7313 Take-Home Exam

In this take-home exam, I will be predicting online or offline purchases made at the Buy N Small (BnS) drugstore. As implied in the task, the selected aggregation-level will be purchases on the receipt-level. Note! With warning messages outputting the XGBoost model, the handin became 5 pages. I hope you can disregard theese and consider it a 4-page hand-in.

Importing from SQL

Let us start by importing the relevant data for this classification problem. The target variable *is_online* is on the correct receipt-level aggregation-level at import. The predictor variables *purchase_date*, *age*, *gender*, and *enrollment* are also by default on the correct aggregation level as the date and customer is same for all transactions under the same receipt. The predictor variables *tot_quantity*, *tot_amount* are aggregated through the SUM() function to be expressed on receipt-level. Finally, for a creative touch I include a sub-query which gets the department code of the department that receives the most revenue from the transactions in the receipt, thus on the correct aggregation level. This department code is stored in the predictor variable *main_dept*.

```
df = dbGetQuery(con,
  "SELECT receipt_id, customer_id, purchase_date, SUM(quantity) AS tot_quantity,
     SUM(amount) AS tot_amount, MAX(is_online) AS is_online,age, gender,
     enrollment, main_dept
  FROM Transactions t
    LEFT JOIN Unseen u USING (receipt_id)
    LEFT JOIN Customers c USING (customer_id)
    LEFT JOIN (
        SELECT receipt_id, dept as main_dept, max(revenue)
        FROM (
            select receipt_id, dept, sum(amount * quantity) as revenue
            FROM Transactions t
            LEFT JOIN Products p USING (item)
            GROUP BY receipt_id, dept
        ) dept_revenues
        GROUP BY receipt_id
    ) max_revenue USING (receipt_id)
  WHERE id23500 = 1 ##change to your studentid
    OR is_online IS NOT NULL
  GROUP BY receipt_id  ")
```

Data Preparation and Feature Transformation

Before presenting the final data set we do some cleaning and feature transformation. For our classification models to work, character variables are converted to factor variables. The variable *purchase_date* is binned into *purchase_month* to remove noise and capture the long-term trend in online shopping over time. The receipt's weekday is extracted from *purchase_date* in the variable *purchase_day*, to capture weekly cyclical patterns. A graphical motivation for these transformations of the *puchase_date* variable are shown in *Appendix A1*. Furthermore, to make the model less noisy we bin ages into generations of 5 years. This should smooth out the distribution of online receipts across age groups. A graphical motivation is found in *Appendix A2*. Due to space limitations to exact code on how these variables were transformed can be found in *Appendix A3*.

Before we proceed to the glimpse of the data, we check for missing values in the variables.

```
sapply(df, function(x) sum(is.na(x)))[sapply(df, function(x) sum(is.na(x))) != 0]
```

```
##      gender enrollment  main_dept generation
##           1          1          2          1
```

We see that there are missing values in the *generation*, *gender* and *enrollment* variables. These missing values are the result of a mismatch in *customer_id* during the joining of tables in SQL. With a full explanation in *Appendix A4*, I reformatted the faulty customer_id from date to numeric and found an existing customer. These correct customer details are updated to replace the missing values.For the missing values in *main_dept*, I wrote an function that imputes a *main_dept* category based on the mean *tot_amount* of each category. The mean *tot_amount* closest to the amounts in the receipts with missing *main_dept*, will determine which category is imputed.

```
# define function for imputing main_dept based on amount
impute_main_dept <- function(df, tot_amount) {
  amounts <- df %>% group_by(main_dept) %>% summarize(avg_tot_amount = mean(tot_amount)) %>%
  filter(!is.na(main_dept)) %>% arrange(desc(avg_tot_amount))
  output <- c()
  for (t in tot_amount) {
    diff_vector <- abs(t - amounts$avg_tot_amount)
    output <- c(output, as.numeric(as.character(amounts$main_dept[which.min(diff_vector)])))
  }
  return(output)
}
# call the impute_main_dept function on all receipts with NA for main_dept
df[is.na(df[,'main_dept']),'main_dept'] <- impute_main_dept(df, df[is.na(df[,'main_dept']),'tot_amount'
```

