7313 Assignment 5: Deployment

Contents

Summary:

See below for an overview, followed by actionable recommendations and helpful visualizations.

Business Use Case Overview: Throughout this Business Use Case, we have tried to predict whether a customer will churn: That is, given a customer's first purchase characteristics, will that customer make only one visit to the store, or are they likely to return? After training several models on customer and purchase characteristics, we are able to determine with **75%** from that *one* purchase whether a customer will return or not. Therefore, using this model, we can make several actionable recommendations with estimated value impact (with associated assumptions of costs).

Target: *is_churn*: "1 if the customer did NOT return, 0 if the customer did return"

Features: as we defined in "Exploratory Data Analysis" Lab:

- *home_delivery*: Binary variable (1 if home delivery, 0 if collected in store)
- *num_items_sold*: The number of items included in the order
- *sales*: The total purchase amount in SEK
- *shipping_cost*: How much the customer paid for shipping in SEK
- *discount*: If a discount was applied to order

Furthermore, **Principal Component Analysis (PCA)** was applied to the features in an attempt to root out unhelpful correlations between the features and find a lower dimensional representation of the data (and hopefully feature selection). However, the PCA applied below led to very limited variable selection: only the very last two principle components were cut, as the share of variation in the data explained by every subsequent PC incorporated was roughly even (as seen in the linear cumulative explained variance plot below); thus it was fruitless to narrow down variables much more than that. (We keep the first 55 principal components, as the last two PCs explain relatively smaller amounts of variance - about a third as much)

Models Evaluated:

The following models were fitted with the specification of "is_churn ~." with the above features, both in "raw" format, and after being transformed by PCA. Every model has an identical train control object that does a 5-fold cross validation and reports other metrics. The models we trained (using ROC) are:

- LDA (with PCA)
- QDA (with PCA)
- Random Forest (with PCA)

We select the **Random Forest** model based on its accuracy, ROC, and especially Sensitivity. We can see how accuracy changed between Training and Test data sets in this graphic:

Metrics-wise, we prioritize *Sensitivity* (or True Positive Rate, TPR), as we want to "capture" as many potential churners ($is\_churn = 1$) as possible, and perhaps are not worried about False Positives at this point – as we see below, we will propose "cheap" methods that don't incur much cost per-customer, and therefore False Positives are not as concerning. (If we were to do a different approach for our final recommendation, we may reconsider which metric to use)

Final Model

Based on its having the best accuracy, as well as the best ROC (balancing true positives and negatives) in our model evaluation, we choose the **Random Forest** model, using PCA-treated features.

Estimated value impact

Finally, we can evaluate the *initial* estimated impact of our model, making two unrealistic simplifying assumptions:

