Identification of Website Visitors Close to Purchase

Choose Your Own Project - Report

Katrin Heimerl

30 5 2021

Contents

1. Introduction

Today, we witness an ever growing importance of online shopping - regardless of whether it is clothing, books, electronics or even furniture or cars. The COVID-19 pandemic has even accelerated this growth with physical stores being forced to temporarily shut down. Since the past year, internet sales have been skyrocketing and it does not look like they are going to let up anytime soon. However, online retail is still a highly competitive market and only those who know their customers best will prevail. A decisive point is the targeted application of costly marketing efforts. Re-targeting previous website visitors is a good first step but the most cost-effective way is to target only those visitors with the highest chance of making a purchase.
This is where this report comes in, the goal of which is to develop a machine learning algorithm to predict if someone will order from a website based on their interaction with this website beforehand. For this purpose, the "Customer propensity to purchase" dataset which is openly available on kaggle and represents user interaction data from a fictional website will be used. The report is structured in the following chapters: Chapter 2 first prepares the data and lays the foundation of the modeling approach through data exploration. Chapter 3 shows the modeling results: Different models will be discussed and compared regarding their performance. Finally, chapter 4 concludes the report with a brief summary. In addition, it will give an outlook on limitations and future research approaches. This report is part of the final assignment of the edX Data Science Capstone course (HarvardX - PH125.9x).

2. Analysis

2.1. Data Import

As a first step, all packages necessary need to be installed resp. loaded.

This project uses the "Customer propensity to purchase dataset - A dataset logging shoppers interactions on an online store" accessed from: https://www.kaggle.com/benpowis/customer-propensity-to-purchase-data/metadata. The data originally comes as a testing and training sample, however, the provided testing sample will not be used as it does not contain any customers that have placed an order which makes it useless for this analysis. Instead, the sufficiently large provided training sample has been split into two datasets for training and testing purposes. The datasets can be downloaded from GitHub.

```
url_testing <- "https://raw.githubusercontent.com/KaHeimerl/ChooseYourOwn/main/training_sample_01.txt"
testing_sample <- read_csv(url(url_testing))
url_training <- "https://raw.githubusercontent.com/KaHeimerl/ChooseYourOwn/main/training_sample_02.txt"
training_sample <- read_csv(url(url_training))
rm(url_testing, url_training)
```

2.2. Data Exploration

The training data consists of 25 columns and roughly 238.000 rows representing unique users. The columns contain data about the unique user (e.g. ID, device used, location) and his resp. her website browsing behavior (e.g. clicks on specific areas, opening / seeing specific pages). The last column contains the outcome variable "ordered" which shows whether someone has placed an order (1) or not (0).

```
dim(training_sample)
```

```
## [1] 238027     25
```

```
head(training_sample)
```

```
## # A tibble: 6 x 25
##   UserID basket_icon_cli~ basket_add_list basket_add_deta~ sort_by image_picker
##   <chr>             <dbl>           <dbl>            <dbl>   <dbl>        <dbl>
## 1 25b5--~               0               0                0       0            0
## 2 1b07--~               1               0                1       0            0
## 3 0752--~               1               0                0       0            0
## 4 bb95--~               1               1                1       0            0
## 5 ab78--~               0               0                0       0            0
## 6 3b48--~               0               0                0       0            0
## # ... with 19 more variables: account_page_click <dbl>,
## #   promo_banner_click <dbl>, detail_wishlist_add <dbl>,
## #   list_size_dropdown <dbl>, closed_minibasket_click <dbl>,
## #   checked_delivery_detail <dbl>, checked_returns_detail <dbl>, sign_in <dbl>,
## #   saw_checkout <dbl>, saw_sizecharts <dbl>, saw_delivery <dbl>,
## #   saw_account_upgrade <dbl>, saw_homepage <dbl>, device_mobile <dbl>,
## #   device_computer <dbl>, device_tablet <dbl>, returning_user <dbl>,
## #   loc_uk <dbl>, ordered <dbl>
```