With necessary data cleaning, feature transformation, and imputations complete, I present a glimpse of the final dataset. (Note additional feature transformation will be done for the XGBoost model).

```
glimpse(df)
```

```
## Rows: 405,411
## Columns: 9
## $ purchase_month <fct> 2019-05, 2019-08, 2019-05, 2018-10, 2018-06, 2018-06, 2~
## $ purchase_day   <fct> Friday, Tuesday, Monday, Saturday, Friday, Saturday, We~
## $ tot_quantity   <dbl> 3, 1, 2, 1, 6, 2, 3, 3, 3, 2, 7, 2, 4, 3, 6, 3, 3, 1, 1~
## $ tot_amount     <dbl> 1430, 3790, 1000, 340, 4610, 650, 2260, 2160, 1350, 197~
## $ is_online      <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ gender         <fct> Female, Female, Female, Female, Female, Female, Female,~
## $ generation     <fct> 40, 55, 35, 60, 40, 35, 60, 25, 40, 60, 30, 60, 25, 40,~
## $ enrollment     <fct> Paperform, POS, Paperform, Special, Special, Special, P~
## $ main_dept      <fct> 343, 357, 343, 349, 341, 342, 348, 339, 342, 354, 339, ~
```

Model Evaluation

I will now proceed with training, testing, and evaluating different machine learning models to predict whether receipts are made online or offline. Due to space limitations I will only present two of the two best-performing models that I explored: random forests, and XGBoost. All models will be trained and tested on the same training (75%) and testing (25%) data sets. I also declare the target variable and predictors in a formula.

```
# use a uniform distribution to split the data into training and test sets.
set.seed(7313)
dist <- runif(nrow(df))
df_train <- df[dist < 0.75,]
df_test <- df[dist >= 0.75,]

target <- "is_online"
predictors <- c("purchase_month", "purchase_day", "tot_quantity", "tot_amount", "generation", "gender",
fmla <- as.formula(paste(target, "~", paste(predictors, collapse = " + ")))
```

Random Forest

I start fitting a random forest model on the training data.

```
# Fit random forest model
online_model_rf <- ranger(fmla, df_train, num.trees = 500,
                          respect.unordered.factors = "order",
                          seed = set.seed(7313))
```

```
## Growing trees.. Progress: 60%. Estimated remaining time: 20 seconds.
```

With the model trained I now evaluate it now on the test set. I compare it to the heuristic which would be predicting all zeros for *is_online* (as it is most likely with a mean of 0.07).

```
# Make predictions on the test set
df_test$pred <- predict(online_model_rf, df_test)$predictions

# Evaluate accuracy of predictions
df_test %>% summarize(
    total = length(is_online), correct = sum(is_online == pred),
    share_correct = (100 * correct / total),incorrect = sum(is_online != pred),
    share_incorrect = (100 * incorrect / total))
```

```
##    total correct share_correct incorrect share_incorrect
## 1 100970   94633      93.72388      6337        6.276122
```

```
# Evaluate accuracy of heuristic
df_test %>% summarize(
    total = length(is_online), correct = sum(is_online == 0),
    share_correct = (100 * correct / total), incorrect = sum(is_online != 0),
    share_incorrect = (100 * incorrect / total)
  )
```

```
##    total correct share_correct incorrect share_incorrect
## 1 100970   93590       92.6909      7380        7.309102
```

Comparing the model with the heuristic, we see that the model performs better than the heuristic with an accuracy of 93.72%, compared to 92.69. The difference between their accuracy and error rates is 1.03 percentage points.

XGBoost

For the XGBoost I make sure to one-hot encode all of the categorical/factor variables. The encoded training and test sets are called *df_train.treat* and *df_test.treat*, respectively. In training the model I run the XGBoost cross-validation to identify the best number of trees in the model, i.e. the number of trees that minimize the estimated out-of-sample learning error.To avoid overfitting I select the optimal number of trees from the cross-validation test sample.

```
cv <- xgb.cv(data = as.matrix(df_train.treat),
            label = as.numeric(as.character(df_train$is_online)),
            nrounds = 200, nfold = 5, objective = "binary:logistic",
            eta = 0.3, max_depth = 6, early_stopping_rounds = 10,
            verbose = 0 )
```

```
## [23:47:22] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
## [23:47:22] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
## [23:47:23] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
## [23:47:23] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
## [23:47:24] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
```

```
# Determine and print how many trees minimize training and test error
elog_summary <- cv$evaluation_log %>%
   summarize(ntrees.train = which.min(train_logloss_mean),   # find the index of min(train_rmse_mean)
             ntrees.test  = which.min(test_logloss_mean))   # find the index of min(test_rmse_mean)
(ntrees <- elog_summary['ntrees.test'][1,1])
```

```
## [1] 185
```

