- churn %>% as_tibble() %>%
    gather(UserId, churn)

# profiler: 25,56 sec
```

```
# Approach two: Group_by and summarise
churn_label <- activity %>%
    mutate(in_period = Timestamp > last_date - CHURN_PERIOD) %>%
    select(UserId, TransactionId, in_period) %>% unique() %>%
    group_by(UserId) %>%
    summarise(churn = sum(in_period) <= CHURN_THRESHOLD) %>%
    mutate_at(vars(churn), as.factor)

# profiler: 3,6 sec

churn_label$churn %>% table()
#> .
#> FALSE  TRUE
#>  4492  5508
```

Feature Engineering

- Total_Quantity
- Total_Value
- StDev_Quantity
- StDev_Value
- AvgTimeDelta
- Recency
- Count_Unique_TransactionId
- Count_Unique_ItemId

- Mean_Quantity_per_Unique_TransactionId
- Mean_Quantity_per_Unique_ItemId
- Mean_Value_per_Unique_TransactionId
- Mean_Value_per_Unique_ItemId

```
activity_before <- activity %>% filter(Timestamp < last_date - CHURN_PERIOD)

activity_measures <-
    activity_before %>%
        group_by(UserId) %>%
        arrange(Timestamp, .by_group=TRUE) %>%
        summarise(Total_Quantity = sum(Quantity),
                  Total_Value = sum(Value),
                  StDev_Quantity = sd(Quantity),
                  StDev_Value = sd(Value),
                  AvgTimeDelta = mean(diff(Timestamp)),
                  Recency = last_date - CHURN_PERIOD - max(Timestamp)) %>%
    full_join(
        activity_before %>%
            group_by(UserId, ItemId) %>%
            summarise(Quantity_sum = sum(Quantity),
                      Value_sum = sum(Value)) %>%
            summarise(Mean_Quantity_per_Unique_ItemId = mean(Quantity_sum),
                      Mean_Value_per_Unique_ItemId = mean(Value_sum),
                      Count_Unique_ItemId = n_distinct(ItemId))) %>%
    full_join(
        activity_before %>%
            group_by(UserId, TransactionId) %>%
            summarise(Quantity_sum = sum(Quantity),
                      Value_sum = sum(Value)) %>%
            summarise(Mean_Quantity_per_Unique_TransactionId = mean(Quantity_sum),
                      Mean_Value_per_Unique_TransactionId = mean(Value_sum),
                      Count_Unique_TransactionId = n_distinct(TransactionId)))
#> Joining, by = "UserId"
#> Joining, by = "UserId"
```

Bind final model dataframe

```
modeldat <- user %>%
    inner_join(churn_label) %>%
    inner_join(activity_measures) %>%
    drop_na() # 724 lines dropped where only one transaction per user
#> Joining, by = "UserId"
#> Joining, by = "UserId"

# check for na's
modeldat %>%
    summarise_all(function(x) is.na(x) %>% sum()) %>%
    gather(var, na) %>% filter(na>0)
#> # A tibble: 0 x 2
#> # ... with 2 variables: var <chr>, na <int>
```

Modelling

Splitting

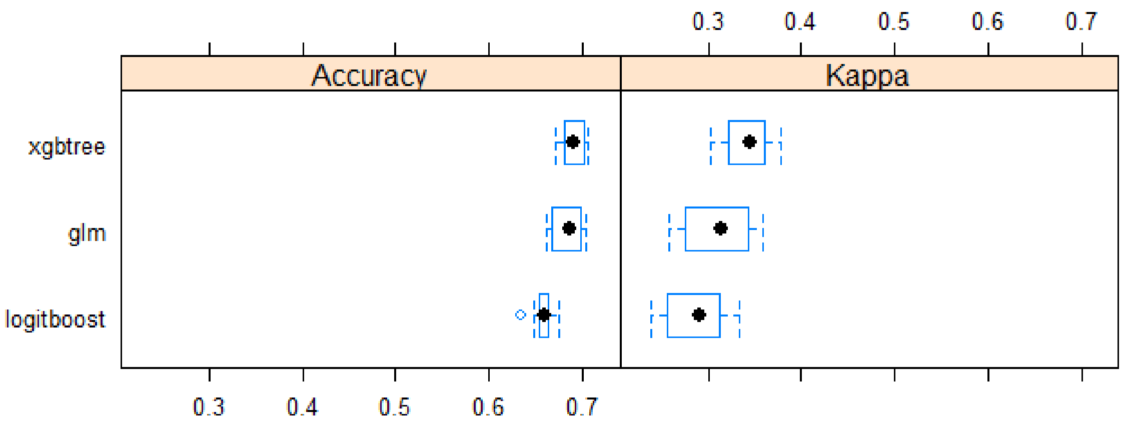
```
inTrain <- createDataPartition(modeldat$churn, p = .8, list=F)
training <- modeldat[inTrain,]
testing <- modeldat[-inTrain,]
```

Training