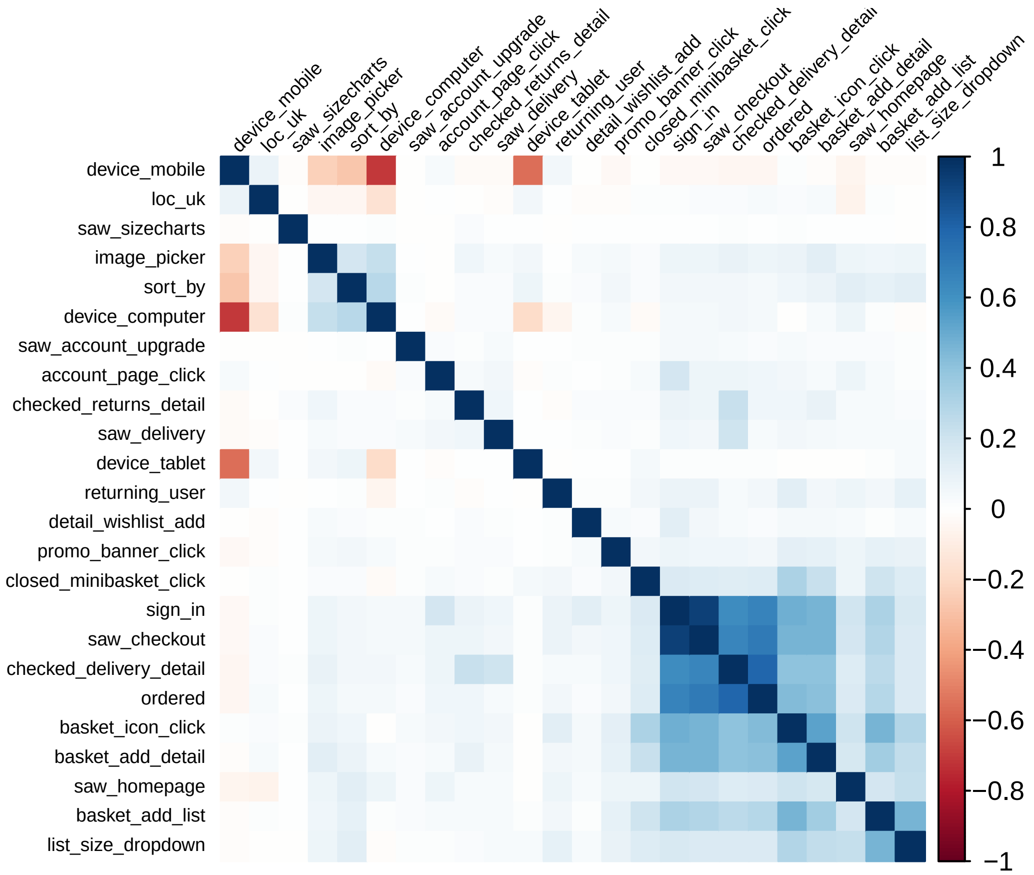
Approximately 4.2% of users in the sample have placed an order.

```
mean(training_sample$ordered)
```

```
## [1] 0.04228092
```

A correlation matrix shows which variables correlate with each other.

```
correlations <- cor(training_sample[-1])
corrplot(correlations, method="color", order="hclust", tl.col="black", tl.srt=45, tl.cex=0.6)
```



The blue squares in the lower right quadrant show that some variables correlate with the outcome variable “ordered”. Calculation shows that the highest correlations exist with “checked_delivery_detail”, “saw_checkout” and “sign_in”.

```
corr_ordered <- cor(training_sample[-1], training_sample$ordered)
print(corr_ordered[order(corr_ordered[,1],decreasing=TRUE),])
```

```
##                 ordered checked_delivery_detail            saw_checkout
##             1.000000000             0.799037468             0.709679387
##                 sign_in       basket_icon_click       basket_add_detail
##             0.666756646             0.432798451             0.419169656
##          basket_add_list            saw_homepage      list_size_dropdown
##             0.288477170             0.158228682             0.153508217
## closed_minibasket_click            image_picker      account_page_click
##             0.141694384             0.071762099             0.061360022
##  checked_returns_detail          returning_user      promo_banner_click
##             0.060831580             0.059260235             0.056492086
##                 sort_by         device_computer                  loc_uk
##             0.049665775             0.047085213             0.031523522
##            saw_delivery     detail_wishlist_add     saw_account_upgrade
##             0.031266246             0.025067672             0.023662211
##           device_tablet          saw_sizecharts           device_mobile
##             0.016816032             0.007633361            -0.040732350
```

