Optimizing Promo Allocation to Minimize Subsidy Spent

Ride Hailing as Case Study

Mohamed Ali & Adesijibomi Aderinto
Linköping University

12/16/2021

Industry overview

GO is a ride-hailing service company in Sudan-Africa they provide an on-demand car booking service by leveraging smart mobility through a combination of user-friendly features and booking software. GO is an on-demand BUS service with a technology-based alternative for transportation, affordable and convenient. It's not only a service for shuttling but a smart solution that inspires reforming the public transportation system in Sudan and improves people's daily experience.

Problem Statement

The commercial department strives to lower the cost of operations (customer subsidy) when conducting customer promotion campaigns. Customers offered a discounted price per trip (using a promo code). In each campaign cycle, bulk SMS and in-app notification are used to communicate with the customer. Campaigns are done at random where they send bulk SMS for the whole customer base this lead to non-optimized customers subsidy and high operation cost. The cost of the operation is defined as follow:
* Cost of SMS.
* Expenditure cost (Subsidy).
Typically, the marketing team will get promotional activities regularly, and they locate customers based on their activity levels (that is, customers who have had activities in the past 90 days).

Implications & Proposed solution

As the main purpose of promotion campaigns is to increase the activity level and increase the number of trips per day, targeting customers in a blind manner will lead to less response rate those campaigns, the main idea is we want to ensure that the customer who is offered with a promo will most probably use it.
Proposed solution
A classification model to identify customer base on a set of features to better understand Promo vs non-Promo users' behaviour, aiming to identify, which patterns distinctively in Promo user's vs non-Promo users. As result, we expect to have a set of criteria to better select potential customers who will adapt to the campaign and generate segment base promos, in addition, to building an ML model where we can predict future unseen users and design better campaigns to target them and test the likability of the customer to use the promo code or not.

Data description and fields

The data set based on customer behaviour in the past three months includes 77879 rows and 15 features. The y variable (target variable) represents Promo and non-promo users. This classification is done based on the useability of customer promotions in the past three months and then labelled as (0,1). The below list describes the set of features used to study the behaviour of the promo vs non-promo users. Y = promo vs non-promo users.

Y = promo vs non-promo.
X1 = customer tenure (total number of days since the customer joined the service).
X2 = Avg.spent (Avg. customer spend for the last 3 months).
X3 = Avg. distance traveled (Avg. distance traveled by the customer for the last three months).
X4 = Male/Female (Gender class).
X5 = most used location (BH/KH/UMD) (Main Cities in Khartoum Capital) (Most used location base on the number of trips conducted by the customer in the last three months).
X6 = Post assigned trips count of three months (Post assigned trips count for the last three months – Post assigned defined as customer take the action to cancel the trip after it been assigned to a captain).
X7 = Pre assigned trips count three months (pre assigned trips count for the last three months – Post assigned defined as customer take the action to cancel the trip before it been assigned to a captain).
X8 = Avg rating (Avg customer rating).
X9 = Preferred ride type (Open / Close) (Preferred ride type close/open base on last three months data).
X10 = Preferred Payment method (Cash / Wallet) (Preferred ride type close/open base on last three months data).
X11 = Customer engagement (number of active days for the last three months).
X12 = Total trips (count) (total completed trips count for the last three months).
X13 = Value segmentation (Customer value segmentation base).

Descriptions of model used and experimental design

Decision Tree (Classification tree)

To achieve our objective, a Decision tree will be used as a classification model to predict Promo vs Non-Promo users based on the selected features. A decision tree works on a set of rules that defines the regions explicitly. The input space is effectively divided into multiple disjoint regions, and in each region, a constant $\hat{Y}(x_*)$ is assigned to the prediction Lindholm et al. (2022). In Classification trees, we use majority vote to compute the predictions associated with each region, expressed as:

$$\hat{y}_l = Majority\ Vote\{y_i : x_i \in R_l\} \ldots\ldots\ldots\ldots\ldots\ldots_{1.1}$$

and the split at any internal node is computed by solving an optimization problem of the form:

$$min_{j,s}\ n_1 Q_1 + n_2 Q_2 \ldots\ldots\ldots\ldots\ldots\ldots_{1.2}$$

where $n1$ and $n2$ denote the number of training data points in the left and right nodes of the current split, respectively, and $Q1$ and $Q2$ are the costs (derived from the prediction errors) associated with these two nodes. The variables j and s denote the index of the splitting variable and the cut-point Lindholm et al. (2022):

Experimental design

The dataset will be divided into two subsets *(Training and Testing),(70%,30% respectively).*To achieve the algorithm described in section 1.2, function *rpart()* from *rpart* Therneau, Atkinson, and others (1997), will be used to predict the regions, function *predict()* from *rstats* will be used to predict the test output. The experiment will start first by fitting the model with the training data set then we predict into our testing dataset for the unseen data. Different model selection and optimization processes will be used to achieve the best fit (as shown in next section).