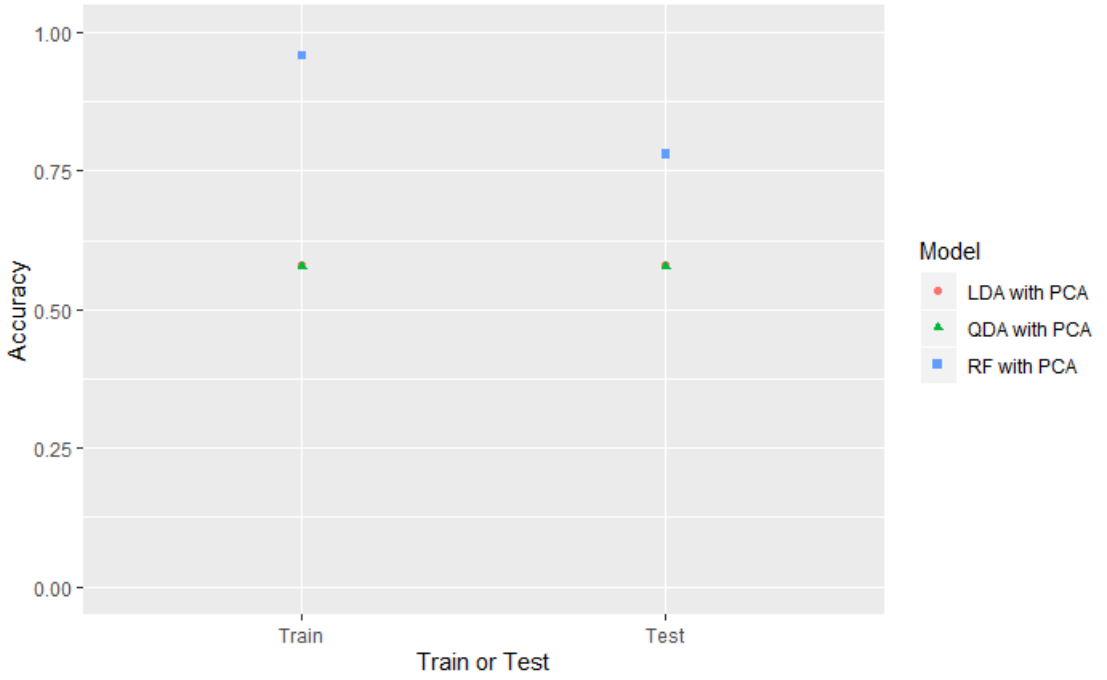

Figure 1: Train and Test Accuracies of Three Models

- The cost of our efforts is 0 SEK (reaching out to churners, bidding them to return to the store, etc)
- Our efforts are 100% successful

As we can see in the confusion matrix for the Random Forest model (in Appendix), when our model predicts "churn", it is correct ~70% of the time (the *Positive Pred Value* value, or Correct # predicted churn / Total predicted churn). So out of 1000 new customers that are predicted as "churn", ~700 can be expected to actually churn.

From the database we calculated the sum of sales and the average sales per customer (of ALL sales, not just the first stale), grouped by those customers that churn vs those that are loyal (in Appendix). Therefore, for every customer we can retain, we expect on average to make **212 SEK** more (the average loyal customer has average sales of 255 SEK while the average churner has average sales of 43 SEK). Therefore, if retention costs are below 212 SEK per customer retained, we can expect a positive net result. For the following value impact, though, we'll assume it costs nothing to retain a customer, and retaining an identified churner is 100% certain (This is NOT realistic, adjust later):

The "Maximum" total expected value of our findings, assuming we reach out to EVERYONE we suspect as a churner and successfully retain them at no cose, then, can be calculated by taking the number of correctly predicted churners (~50,000) and multiplying by the difference in average sales (between churners and loyal customers), or 212 SEK. Using this method, we get ~50,000*212 = 10.6 million SEK.

A more marginal (and thus applicable) estimate of value impact: For every 1000 predicted churners that we approach (and successfuly retain at no cost), we can assume 700 are actually would-be churners, which, if we retain, will net us 700*212 SEK, or 148,400 SEK.

However, to be more realistic, we can multiply our estimate by a probability of success (we retain X% of customers we approach, e.g.) and also subtracting out marginal costs for the effort of reaching each predicted churner. We do this below, for one of our recommendations:

Actionable Final Recommendations:

Then, for our final recommendations, we will take a "marginal" approach – reaching out to individual "predicted churn" customers as they finish their payments, with treatments that can be easily applied and modified.

Note that, if we identify a churning customer with a probability above a certain-enough threshold, then we assume that (since that customer was never going to come back anyway) any future profits made from the churning customer is pure net gain, as long as the costs of getting them to return are minimal.

Policy recommendations include:

- (for online purchases:) if the customer is predicted to churn based on our model, a post-checkout "coupon" for future purchases can be provided. Because this coupon will apply to future purchases, if we choose a high enough "certainty" threshold, this treatment will be costless to enact (assuming the coupon doesn't allow them to purchase future toys at costs below their prices).
- (for in-store purchases:) if the customer's purchase triggers the model/algorithm for being a "churn-likely" one, the cashier can give out a "Lucky Winner Coupon" for future in-store purchases. The customer will feel grateful for their "win" and be more likely to return.
- for any type of customer, we can offer *loyalty programs*, wherein every x number of store visits or total purchase benchmarks are rewarded, with a cost-effective discount or a promotional free item, for instance
- offer discounts with email sign-up: this recommendation would allow for future (automated) messages and continued contact with the customer
- say "Thank-you" and re-focus the customer-service aspect of the business to increase support/loyalty. Though it might be more difficult to quantify, positive customer service can be a powerful tool to foster loyalty

Because we will choose "churn" customers based on *high confidence* (high probability threshold) churning predictions, both of the first two "percentage"-approaches are virtually costless to enact. The rest are minimal cost, as well. We can update these policies as we gather more data and determine their costs and effectiveness. For one (simplisitic) example of estimated impact, modifying our costs with more reasonable assumptions:

- assume a coupon is free to implement, but takes out 50 sek from all of that customer's future purchases
- a customer who will return will spend 212 SEK more
- the coupon treatment causes a customer to return with 80% probability
- we have a 70% positive prediction rate in our final model
- Therefore: for every 1000 customers we offer the coupon, we net: 700*(212-50)*0.8 SEK, or about 90,000 SEK (but then, perhaps subtract out the 300 already-returning customers who now will pay 50 SEK less on future purchases, so -15,000 SEK)

We can update these estimates as we get more and better data. Which bring us to the final recommendation:

Finally, we would approach the business(es) and collect/ask for more data about churning customers – maybe those using online are frustrated with the user interface? Perhaps we should ask for more feedback in order to be able to make more relevant actionable recommendations.