With the optimal number of trees determined from the cross-validation, I now run the XGBoost model and make predictions on the online/offline variable of the receipts.

```
online_model_xgb <- xgboost(data = as.matrix(df_train.treat), # training data as matrix
                   label = as.numeric(as.character(df_train$is_online)),
                   nrounds = ntrees, objective = "binary:logistic",
                   eta = 0.3, depth = 6, verbose = 0 )
```

```
## [23:52:44] WARNING: amalgamation/../src/learner.cc:576:
## Parameters: { "depth" } might not be used.
##
##   This could be a false alarm, with some parameters getting used by language bindings but
##   then being mistakenly passed down to XGBoost core, or some parameter actually being used
##   but getting flagged wrongly here. Please open an issue if you find any such cases.
##
##
## [23:52:45] WARNING: amalgamation/../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evalu
```

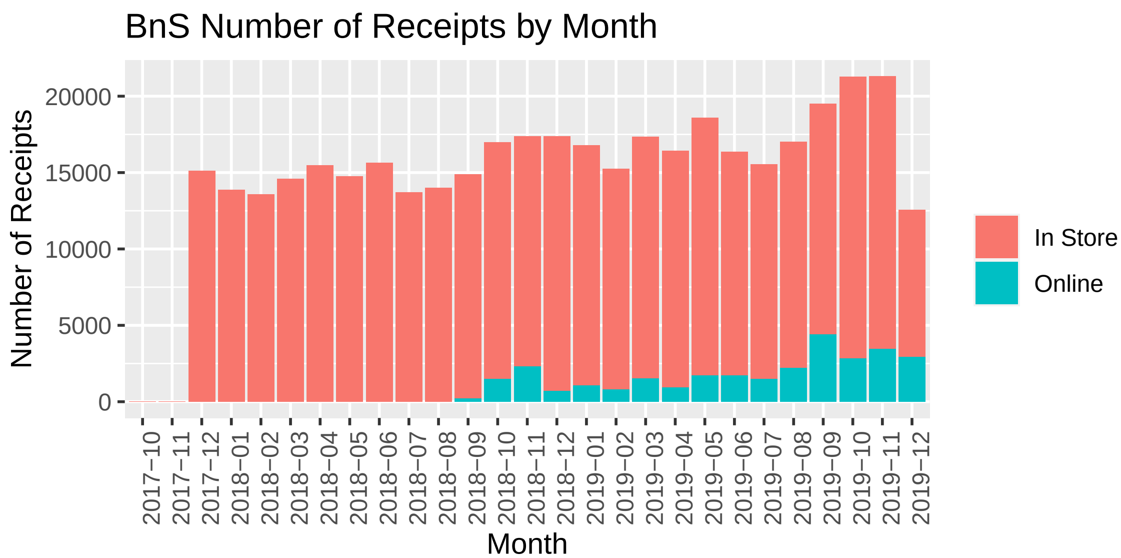
```r
# Make predictions
df_test$xgb_prob <- predict(online_model_xgb, as.matrix(df_test.treat))
```

The XGBoost model predicts a probability of receipts being online. To convert this to binary outcomes, I loop through thresholds to find the threshold which gives the highest accuracy. This is demonstrated in Appendix A5. In a similar comparison between model and heuristic (as done with random forest) we find that the model has an accuracy of 93.89%, which is 1.20 percentage points higher than the heuristic of 92.69%. As the XGBoost model performs better than the random forest I select it as the model for the unseen dataset (using Occam's razor principle of selecting simplest model given efficiency).