First Model: Classification tree

As we mentioned in the data description, our data-set consist of 15 feature, our end point Y(*target end point*) denote if the customer is *promo user (1)* or *non-promo (0)* features of the model as described in *Data description and fields.* function *read_csv()* is used to load data into Rstudio environment, next we divide the data-set into two subsets *(Training 70% and Testing 30%)*, thereafter use *summary()* to summarize our data set as shown in **Output 1.1**.

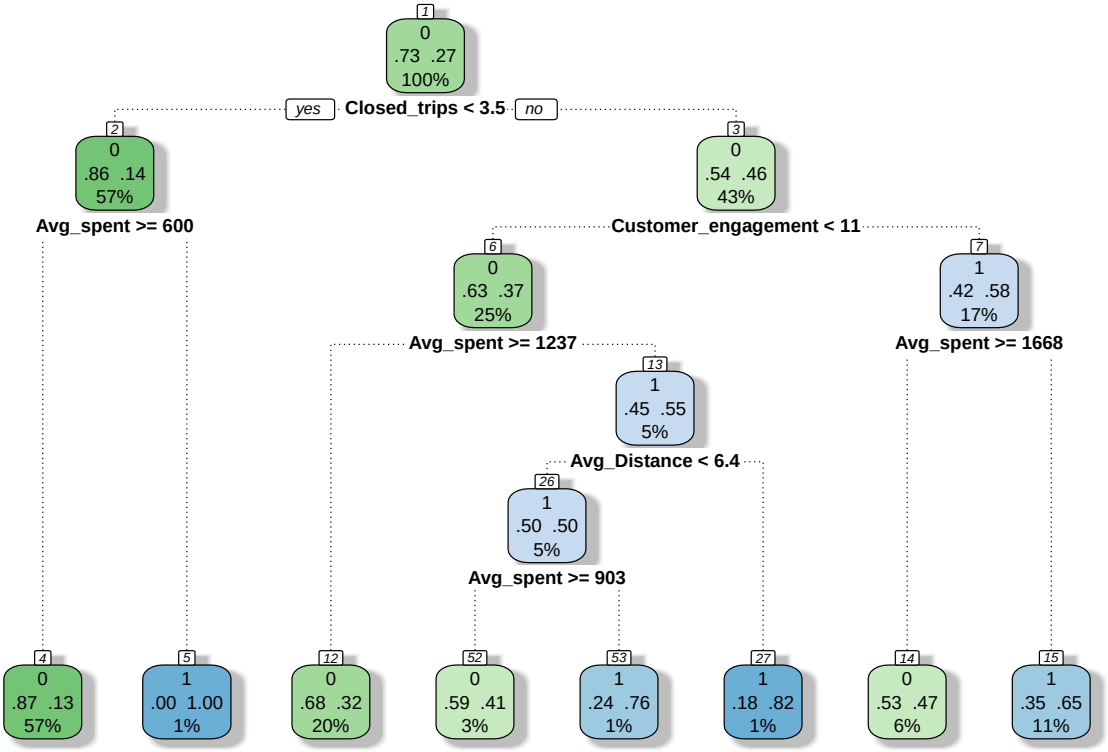
```
## [1] "numeric"
```

Implemintaion

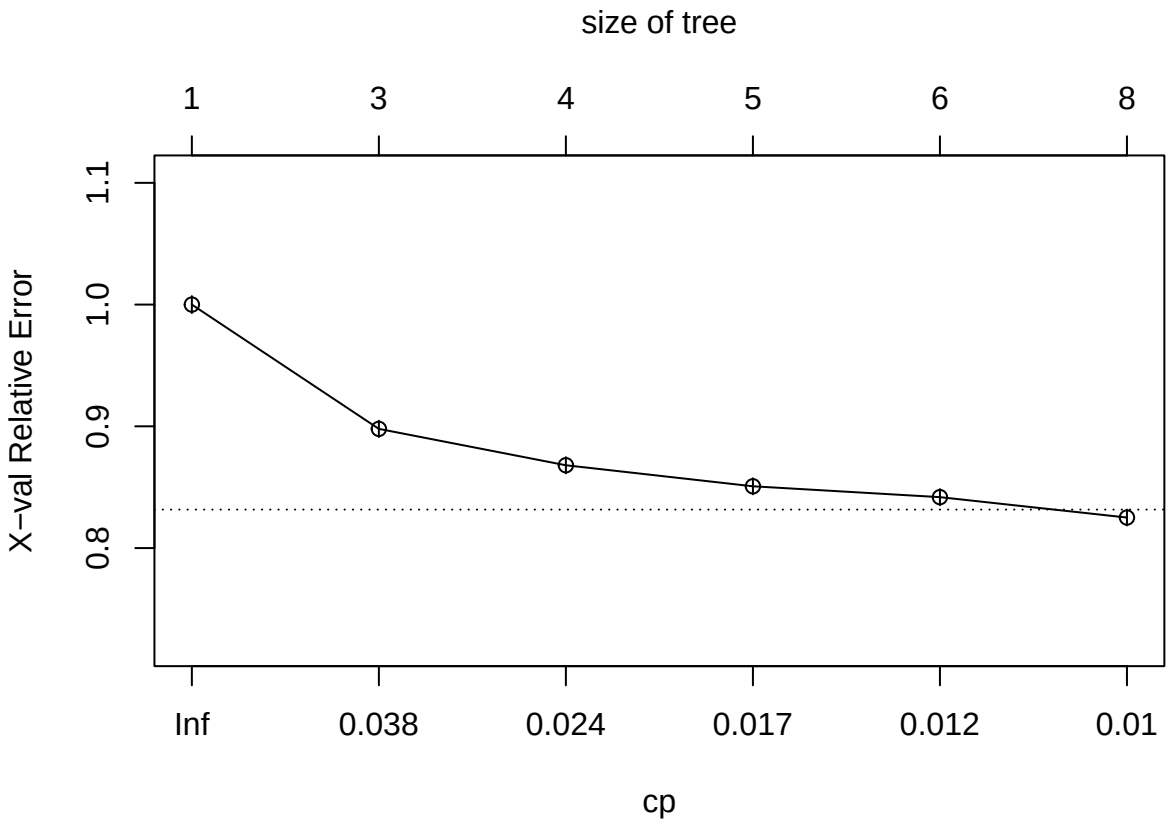
The model is fitted by using the *rpart* function. The first argument of the function is a model formula, with the $\sim$ symbol interpreted as *"is modeled as"*. The print function gives an abbreviated output and *fancyRpartPlot()* from package rattle then used to plot our result tree. In r part method *class* to indicated that it's a classification tree:

```