Visualizations

First, and most importantly, we can highlight how *is_churn*, as we define it in our limited dataset, has an inherent problem: Customers who enter the store for the frist time in later months have fewer chances to return to the store, and are therefore more likely to churn, when they may in fact be as "loyal" as earlier-month customers. This is a large shortcoming in our target variable, as we have defined it.

Next, we can see that each country is equally likely to have churners:

We can see from a density plot that churning customers are much more likely to see lower profits, as the loyal customers are distributed more to the right on this graph (with a very long tail, for highly profitable returning customers):

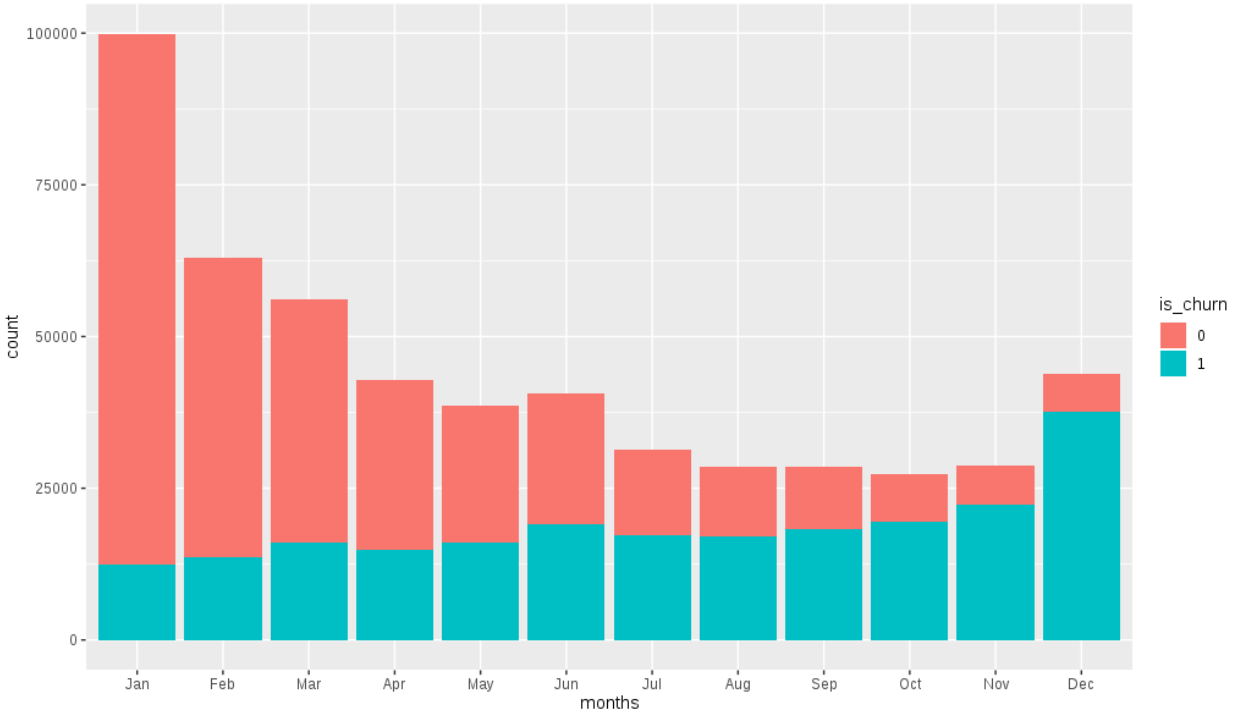

Figure 2: Churn by Month

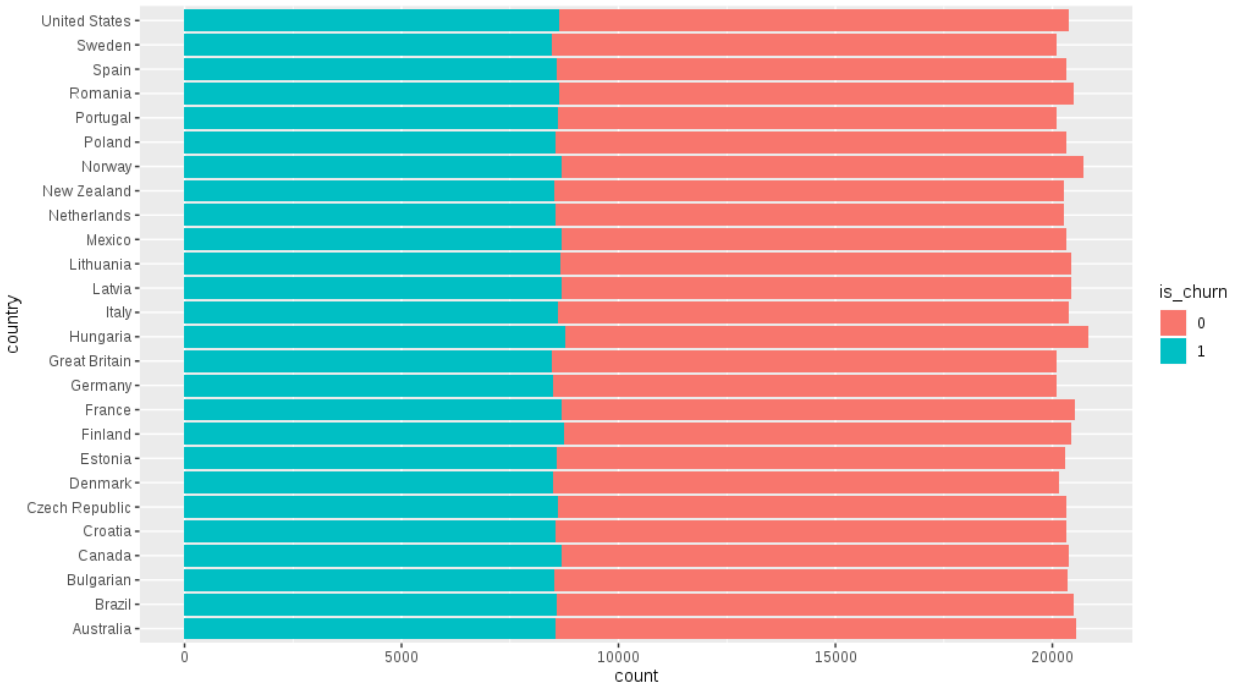

Figure 3: Churn by Country

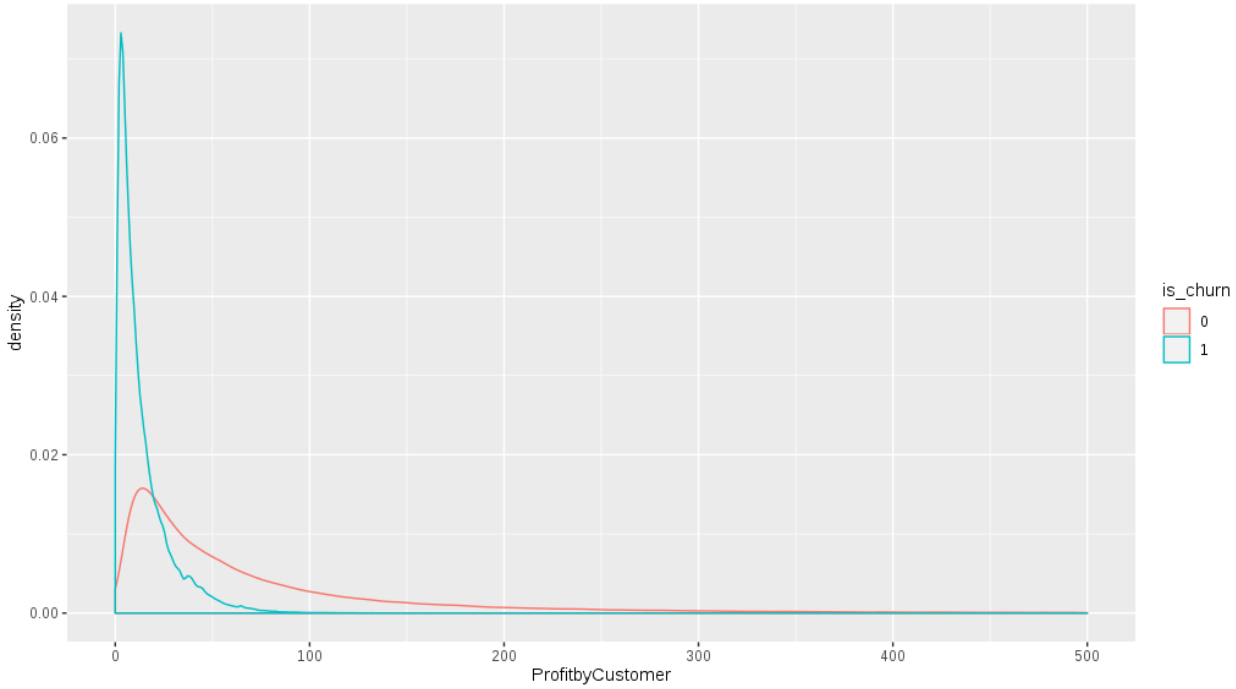

Figure 4: Churn Density of Profit

A box plot visualized roughly the same thing, with clearer depictions of the medians and interquartile ranges for churning and non-churning customers. We can see from this plot that the profit amount (total price - cost for that customer) between churn and non-churn customers is distinct: the 75th percentile for profits from a churning customer is around the 25th percentile of profits for a non-churning customer. Again, we can see the long tail for high-profit non-churn customers.