So, if someone has checked the delivery details, got to the checkout page or has signed into their account, there is a high likelihood that an order got placed during that website visit. Noticeable correlations exist also with “basket_icon_click”, “basket_add_detail” and “basket_add_list”. If someone has added an item to their basket / cart, they are also quite likely to have placed an order. These variables will be used in further analysis. However, “saw_checkout” will not be considered as a user only gets to the checkout place when he resp. she has already ordered so this variable can not be used for prediction. All predictors are sufficiently present in the sample.

```
mean(training_sample$checked_delivery_detail)
```

```
## [1] 0.06338357
```

```
mean(training_sample$sign_in)
```

```
## [1] 0.08922517
```

```
mean(training_sample$basket_icon_click)
```

```
## [1] 0.09892995
```

```
mean(training_sample$basket_add_detail)
```

```
## [1] 0.1127519
```

```
mean(training_sample$basket_add_list)
```

```
## [1] 0.07440753
```

2.3. Modeling Approach

The exploration of the data has shown that there are several independent variables in the sample that correlate with the dependent variable “ordered”. These predictors can be used to build a model that predicts whether or not an order will be placed. The aim is to build and evaluate several models making use of the predictors and choose the model that performs best. Performance will be measured using the weighted harmonic average of sensitivity and specificity (F1-score) with a higher importance placed on sensitivity (beta=2). Taking into consideration that the outcome variable “ordered” has a rather low prevalence in the sample (4.2%), sensitivity needs to be ranked higher in order to ensure that not too much marketing budget will be spent on users who will not order. In order to test out several models without the need to touch the final testing sample reserved for validation, the training sample will be split into another test and train set. Following the logic that it is necessary to have as much data as possible to train but, at the same time, have a test set that is large enough to obtain stable results, 10% of the original training sample will be used for testing.

```
set.seed(1)
test_index <- createDataPartition(training_sample$ordered,
                                  times = 1, p = 0.1,
                                  list = FALSE)
test_set <- training_sample[test_index, ]
train_set <- training_sample[-test_index, ]
```

3. Results

3.1. Model 1: Guessing the Outcome

The simplest model possible would be to just guess who will place an order and who will not regardless of any other website behavior.

```
set.seed(1)
y_guessing <- sample(c(0,1), length(test_index), replace = TRUE)
mean(y_guessing == test_set$ordered)
```

```
## [1] 0.5050204
```

By just guessing the outcome, approx. 50% of cases are predicted correctly. This might even be improved when taking into account the low prevalence of orders in the sample.

```
set.seed(1)
y_guessing_prob <- sample(c(0,1), length(test_index), replace = TRUE, prob=c(0.95,0.05))
mean(y_guessing_prob == test_set$ordered)
```

```
## [1] 0.9131202
```

Now, more than 90% of cases are correctly predicted. This is a very high value for just guessing, however, only those who do not order (0) are correctly identified - not necessarily those who do order (1).

```
table(predicted = y_guessing_prob, actual = test_set$ordered)
```

```
##          actual
## predicted     0     1
##         0 21681   923
##         1  1145    54
```