## n= 54515
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 54515 14927 0 (0.7261855 0.2738145)
##    2) Closed_trips< 3.5 31321  4311 0 (0.8623607 0.1376393)
##      4) Avg_spent>=599.5 30916  3906 0 (0.8736577 0.1263423) *
##      5) Avg_spent< 599.5 405     0 1 (0.0000000 1.0000000) *
##    3) Closed_trips>=3.5 23194 10616 0 (0.5422954 0.4577046)
##      6) Customer_engagement< 10.5 13737  5088 0 (0.6296135 0.3703865)
##       12) Avg_spent>=1236.5 10768  3451 0 (0.6795134 0.3204866) *
##       13) Avg_spent< 1236.5 2969  1332 1 (0.4486359 0.5513641)
##         26) Avg_Distance< 6.35 2518  1253 1 (0.4976172 0.5023828)
##           52) Avg_spent>=902.5 1869   774 0 (0.5858748 0.4141252) *
##           53) Avg_spent< 902.5 649   158 1 (0.2434515 0.7565485) *
##         27) Avg_Distance>=6.35 451    79 1 (0.1751663 0.8248337) *
##      7) Customer_engagement>=10.5 9457  3929 1 (0.4154594 0.5845406)
##       14) Avg_spent>=1667.5 3385  1588 0 (0.5308715 0.4691285) *
##       15) Avg_spent< 1667.5 6072  2132 1 (0.3511199 0.6488801) *
```