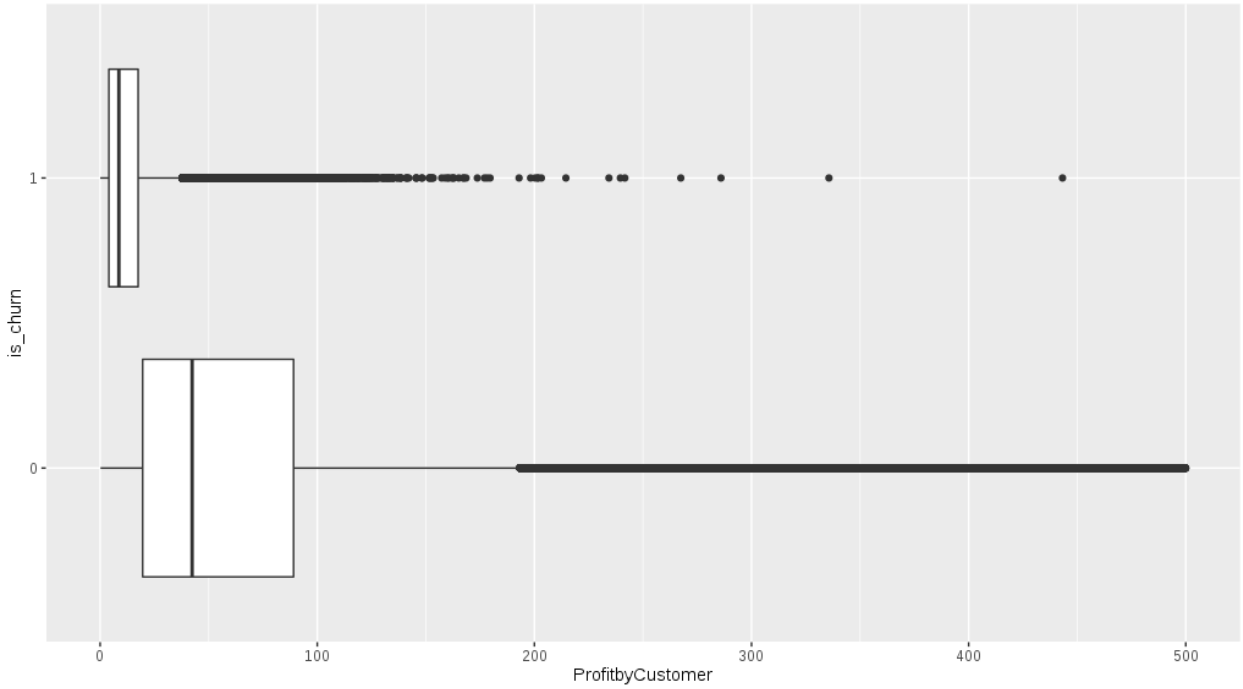

Figure 5: Churn Boxplots of Customer-level Profits

Data Preparation:

Load datasets:

```
setwd("C:/Users/nickp/Documents/7313DataAnalysis")

#Load churn dataset directly from csv:
df = read.csv("churn_combo.csv") %>% dplyr::select(-X)

#make sure target variable is factor (for classification models; for regression models, not needed):
df$is_churn = as.factor(ifelse(df$is_churn==1, "churn", "loyal"))
```

Split into training/test:

Split data into training and test set:

```
set.seed(7313)
#partition the data evenly along the dependent variable: 70% will be training, 30% testing
training.indices <- createDataPartition(df$is_churn, p = 0.7, list = F)
#training data set
churn_train <- df[training.indices,]
#testing data set
churn_test <- df[-training.indices,]
```

Create a "train control" object

This train control object will apply to all models below. We use 5-fold cross validation. We choose "twoClassSummary" so that we can access metrics other than Accuracy.

```
control <- trainControl(method = "cv", number = 5, allowParallel = T,
                        classProbs = T, summaryFunction = twoClassSummary)
```

Dimension Reduction (PCA):