The table shows that indeed more than 20.000 users who have not ordered were correctly identified - but less than 100 who did. Applying this model would result in a waste of marketing budget. Accordingly, the confusion matrix shows a really low sensitivity and a balanced accuracy of only 50%.

```
confusionMatrix_guessing <- confusionMatrix(data = as.factor(y_guessing_prob),
                                            reference = as.factor(test_set$ordered), positive = "1")
print(confusionMatrix_guessing)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 21681   923
##          1  1145    54
##
##                Accuracy : 0.9131
##                  95% CI : (0.9095, 0.9167)
##     No Information Rate : 0.959
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0046
##
##  Mcnemar's Test P-Value : 1.175e-06
##
##             Sensitivity : 0.055271
##             Specificity : 0.949838
##          Pos Pred Value : 0.045038
##          Neg Pred Value : 0.959167
##              Prevalence : 0.041045
##          Detection Rate : 0.002269
##    Detection Prevalence : 0.050372
##       Balanced Accuracy : 0.502555
##
##        'Positive' Class : 1
##
```

The questions is if prevalence will be similarly low in other datasets. As expected, the F1-score is also pretty low.

```
fmeas_guessing <- F_meas(data = as.factor(y_guessing_prob),
reference = as.factor(test_set$ordered), relevant = "1", beta = 2)
print(fmeas_guessing)
```

```
## [1] 0.05286861
```

In order to store and compare all modeling results, an overall table is created.

3.2. Model 2: Logistic Regression

As a second, more advanced model, logistic regression is applied. Logistic regression is a common generalized linear model (GLM) that takes the possible outcomes of the dependent variable (0, 1) and finds a function that considers the linear combination of the predictors, thereby making use of logistic transformation. At this point, the two variables with the highest correlation ("checked_delivery_detail", "sign_in") will be considered as predictors.

```
fit_log_regression <- glm(ordered ~ sign_in + checked_delivery_detail, data = train_set,
                          family = "binomial")
prob_log_regression <- predict(fit_log_regression, test_set, type = "response")
table(prob_log_regression)
y_log_regression <- ifelse(prob_log_regression > 0.11, 1, 0)
```

The confusion matrix shows highly improved results. Balanced accuracy is up to 98.9% with less than 1% incorrectly predicted cases.

```
confusionMatrix_log_regression <- confusionMatrix(data = as.factor(y_log_regression),
                                                  reference = as.factor(test_set$ordered),
                                                  positive = "1")
print(confusionMatrix_log_regression)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 22640    12
##          1   186   965
##
##                Accuracy : 0.9917
##                  95% CI : (0.9904, 0.9928)
##     No Information Rate : 0.959
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9026
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.98772
##             Specificity : 0.99185
##          Pos Pred Value : 0.83840
##          Neg Pred Value : 0.99947
##              Prevalence : 0.04105
##          Detection Rate : 0.04054
##    Detection Prevalence : 0.04836
##       Balanced Accuracy : 0.98978
##
##        'Positive' Class : 1
##
```

It is already clear that it will be difficult to outperform this approach. The F1-score is 0.95.

```r
fmeas_log_regression <- F_meas(data = as.factor(y_log_regression),
                               reference = as.factor(test_set$ordered), relevant = "1", beta = 2)
print(fmeas_log_regression)
```

```
## [1] 0.9537458
```

3.3. Model 3: Linear Discriminant Analysis (LDA)

Although the results for logistic regression are already hard to beat, as a third model, Linear Discriminant Analysis (LDA) will be applied. Logistic regression can become unstable if the two classes that get predicted (0, 1) are well separated or there are few examples from which to estimate the parameters. LDA does address these shortcomings. Again, the two variables with the highest correlation ("checked_delivery_detail", "sign_in") will be used as predictors.

```r
fit_lda <- train(as.factor(ordered) ~ checked_delivery_detail + sign_in, method = "lda",
                 data = train_set)
y_lda <- predict(fit_lda, test_set)
```

The confusion matrix shows that lda does not perform as well as logistic regression. While there are slightly less 1s that are predicted as 0s, a some more 0s are predicted as 1s. Balanced accuracy is also minimally lower.

```r
confusionMatrix_lda <- confusionMatrix(data = y_lda,
                                       reference = as.factor(test_set$ordered), positive = "1")
print(confusionMatrix_lda)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 22290     7
##          1   536   970
##
##                Accuracy : 0.9772
##                  95% CI : (0.9752, 0.979)
##     No Information Rate : 0.959
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7699
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
##             Sensitivity : 0.99284
##             Specificity : 0.97652
##          Pos Pred Value : 0.64409
##          Neg Pred Value : 0.99969
##              Prevalence : 0.04105
##          Detection Rate : 0.04075
##    Detection Prevalence : 0.06327
##       Balanced Accuracy : 0.98468
##
##        'Positive' Class : 1
##
```