Printing fit_def explains steps of the splits, we can see that we started with *54,515* observation at the root node, and the first feature we split on is Avg of Closed_trips *(which indicate that it's the first feature that minimize our optimization problem)*, we can see that all customers with an *average closed trips less than 3.5* goes to the left (our 4th branch) and those with *Avg.trips more than 3.5* goes to the right (3rd branch).
The second feature that can also describe the customer behavior toward promo campaign is *Avg_spend*, as we can see if the customer had *on average spend of >=599.5* then it's most probable to ignore the promo campaign, similarly the right branch which divide the customers again base one their *Avg Closed_trips*, but this time for those who have average trips less than *12* and average spent greater than *1282* are most likely to ignore the campaign.
We can also plot our tree to see the result in more visual way, using *fancyRpartPlot()* from package *rattle* we achieve:



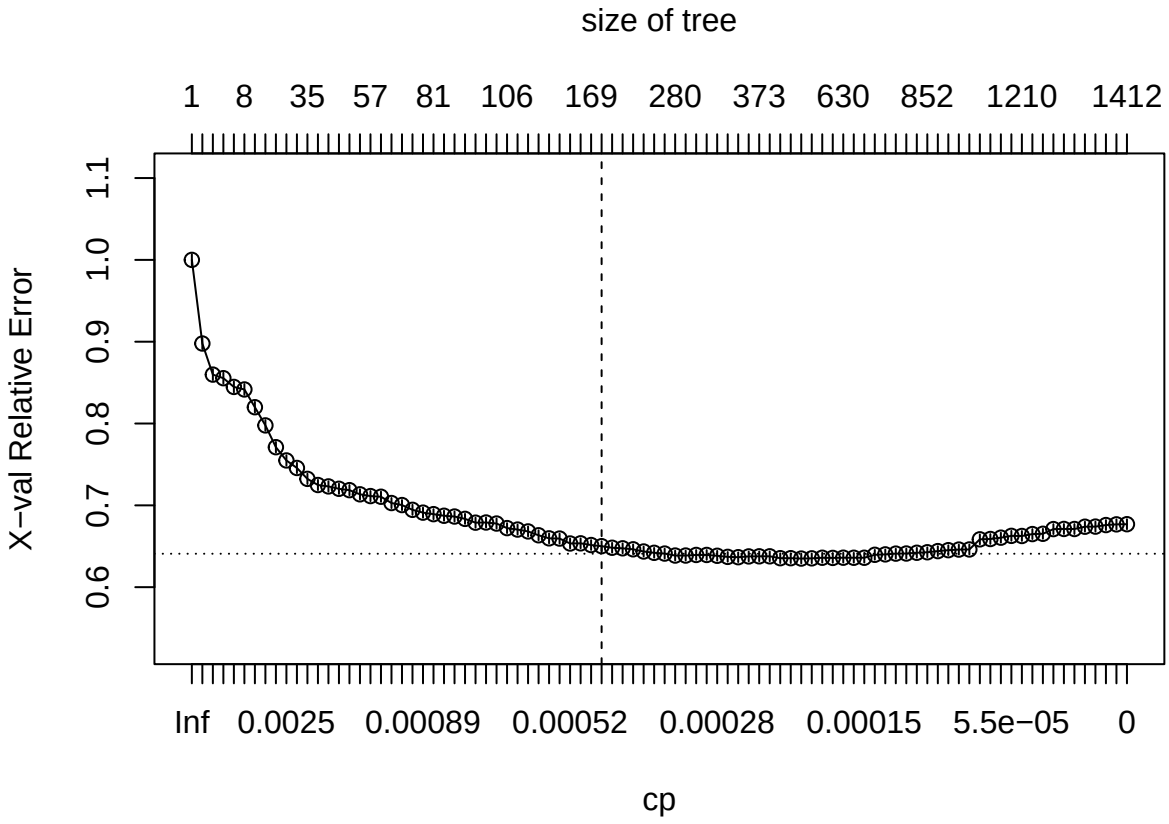
Analyzing the above plot we can observe that, features like *(close_trips, Avg_spent, Avg_distance)* highly affect our response variable *(customer promo usage)* We can see that the last node on the bottom left side *(represent almost 58% of our classification)* which when customers have *less than 3.5 trips in the last three month* and they spent on avg *698 SDG (Sudanese Pound)* most likely to not use the promo code.
In *rpart* the parameter *cp (Complexity parameter)* indicates the minimum improvement in the model needed at each node. To select the right *cp level*, Many techniques have been discussed in the literature *(ie. minimum possible cross-validated error,one-standard error* Lindholm et al. (2022), however for the purpose of this research we will consider the *cp* value by choosing the lowest level with the minimum *xerror (10-fold cross) value*, we can see in the plot below that our tree find diminishing returns after 7 terminal nodes.