```
trControl <- trainControl(method='cv')

fit <- list()

fit$glm <- train(churn ~ . -UserId,
                 method = 'glm',
                 family = 'binomial',
                 trControl = trControl,
                 preProcess = c('center', 'scale'),
                 data = training)

fit$logitboost <- train(churn ~ . -UserId,
                 method = 'LogitBoost',
                 trControl = trControl,
                 preProcess = c('center', 'scale'),
                 data = training)

fit$xgbtree <- train(churn ~ . -UserId,
                 method = 'xgbTree',
                 trControl = trControl,
                 preProcess = c('center', 'scale'),
                 data = training)
```

Evaluation

```
resamples(fit) %>% bwplot()
```




```
varImp(fit$glm)
#> glm variable importance
#>
#>   only 20 most important variables shown (out of 29)
#>
```

```
#>                                      Overall
#> Recency                              100.000
#> AvgTimeDelta                          68.117
#> AddressE                              43.496
#> AddressF                              33.461
#> Mean_Value_per_Unique_TransactionId   28.174
#> Count_Unique_TransactionId            27.860
#> AddressD                              27.566
#> StDev_Value                           22.180
#> AddressH                              21.786
#> AddressG                              18.899
#> AgeJ                                  14.706
#> Mean_Value_per_Unique_ItemId          13.378
#> AddressC                              11.109
#> Total_Quantity                        10.964
#> Total_Value                           10.604
#> AgeC                                   9.446
#> AddressB                               7.652
#> AgeB                                   5.492
#> AgeI                                   4.946
#> AgeK                                   4.720
#varImp(fit$logitboost)
varImp(fit$xgbtree)
#> xgbTree variable importance
#>
#>   only 20 most important variables shown (out of 29)
#>
#>                                          Overall
#> AvgTimeDelta                            100.0000
#> Count_Unique_TransactionId               33.7933
#> Recency                                  19.8737
#> Total_Quantity                           16.2686
#> StDev_Value                               7.3689
#> Count_Unique_ItemId                       7.0082
#> Mean_Value_per_Unique_TransactionId       5.4638
#> Mean_Value_per_Unique_ItemId              5.4284
#> Total_Value                               4.2315
#> Mean_Quantity_per_Unique_TransactionId    3.4107
#> AddressE                                  3.3424
#> Mean_Quantity_per_Unique_ItemId           3.2781
#> StDev_Quantity                            1.8020
#> AddressC                                  1.3987
#> AgeJ                                      0.8408
#> AgeC                                      0.3751
#> AgeE                                      0.2329
#> AddressB                                  0.2147
#> AddressF                                  0.1489
#> AgeD                                      0.0000

fit$glm %>% predict(testing) %>% confusionMatrix(testing$churn)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction FALSE TRUE
#>      FALSE   336  172
#>      TRUE    372  849
#>
#>                Accuracy : 0.6854
#>                  95% CI : (0.6629, 0.7072)
#>     No Information Rate : 0.5905
#>     P-Value [Acc > NIR] : 2.359e-16
#>
#>                   Kappa : 0.32
#>  Mcnemar's Test P-Value : < 2.2e-16
```

```
#>
#>             Sensitivity : 0.4746
#>             Specificity : 0.8315
#>          Pos Pred Value : 0.6614
#>          Neg Pred Value : 0.6953
#>              Prevalence : 0.4095
#>          Detection Rate : 0.1943
#>    Detection Prevalence : 0.2938
#>       Balanced Accuracy : 0.6531
#>
#>        'Positive' Class : FALSE
#>
fit$logitboost %>% predict(testing) %>% confusionMatrix(testing$churn)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction FALSE TRUE
#>      FALSE   432  305
#>      TRUE    276  716
#>
#>                Accuracy : 0.664
#>                  95% CI : (0.6411, 0.6862)
#>     No Information Rate : 0.5905
#>     P-Value [Acc > NIR] : 1.989e-10
#>
#>                   Kappa : 0.3095
#>  Mcnemar's Test P-Value : 0.2454
#>
#>             Sensitivity : 0.6102
#>             Specificity : 0.7013
#>          Pos Pred Value : 0.5862
#>          Neg Pred Value : 0.7218
#>              Prevalence : 0.4095
#>          Detection Rate : 0.2499
#>    Detection Prevalence : 0.4263
#>       Balanced Accuracy : 0.6557
#>
#>        'Positive' Class : FALSE
#>
fit$xgbtree %>% predict(testing) %>% confusionMatrix(testing$churn)
#> Confusion Matrix and Statistics
#>
#>           Reference
#> Prediction FALSE TRUE
#>      FALSE   378  206
#>      TRUE    330  815
#>
#>                Accuracy : 0.69
#>                  95% CI : (0.6676, 0.7117)
#>     No Information Rate : 0.5905
#>     P-Value [Acc > NIR] : < 2.2e-16
#>
#>                   Kappa : 0.3413
#>  Mcnemar's Test P-Value : 1.08e-07
#>
#>             Sensitivity : 0.5339
#>             Specificity : 0.7982
#>          Pos Pred Value : 0.6473
#>          Neg Pred Value : 0.7118
#>              Prevalence : 0.4095
#>          Detection Rate : 0.2186
#>    Detection Prevalence : 0.3378
#>       Balanced Accuracy : 0.6661
#>
```

```
#>        'Positive' Class : FALSE
#>
```