We do PCA transformation to generate principal component features, which are uncorrelated (orthogonal) linear combinations of our raw features that capture the highest variance in the (remaining) data:

```
#Data Preparation:
#remove the target (and other vars?)
x.train <- churn_train %>% dplyr::select(-customer_id, -is_churn)
x.test <- churn_test %>% dplyr::select(-customer_id, -is_churn)

#PCA on training set (minus target!)
pca_train <- prcomp(x.train, scale = T, center = T)
#calculate variance
pca_train$var <- pca_train$sdev^2
#how much variance explained by each component
pve <- pca_train$var/sum(pca_train$var)
#pve
#graph of cumulative sum of variance explained by each PC
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cum. Prop. of Var. Explained")
```

Based on the amount of variance we see explained by the subsequent Principal Components, we'll cap how many PCs we use. It appears, though, that the cumulative proportion of variance explained by every additional component is rather linear, meaning that we can't reduce the number of variables significantly without losing variance in our data. Still, we'll cap it at 55 principal components (out of 59 original features) because the last few PCs appear to explain less variation than the first 55. (just for fun)

We do PCA transformation using "preProcess" from the caret package here (for consistency), creating the principal components out of the x variables, then adding by the y variable (churn):

```
#Data Preparation:
#remove the target
#x.train <- churn_train %>% dplyr::select(-customer_id, -is_churn)
#x.test <- churn_test %>% dplyr::select(-customer_id, -is_churn)

#create a pre-processing transformation object
preProc <- preProcess(x.train, #don't include target
                      method = c("pca", "scale", "center"),
                      pcaComp = 55)  #how many PCA components to keep (up to "p")

#apply (same) preprocessing to both train and test sets (minus target):
#Thus, we have our (x) features (all PC's or combinations of previous features)
train.pca <- predict(preProc, x.train)
test.pca <- predict(preProc, x.test)
#add the target variable (y) back:
train.pca$is_churn <- churn_train$is_churn
test.pca$is_churn <- churn_test$is_churn
```

Models (all features):

See a list of models available in R package for classification and regression training *caret* (and their tuning parameters) here:

Various methods of estimating model accuracy with caret here. (Note in the following models we just use the same train control objects).

Tuning the "trainControl" object here.

Logistic Model

First using our raw features.

```
start.time <- Sys.time()  #measure how much time it takes

#register cluster
cluster <- makeCluster(detectCores() - 1) # convention to leave 1 core for OS
registerDoParallel(cluster)

#train a logistic model with caret:
glm_model = caret::train(
  form = is_churn ~.,
  data = churn_train,
  trControl = control,
  method = "glm",
  family = "binomial"
)

#de-register parallel processing cluster:
stopCluster(cluster)
registerDoSEQ()     #forces R to return to single-thread processing


#measure passed time:
end.time <- Sys.time()
end.time - start.time


## EVALUATION on test data
#make predictions, using the generated model, on test data
glm_prediction <- predict(glm_model, churn_test)
#create "confusion matrix": Calculates a cross-tabulation of observed and predicted classes with associ
glm.cm <- confusionMatrix(glm_prediction, churn_test$is_churn)
glm.cm
```

Then using our PCA-transformed feature variables:

```
#train a logistic model with caret:
glm.pca_model = caret::train(
  form = is_churn ~.,
  data = train.pca,
  trControl = control,
```

```
  method = "glm",   #linear logistic model (Change this later)
  family = "binomial",
  metric = "ROC"
)

## EVALUATION on test data
#make predictions, using the generated model, on test data
glm.pca_prediction = predict(glm.pca_model, test.pca)
#create "confusion matrix": Calculates a cross-tabulation of observed and predicted classes with associ
glm.pca.cm <- confusionMatrix(glm.pca_prediction, test.pca$is_churn)
glm.pca.cm
```

LDA model:

Linear discriminant analysis:

```
#train a LDA model with caret:
LDA_model = caret::train(
  is_churn ~.,
  data = churn_train,
  trControl = control,
  method = "lda"
)


#Predicted classes:
lda_prediction <- predict(LDA_model, churn_test)

#confusion matrix:
lda.cm <- confusionMatrix(lda_prediction, churn_test$is_churn)
lda.cm
```

Then using our PCA-transformed feature variables:

```
#train a LDA model with caret:
LDA.pca_model = caret::train(
  is_churn ~.,
  data = train.pca,
  trControl = control,
  method = "lda"
)


#Predicted classes:
lda.pca_prediction <- predict(LDA.pca_model, test.pca)

#confusion matrix:
lda.pca.cm.train <- confusionMatrix(predict(LDA.pca_model, train.pca), train.pca$is_churn)
lda.pca.cm <- confusionMatrix(lda.pca_prediction, test.pca$is_churn)
lda.pca.cm
```