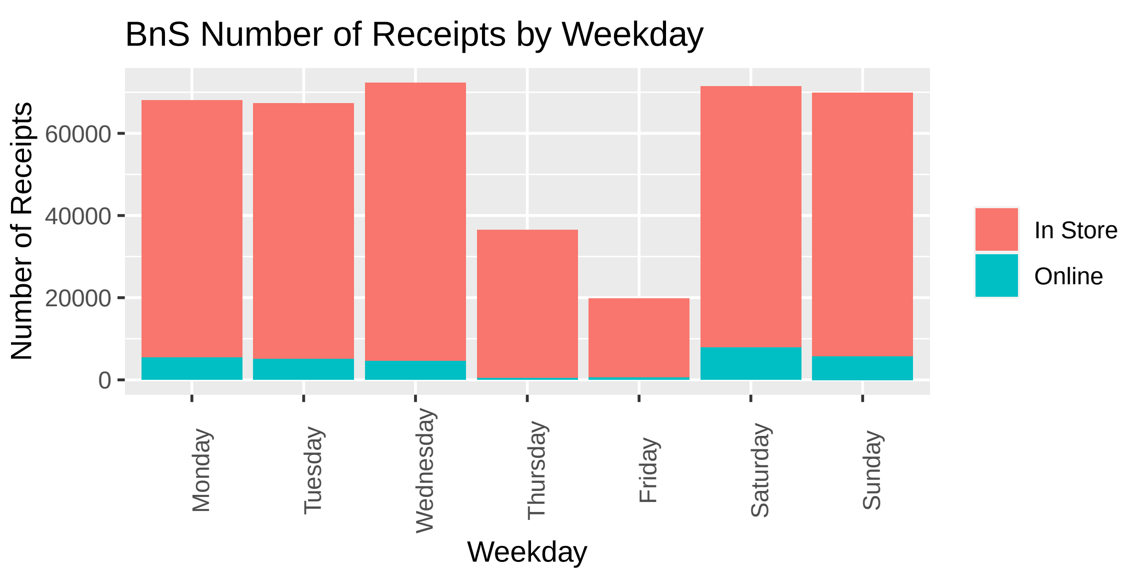
Appendix

A1. Splitting the purchase_date Variable

The share of receipts being online, varies over time. By splitting up the *puchase_date*, we can capture two different effects over time. When looking at the monthly development of online receipts, we see that in the months October 2017 - August 2018, there share of online receipts is negligible.After August 2018 we see an increase in online purchases. Knowing which month a purchase occured will be very valuable in predicting the online/offline status of the purchase.

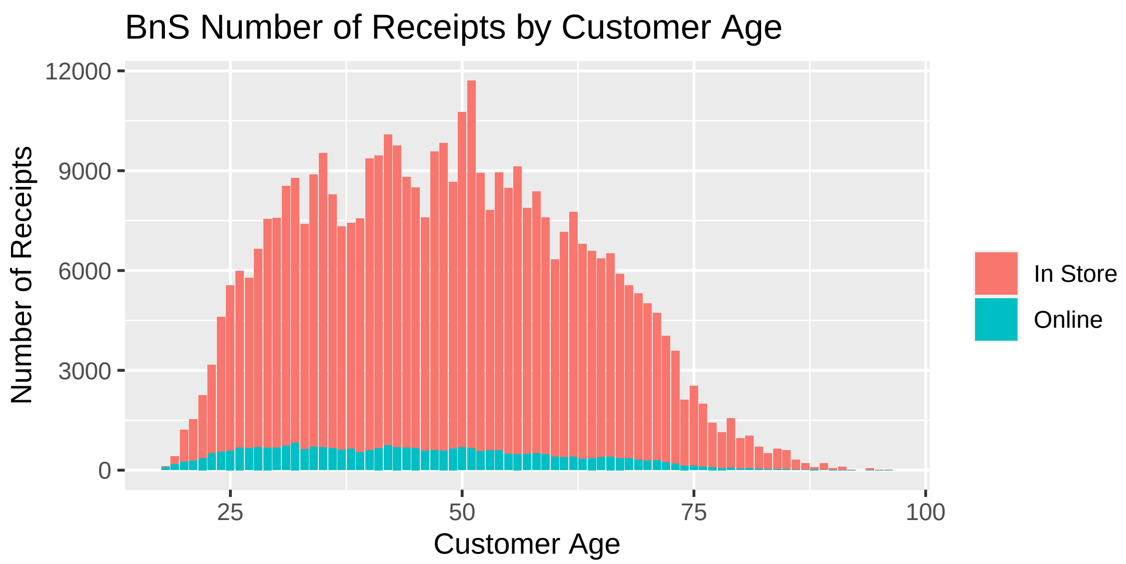


Furthermore we see that there is significant variation in the share of both purchases and online purchases between different weekdays. In the plot below, we see that Thursdays and Fridays show significantly fewer pruchases as a whole, but even more so there are disproportionately fewer online purchases being made on these days. The other days exhibit quite similar levels of purchases, with a higher share of online receipts on the weekend.