(Note that y-axis is cross validation error, lower x-axis is cost complexity (cp) value, upper x-axis is the number of terminal nodes (tree size)

Now lets grow different trees with different settings, and try to find out which is the optimal model we can achieve to better fit our data. in general *rpart()* automatically applying a range of *cp* values to prune the tree. However, to achieve this task we can use *rpart.control()* to change the settings of the tree, initially we can obtain a full grown tree where we set our *cp* to *0* and *minisplit* equal to *1* (the result of this tree most probably will over fit our data).

Note that minisplit the minimum defind as number of data points required to attempt a split before it is forced to create a terminal node Therneau, Atkinson, and others (1997).



We can see that the tree converges to 1 observation per node, thereby the *cp* parameter controls the tree number of nodes in which as *cp* get small the tree complexity increase.
It's clear that changing in the hyper-parameters gives different tuned models, therefore.We'll simulate different scenarios so we can achieve different models, *rpart.control()* parameters such as *minisplt* and *cp* and *maxdepth* could be used here to control the tree size, where each parameter defined as Therneau, Atkinson, and others (1997):
minsplit: the minimum number of data points required to attempt a split before it is forced to create a terminal node the default is 20.
maxdepth: the maximum number of internal nodes between the root node and the terminal nodes. The default is 30.
As the cost of generating those scenarios manually is time consuming, a grid search will be performed to automatically search across a range of differently tuned models to identify the optimal hyperparameter setting.
Note that we define a sequence of *minsplit from 10 to 20* with *2 steps* and *maxdepth with sequence for 1 to 40*, we result with *100* combinations of models (since we have min cp level at tree depth of 40 from the above graph).

minsplit	maxdepth	cp	error
18	24	0.01	0.8200576
20	40	0.01	0.8240102
18	30	0.01	0.8256850
18	16	0.01	0.8272258
14	24	0.01	0.8273598