QDA model:

```
#train a QDA model:
QDA_model = caret::train(
  is_churn ~. ,
  data = churn_train,
  trControl = control,
  method = "qda"
)

#Predicted classes:
qda_prediction <- predict(QDA_model, churn_test)

#confusion matrix:
qda.cm <- confusionMatrix(qda_prediction, churn_test$is_churn)
qda.cm
```

Then using our PCA-transformed feature variables:

```
#train a QDA model:
QDA.pca_model = caret::train(
  is_churn ~. ,
  data = train.pca,
  trControl = control,
  method = "qda"
)

#Predicted classes:
qda.pca_prediction <- predict(QDA.pca_model, test.pca)

#confusion matrix:
qda.pca.cm.train <- confusionMatrix(predict(QDA.pca_model, train.pca), train.pca$is_churn)
qda.pca.cm <- confusionMatrix(qda.pca_prediction, test.pca$is_churn)
qda.pca.cm
```

Random Forest:

Finally, we train a random forest (only with PCA features):

```
set.seed(7313)
### Random forests, with sqrt(p) features per tree, 10 trees and principal components ##
rf.fit.pca <- caret::train(is_churn~.,
                      data=train.pca,
                      method='rf',
                      ntree = 10,
                      trControl= control )

#calculate training accuracy
#pred = predict(rf.fit.pca, train.pca)
#confusionMatrix(pred, train.pca$is_churn)
#accuracy is 0.9588
```

```r
#calculate test accuracy
rf.pca.cm.train <- confusionMatrix(predict(rf.fit.pca, train.pca), train.pca$is_churn)
rf.pca.cm <- confusionMatrix(predict(rf.fit.pca, test.pca), test.pca$is_churn)
rf.pca.cm
#accuracy is ~75% on test data
```

Comparison of Models

Using the models, we can see that QDA (both using all variables and using PCA variables) generally scores high on sensitivity (the true positive rate, TPR), while Logistic and LDA models score high on Specificity (True Negative Rate, TNR). However, Random forest, seems to do best on both metrics, and therefore does best with ROC. Furthermore, we can see RF does best for accuracy as well. Thus, we choose our **Random Forest** model.

```r
# Evaluate logit, lda and qda models with full features / with
results = resamples(list(
  logit = glm_model, logitPCA = glm.pca_model,
  lda = LDA_model, ldaPCA = LDA.pca_model,
  qda = QDA_model, qdaPCA = QDA.pca_model,
  RF = rf.fit.pca))
#Plot
dotplot(results, main = "Model Metrics")
#plot Accuracies
dotplot(c(qda = qda.cm$overall[1],qdaPCA =qda.pca.cm$overall[1],
          lda = lda.cm$overall[1],ldaPCA = lda.pca.cm$overall[1],
          logit = glm.cm$overall[1], logitPCA = glm.pca.cm$overall[1],
          RF = rf.pca.cm$overall[1]), xlab = "Accuracy", main = "Accuracies by Model")
```

Thus, we choose the **Random Forest** (using PCA with *some* dimension reduction) as our best model.

Estimated value impact

Finally, we can evaluate the impact of our model, making a few (unrealistic) simplifying assumptions:

- The cost of our efforts is 0 SEK (reaching out to churners, bidding them to return to the store, etc)
- Our efforts are 100% successful

We can see the results of our chosen model (Random Forest), when applied to test data:

As we can see, when our model predicts "churn", it is correct ~70% of the time (the *Positive Pred Value*, or Correct # churn / Total pred churn). So out of 1000 new customers that are predicted as "churn", ~700 can be expected to actually churn.

From the database we can calculate the sum of sales and the average sales per customer (of ALL sales, not just the first stale), grouped by those customers that churn vs those that are loyal.

```sql
SELECT is_churn, SUM(sales) AS SumSales, AVG(sales) AS AvgSales, COUNT(*) AS Freq
FROM
(SELECT Receipt.customer_id,
    t.is_churn,
```

```
    SUM(IFNULL(Product.price, 0)) AS sales,
    COUNT(ReceiptProduct.receipt_id) AS num_sold_items
FROM Receipt
LEFT JOIN ReceiptProduct
    ON (Receipt.receipt_id = ReceiptProduct.receipt_id)
LEFT JOIN Product
    ON (ReceiptProduct.product_id = Product.product_id)
LEFT JOIN (SELECT customer_id, IF(COUNT(*) = 1, 1, 0) AS is_churn FROM Receipt GROUP BY customer_id) t
    ON (Receipt.customer_id = t.customer_id)
GROUP BY customer_id) innerquery
GROUP BY is_churn;
```

Therefore, for every customer we can retain, we expect on average to make **212 SEK** more (the average loyal customer has average sales of 255 SEK while the average churner has average sales of 43 SEK). Therefore, if retention costs are below 212 SEK per customer retained, we can expect a positive net result. For the following value impact, though, **we'll assume it costs nothing to retain a customer, and retaining an identified churner is 100% certain** (not realistic, adjust later):

One very basic measure of total expected value, assuming we reach out to EVERYONE we suspect as a churner, then, could be calculated by taking the number of correctly predicted churners (~50,000) and multiply by the difference in average sales (between churners and loyal customers), or 212 SEK. We get ~50,000*212 = 10.6 million SEK.