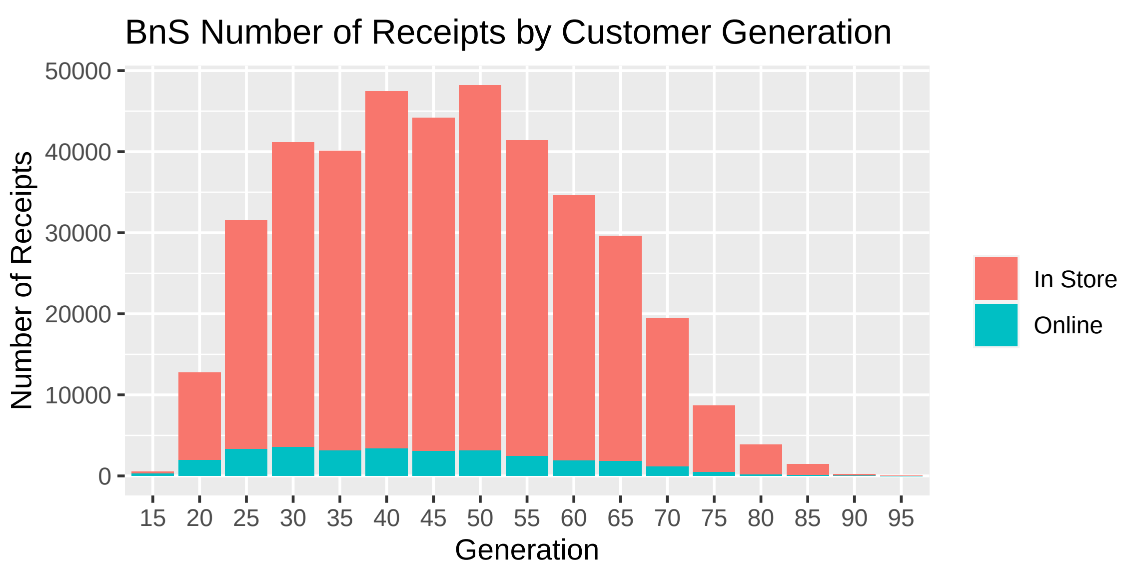


A2. Binning the age Variable into 5-year Generations

The distribution of receipts (online/offline) across customer ages is shown in the graph below. In the distribution we see that there are no customers between ages 0-10, and very few older than 80.



To smoothen out the distribution, and remove noise specific to the customers in the data set, I choose to bin the customers into fixed-width bins of 5 years. The group of 5 years will be represented by the first year in the group. These groups are stored in the *generation* variable.



A3.

```
df <- df_raw
```

```
# recode variables
df <- df %>%
  mutate(
    is_online = as.factor(is_online),
    purchase_date = as.Date(purchase_date,"%Y-%m-%d"),
    purchase_month = as.factor(format(purchase_date, "%Y-%m")),
    purchase_day = factor(format(purchase_date, "%A"),
                      levels = c("Monday", "Tuesday", "Wednesday", "Thursday",
                                 "Friday", "Saturday", "Sunday")),
    main_dept = as.factor(main_dept),
    enrollment = as.factor(enrollment),
    gender = factor(gender),
    generation = NA
    )
levels(df$gender) <- c("Female", "Male")


# group ages into generation bins of five years
generations <- seq(0, 100, 5)
for (g in generations) {
  df <- df %>%
  mutate (
    generation = ifelse(age >= g & age < (g+10), g, generation)
  )}
df$generation <- as.factor(df$generation)

# remove the age variables from the df
df <- df %>% select(-age)
```

A4. Updating Missing Values for Customer "01-nov-00"

In the data preparation there we found missing values for age, gender and enrollment for one receipt. The reason for this was that the receipt had a $customer_id$ of "01-nov-00". If we look at a summary of the purchase_date distribution, we see that the equivalen date of 2000-11-01 is a severe outlier compared to the data.

```
summary(df$purchase_date)
```

```
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2017-10-03" "2018-06-27" "2019-01-12" "2019-01-02" "2019-07-13" "2019-12-16"
```