The F1-score is also a bit lower.

```r
fmeas_lda <- F_meas(data = as.factor(y_lda),
reference = as.factor(test_set$ordered), relevant = "1", beta = 2)
print(fmeas_lda)
```

```
## [1] 0.8958256
```

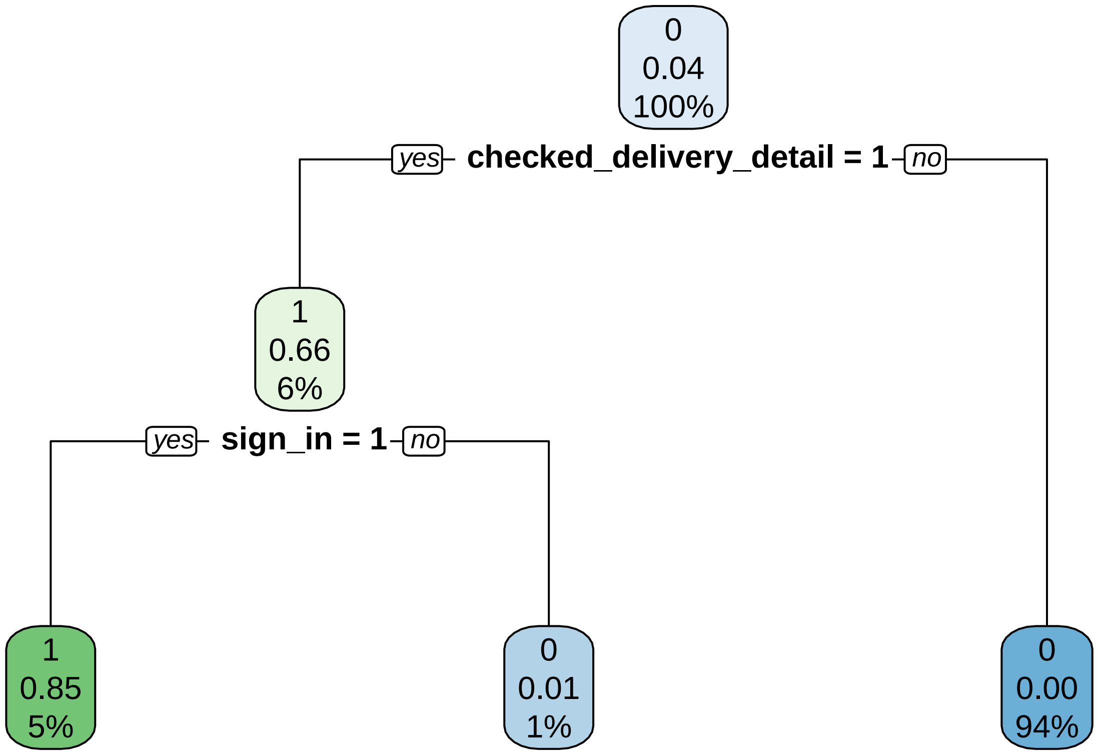
3.4. Model 4: Classification Tree

As a fourth approach, a classification tree will be applied. This model simulates a binary tree. Each root node represents a predictor variable and a split point on that variable. The leaf nodes contain the outcome variable. Again, the two variables with the highest correlation are used as predictors.

```r
fit_tree <- rpart(ordered ~ checked_delivery_detail + sign_in, data = train_set,
                  method = 'class')
y_tree <- predict(fit_tree, test_set, type = 'class')
```

Visualized, the classification tree looks like this:

```r
rpart.plot(fit_tree,
           extra = 106,
           yesno = 2,
           xflip = TRUE)
```