A more marginal (and thus applicable) estimate of value impact: For every 1000 predicted churners that we approach (and successfuly retain at no cost), we can assume 700 are actually would-be churners, which, if we retain, will net us 700*212 SEK, or 148,400 SEK.

To be more realistic, then, we can multiply our estimate by a probability of success (we retain X% of customers we approach, e.g.) and also subtracting out costs for the effort of reaching predicted churners. To be done later...

Final Accuracy Table:

```
Accuracy <- data.frame(Model = c("LDA with PCA","QDA with PCA", "RF with PCA","LDA with PCA","QDA with P
                       TrainTest = c("Train","Train","Train","Test","Test","Test"),
                       Accuracy = c(lda.pca.cm.train$overall[1],
                                    qda.pca.cm.train$overall[1],
                                    rf.pca.cm.train$overall[1],
                                lda.pca.cm$overall[1],
                                    qda.pca.cm$overall[1],
                                    rf.pca.cm$overall[1]))
Accuracy$TrainTest %<>% factor(levels = c("Train","Test"))

ggplot(Accuracy, aes(x=TrainTest, y=Accuracy, shape=Model, color = Model)) +
  geom_point() + ylim(0,1) + xlab("Train or Test")
```

Graphs

See code here for graph creation

```r
library(RMySQL)   #MySQL interaction
library(dplyr)    #data handling
library(ggplot2)   #for graphs
library(magrittr)  #for pipes

#Load final dataset
df <- read.csv("churn_combo.csv")

mydb <- dbConnect(MySQL(), dbname = "ToyStorey", host = 'db.cfda.se',
                  port = 3306, user = "toystorey", password = "toys@sse")

#Load customer-level aggregates:
query2 <- 'SELECT customer_id, COUNT(DISTINCT(receipt_id)) AS NumberStoreVisits, SUM(price) - SUM(cost)
FROM Receipt LEFT JOIN ReceiptProduct USING(receipt_id) LEFT JOIN Customer USING(customer_id) LEFT JOIN
GROUP BY customer_id'
customers <- fetch(dbSendQuery(mydb,query2), n=-1)

#join churn data with customer aggregates:
df <- left_join(df, customers, by = "customer_id")

#create new variables/clean types:
df$month <- lubridate::month(as.Date(df$FirstDate))
mymonths <- c("Jan","Feb","Mar",
              "Apr","May","Jun",
              "Jul","Aug","Sep",
              "Oct","Nov","Dec")
df$MonthAbb <- mymonths[ df$month ]
df$is_churn %<>% as.factor()
df$months <- factor(df$MonthAbb, levels = c("Jan","Feb","Mar",
                                            "Apr","May","Jun",
                                            "Jul","Aug","Sep",
                                            "Oct","Nov","Dec"))
df$FirstDate %<>% as.Date()


##VISUALIZATIONS:

#Variables to work with:
colnames(df)

##BARPLOTS
#the problem with the churn variable: by month
ggplot(df,
       aes(x = months, y = ..count..,
           fill = is_churn)) +
  geom_bar(position = "stack")

#loyalty in countries, about even...:
ggplot(df,
       aes(x = country, y = ..count..,
           fill = is_churn)) +
  geom_bar(position = "stack") + coord_flip()
```

```
##SCATTERPLOTS
## number of sold items by profit, colored by churn or not
ggplot(df,
       aes(x = num_sold_items, y = ProfitbyCustomer,
           color = is_churn)) + geom_point()

ggplot(df,
       aes(x = num_sold_items, y = ProfitbyCustomer,
           color = is_churn)) + geom_point()

##DENSITYPLOTS
cdplot(is_churn ~ months, data=df)
ggplot(df, aes(x= ProfitbyCustomer, color = is_churn)) +  xlim(0,500) +
  geom_density()

##BOXPLOTS
ggplot(df, aes(x=is_churn, y=ProfitbyCustomer)) +
  geom_boxplot()  + coord_flip() + ylim(0,500)
```