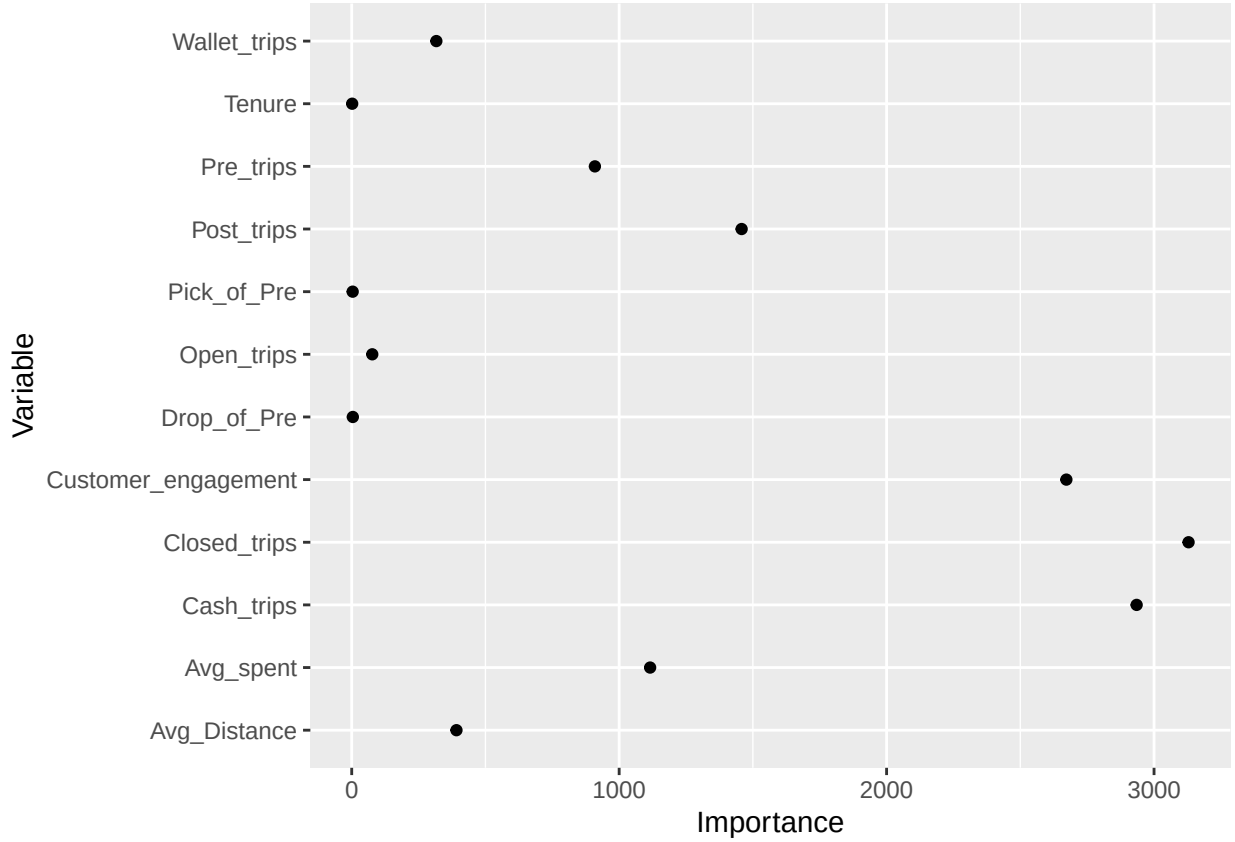
The above table tell us that we recive the min xerror at model settings *minsplit = 18* and *maxdepth=24* and

cp=0.01.
Now let's add more parameters to achieve better fit for our model, let assume that there is a cost for wrong classification, in which if we classify promo user as *non-promo* user this will raise the marketing cost *(2 times) (SMS cost)*, thought we introduce *loss matrix (0,1,2,0)*, thereby we control our classification by saying that it's better to classify customer as *promo* user if you are not certain that it's a *non-promo* user. In *rpart*, *prams()* handle the loss function that we define below and the model give us another clarification base on this role.

Now let's have a look at the three models together in term on misclassification rate comparing training and testing data set:

Data	Default	Prune_tree	Loss_Mat_tree
Train	0.2370540	0.2217371	0.2372741
Testing	0.2360998	0.2205624	0.2378547
Tree size	9.0000000	15.0000000	9.0000000

From the results above we can see that prune tree with sittings *minsplit = 18*, *maxdepth = 24*,*cp=0.01*, which been obtained from the grid search method gives the lowest misclassification rate for both Training and testing data sets, it also has a considerable *number of leaves 15*. Now using this tree sitting and *rpart* we can obtain an analysis on which features are most important when we do *a promo* vs *non-promo* users classification.



We can see that features like *Avg count of closed trips, Avg count of cash paid trips, Post assigned trips and Avg spent* considered as the most important feature in our model.

Descriptions of Model used and Experimental Design

Logistic Regression

Logistic regression can be viewed as a modification of the linear regression model so that it fits the classification (instead of the regression) problem[1].The key Idea behind logistic regression is to squeeze from:

$$z = \theta_0 + \theta_1 x_1 + ... + \theta x_p = \theta^T x$$

By using the logistic function $h(z) = e^z/1 + e^z$ to interval $[1, 0]$, this results as:

$$g(x) = \frac{e^{\theta^T x}}{1 + e^{\theta^T x}}$$

Thus the algorithm[1] for solving equation 2.1:
1- Learn binary logistic regression.
1.2- Data: Training data $T = \{x_i, y_i\}_{i=1}^n$ with output classes $y = \{-1, 1\}$.
1.2- Result: Learned parameter vector $\hat{\theta}$.
2- Predict with binary logistic regression.
2.1- Data: Learned parameter vector $\hat{\theta}$ and test inout x_*.
2.2- Result: Prediction $\hat{y}(x_*)$.