This suggests that the *customer_id* of "01-nov-00" is not the accidental insert of a true *purchase_date*, but rather an error in variable type. If we look at the numeric equivalent of the date 2000-11-01, we find that a potential true *customer_id* could be 11262.

```
as.numeric(as.Date("01-nov-00", "%d-%b-%y"))
```

```
## [1] 11262
```

We can then check if this customer_id exists in the df. We present all receipts with *receipt_id* in the table below, along with the receipt with the missing values:

```
df %>%
  filter(
    (customer_id == as.numeric(as.Date("01-nov-00", "%d-%b-%y"))) | (customer_id == "01-nov-00")
  )
```

```
##        receipt_id customer_id purchase_date tot_quantity tot_amount is_online
## 1 314238412747203       11262    2019-07-23            1        500         0
## 2 379844308696576       11262    2018-11-06            2       1900         1
## 3 382324908696576   01-nov-00    2019-03-23            1       3590         1
## 4 682031087262690       11262    2018-08-30            1        590         0
## 5 701003549056052       11262    2018-10-04            1        750         0
## 6 710143849046631       11262    2019-04-25            3       1860         0
##   gender enrollment main_dept purchase_month purchase_day generation
## 1   Male        Web       354        2019-07      Tuesday         30
## 2   Male        Web       345        2018-11      Tuesday         30
## 3   <NA>       <NA>       357        2019-03     Saturday       <NA>
## 4   Male        Web       341        2018-08     Thursday         30
## 5   Male        Web       339        2018-10     Thursday         30
## 6   Male        Web       352        2019-04     Thursday         30
```

Seeing these receipts in the same table, reencoding "01-nov-00" to its numerical equivalent is justified as the *purchase_date*, *quantity*, *amount*, and *is_online* details fit in with the receipts of customer 11262. Thus we correct the missing values to a 30-35 year-old male with web enrollment.

While, this analysis took a bit more time, this process is arguable better than imputing. If we were to impute the categorical mode, then the missing values would take the values of the average customer to BnS, grouped by age, gender, and enrollment. Furthermore we would impute values into a seemingly invalid customer_id of "01-nov-00". The imputed values would then be a 40-45 year-old female with paperform enrollment.

```
summary <- df %>%
  group_by(gender, generation, enrollment) %>%
  summarize (
    receipt_count = length(receipt_id)
  ) %>%
  arrange(desc(receipt_count))
```

```
## `summarise()` has grouped output by 'gender', 'generation'. You can override using the `.groups` argu
```

```
summary[1:5,]
```

```
## # A tibble: 5 x 4
## # Groups:   gender, generation [5]
##   gender generation enrollment receipt_count
##   <fct>  <fct>      <fct>              <int>
## 1 Female 40         Paperform          17781
## 2 Female 30         POS                16279
## 3 Female 50         Paperform          15933
## 4 Female 45         Paperform          14999
## 5 Female 25         POS                14493
```

A5. Threshold computing for XGBoost and evaluation

```
accuracy <- c()
threshold <- seq(0.01, 1, 0.01)
for (t in threshold){
  df_test$temp_pred <- as.numeric(df_test$xgb_prob >= t)
  df_temp <- df_test %>%
  summarize(
    total = length(is_online),
    correct = sum(is_online == temp_pred),
    share_correct = (100 * correct / total),)
  accuracy <- c(accuracy, df_temp[1, 'share_correct'])
}


max_accuracy <- data.frame(threshold, accuracy) %>%
  arrange(desc(accuracy)) %>%
  filter(row_number()==1)

max_accuracy    # Evaluate accuracy of predictions    = 93.89%    (+1.20%)
```

```
##   threshold accuracy
## 1      0.47 93.89026
```

```
df_test$pred_xgb <- as.numeric(df_test$xgb_prob >= max_accuracy[1, 'threshold'])

# Evaluate accuracy of heuristic    = 92.69%
df_test %>%
  summarize(
    total = length(is_online),
    correct = sum(is_online == 0),
    share_correct = (100 * correct / total),
    incorrect = sum(is_online != 0),
    share_incorrect = (100 * incorrect / total)
  )
```

```
##    total correct share_correct incorrect share_incorrect
## 1 100970   93590       92.6909      7380        7.309102
```