It is already apparent that, based on the decision rule chosen for the logistic regression approach above, the classification tree and logistic regression will yield the same results. The advantage is, however, that the classification tree automatically chooses the best decision rule. The confusion matrix shows the same values as for logistic regression.

```
confusionMatrix_tree <- confusionMatrix(data = as.factor(y_tree),
                                        reference = as.factor(test_set$ordered), positive = "1")
print(confusionMatrix_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 22640    12
##          1   186   965
##
##                Accuracy : 0.9917
##                  95% CI : (0.9904, 0.9928)
##     No Information Rate : 0.959
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9026
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.98772
##             Specificity : 0.99185
##          Pos Pred Value : 0.83840
##          Neg Pred Value : 0.99947
##              Prevalence : 0.04105
##          Detection Rate : 0.04054
##    Detection Prevalence : 0.04836
##       Balanced Accuracy : 0.98978
##
##        'Positive' Class : 1
##
```

The F1-score is also the same.

```
fmeas_tree <- F_meas(data = as.factor(y_tree), reference = as.factor(test_set$ordered),
                     relevant = "1", beta = 2)
print(fmeas_tree)
```

```
## [1] 0.9537458
```

3.5. Model 5: Random Forest

Classification trees are a precursor for random forest models. While a classification tree combines several decisions, a random forest combines several, randomly created decision trees. A characteristic of the random forest is to include all possible predictor variables and ensure randomness by randomly selecting sets of variables for prediction. Thus, a new training dataset will be created only including the variables supposed to function as predictors (i.e. variables relating to website behavior).

```
train_set_forest <- subset(train_set,
                           select = -c(UserID, saw_checkout, device_mobile, device_computer,
                                       device_tablet, returning_user, loc_uk))
```

Then, the random forest will be computed.

```
memory.limit(9999999999)
fit_forest <- randomForest(as.factor(ordered) ~ ., data = train_set_forest)
y_forest <- predict(fit_forest, test_set)
```

The confusion matrix shows very good results. The balanced accuracy is slightly lower than for logistic regression or the classification tree because the sensitivity is minimally worse.

```
confusionMatrix_forest <- confusionMatrix(data = as.factor(y_forest),
                                          reference = as.factor(test_set$ordered), positive = "1")
print(confusionMatrix_forest)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 22659    23
##          1   167   954
##
##                Accuracy : 0.992
##                  95% CI : (0.9908, 0.9931)
##     No Information Rate : 0.959
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9053
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.97646
##             Specificity : 0.99268
##          Pos Pred Value : 0.85103
##          Neg Pred Value : 0.99899
##              Prevalence : 0.04105
##          Detection Rate : 0.04008
##    Detection Prevalence : 0.04709
##       Balanced Accuracy : 0.98457
##
##        'Positive' Class : 1
##
```

This also effects the F1-score which is fractionally lower than for logistic regression resp. the classification tree.

```r
fmeas_forest <- F_meas(data = as.factor(y_forest),
                       reference = as.factor(test_set$ordered), relevant = "1", beta = 2)
print(fmeas_forest)
```

```
## [1] 0.9484987
```

```r
fmeas_results <- bind_rows(fmeas_results,
data.frame(method = "random forest", F1 = fmeas_forest))
```

A look at the variable importance confirms that the already identified variables play the most important role in determining whether someone will place an order or not. These are predominantly "checked_delivery_detail" and "sign_in" but also "basket_icon_click", "basket_add_detail" and "basket_add_list" are of relevance.

```r
variable_importance <- importance(fit_forest)
variable_importance[order(variable_importance[,1], decreasing = TRUE),]
```

```
## checked_delivery_detail                 sign_in       basket_add_detail
##             7780.065501             4049.923784              961.804388
##       basket_icon_click         basket_add_list            saw_delivery
##              868.833667              314.882289              150.177784
##            saw_homepage  checked_returns_detail      list_size_dropdown
##              103.672978               86.807395               64.616945
## closed_minibasket_click     detail_wishlist_add      account_page_click
##               37.245278               26.197849               24.076205
##            image_picker      promo_banner_click                 sort_by
##               18.821051               16.355515               15.078542
##     saw_account_upgrade          saw_sizecharts
##                9.426268                2.277312
```

The next step is to find out whether the best performing model will even improve by adding these three variables.

```r
print(fmeas_results)
```

```
##                method         F1
## 1            guessing 0.05286861
## 2 logistic regression 0.95374580
## 3                 lda 0.89582564
## 4 classification tree 0.95374580
## 5       random forest 0.94849871
```

So far, the best performing models with the highest F1-score are logistic regression and the classification tree.

Since the classification tree has the advantage of automatically finding the best cutoff value, this model will be chosen going forward.

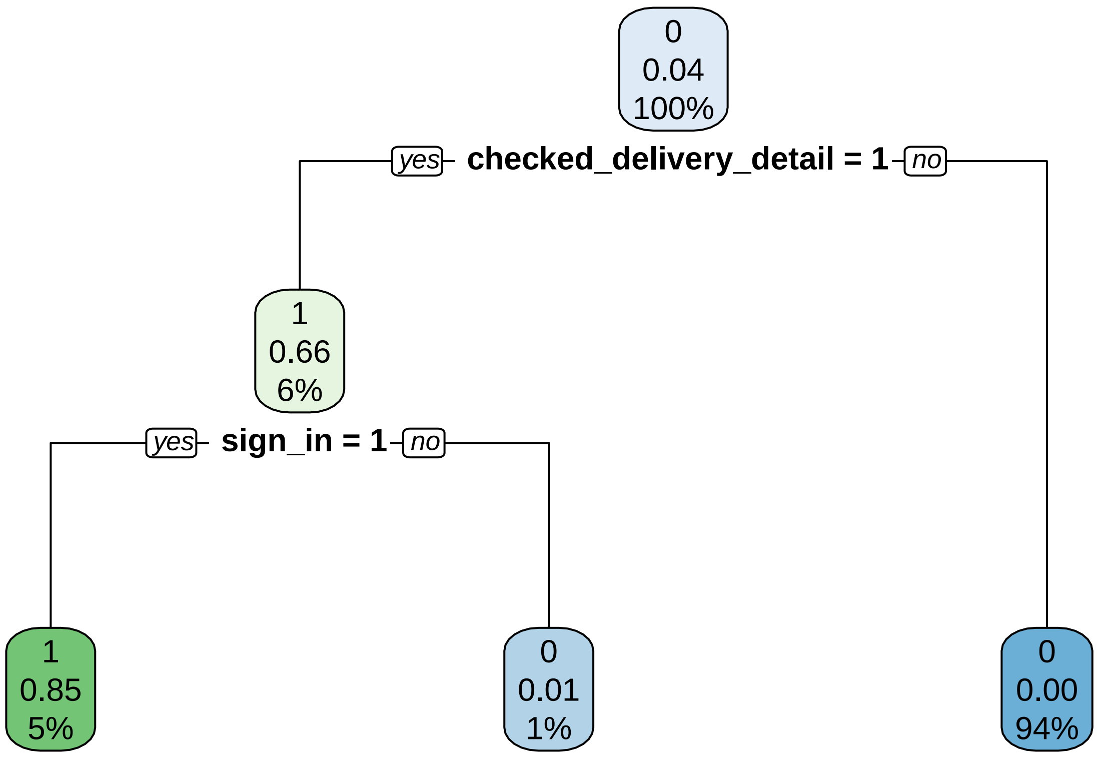
3.6. Model 6: Classification Tree with 5 Predictors

The three additional variables ("basket_add_detail", "basket_icon_click", "basket_add_list") will be added to the computation.

```
fit_tree_5 <- rpart(ordered ~ checked_delivery_detail + sign_in + basket_add_detail +
                      basket_icon_click + basket_add_list, data = train_set, method = 'class')
```

The plotted classification tree does not include the additional variables, indicating that adding them does not further improve the model.

```
rpart.plot(fit_tree_5,
           extra = 106,
           yesno = 2,
           xflip = TRUE)
```



This is also confirmed by the confusion matrix and the accompanying F1-score.

```
y_tree_5 <- predict(fit_tree_5, test_set, type = 'class')
confusionMatrix_tree_5 <- confusionMatrix(data = as.factor(y_tree_5),
                                          reference = as.factor(test_set$ordered), positive = "1")
print(confusionMatrix_tree_5)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction     0     1
##          0 22640    12
##          1   186   965
##
##                Accuracy : 0.9917
##                  95% CI : (0.9904, 0.9928)
##     No Information Rate : 0.959
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9026
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.98772
##             Specificity : 0.99185
##          Pos Pred Value : 0.83840
##          Neg Pred Value : 0.99947
##              Prevalence : 0.04105
##          Detection Rate : 0.04054
##    Detection Prevalence : 0.04836
##       Balanced Accuracy : 0.98978
##
##        'Positive' Class : 1
##
```

```r
fmeas_tree_5 <- F_meas(data = as.factor(y_tree_5),
reference = as.factor(test_set$ordered), relevant = "1", beta = 2)
print(fmeas_tree_5)
```

```
## [1] 0.9537458
```

The F1-score does not change by adding the three more variables to the model. It is not further improved.

3.7. Model 7: Classification Tree Applied to Testing Sample

Finally, as a last step, the model will be validated by applying it to the testing sample.

```r
fit_tree_fin <- rpart(ordered ~ checked_delivery_detail + sign_in, data = training_sample,
                      method = 'class')
y_tree_fin <- predict(fit_tree_fin, testing_sample, type = 'class')
```

The confusion matrix still shows very good results when being applied to the testing sample. Balanced accuracy is only fractionally lower.

```r
confusionMatrix_tree_fin <- confusionMatrix(data = as.factor(y_tree_fin),
                                            reference = as.factor(testing_sample$ordered),
                                            positive = "1")
print(confusionMatrix_tree_fin)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction      0      1
##          0 206811    142
##          1   1534   8887
##
##                Accuracy : 0.9923
##                  95% CI : (0.9919, 0.9927)
##     No Information Rate : 0.9585
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9098
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.98427
##             Specificity : 0.99264
##          Pos Pred Value : 0.85280
##          Neg Pred Value : 0.99931
##              Prevalence : 0.04154
##          Detection Rate : 0.04088
##    Detection Prevalence : 0.04794
##       Balanced Accuracy : 0.98846
##
##        'Positive' Class : 1
##
```

F1-score is even minimally better than for the training model.

```r
fmeas_tree_fin <- F_meas(data = as.factor(y_tree_fin),
reference = as.factor(testing_sample$ordered), relevant = "1", beta = 2)
print(fmeas_tree_fin)
```

```
## [1] 0.9548316
```

4. Conclusion

This report has applied several different models (logistic regression, lda, classification tree, random forest) to the “Customer propensity to purchase” dataset in order to develop a machine learning algorithm that helps to identify website visitors with a high inclination to purchase for re-targeting purposes. Two variables were decisive to identify those users with a high probability to purchase: “checked_delivery_detail” and “sign_in”. Using these two variables, logistic regression and classification tree have shown the best results with a balanced accuracy of 0.99 and an F1-score of 0.95.

Throughout this report, almost unrealistically good results were produced regardless of which model was applied. This must be attributed in large part to the specific dataset against which the algorithm was developed. Not much is known about the context in which these data were collected, on kaggle.com it only states that “This data set represents a day’s worth of visit to a fictional website.”. It is to be expected that user behavior in an online shop can vary greatly depending on which products are sold there. Purchase decisions on low-priced items (e.g. groceries, clothing) are made more spontaneously and with less prior research than on high-priced items (e.g. electronics, furniture). Consequently, some online stores might also have a higher or lower order rate depending on their merchandise. Plus, different information may be required before purchase (e.g. sizing info, delivery details, images). Thus, going forward, it would be interesting to apply this algorithm to multiple datasets representing different kinds of web shops and see how results would vary. The main challenge for this algorithm is to define the context in which it produces the best results and then to refine it within this specific context as there probably is not one-size-fits-all solution for all types of online purchasing behavior.