Experimental Design

Second Model Logistic Regression

In this research we will use functions *glm()* and *glmnet()* from glmnet package, first we'll perform the model with default settings using *glm()* and estimate which features are significantly affect our model, thereafter we'll use different sittings to penalized the logistic regression and try to find the best model for our data. Finally we'll conduct TPR FPR analysis to compare logistic regression model to Tree model that we obtained in section 1 and find which model could be used to fit our data.

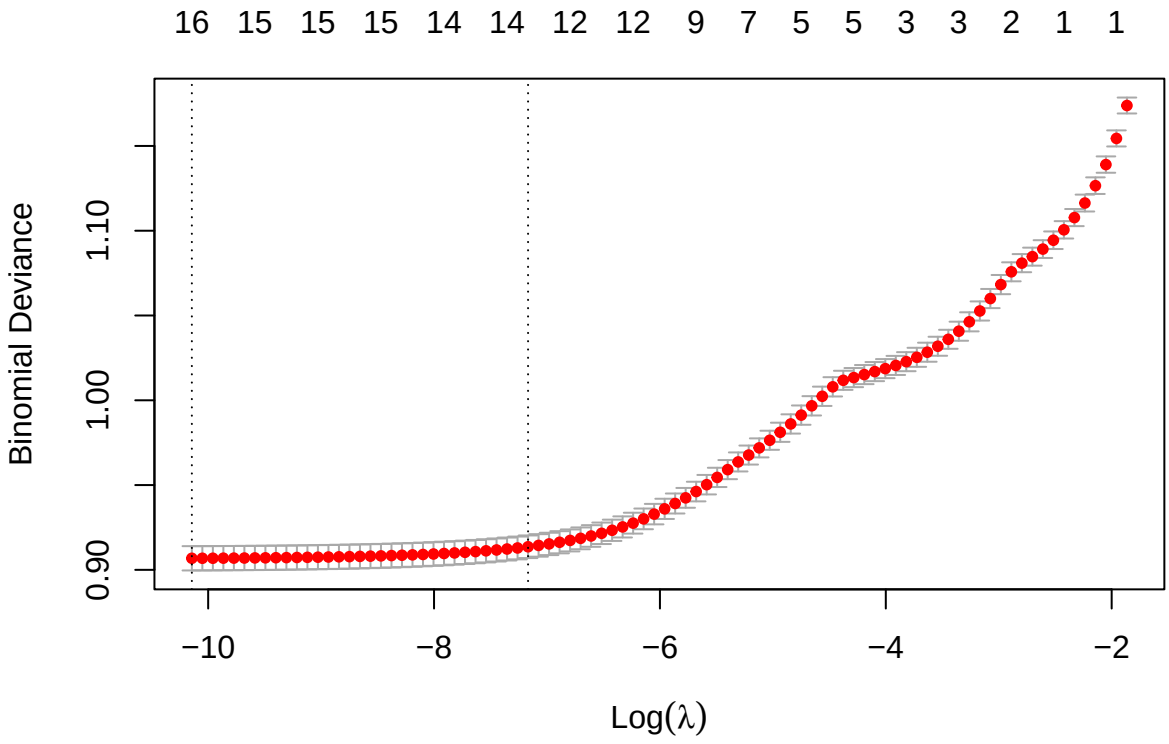
Implemintaion

Data preparation Again as we did in first section we'll divide our data set into *training and testing, 70% 30%.* Model with default sittings *Output 2.1.*

Result shows logistic regression model output, * indicated the features with high impact in our model, which are, *Tenure, Avg_spent,Avg_Distance , Pre_trips, Customer_engagement , Drop_of_PreOMD, Pick_of_PreOMD. Avg Distance travelled and Customer_engagement* represent the most features with impact in our model, interestingly none of *average cash trips or average closed trips* were included in this model compared to the ones that we obtained in *section 1.*
Interpreting the result of logistic regression is bit different to Decision tree, where we look at the $logit(p)$ is just a shortcut for $log(p/1-p)$, where $P = P\{Y = 1\}$, i.e. the probability of "*success*", or the presence of an outcome in our case is the customer been promo user, motivated by linear regression interpretation $Y = a + bX_1 + cX_2$ a 1-unit increase in X_1 will result in an increase in Y by b units, and of course if is negative then we will say one unit increase in X will result as one unit decrease in Y by b unit, logistic regression works in the same manner where we use the log-odds ratio. a 1 unit increase in X_1 will result in b increase in the log-odds ratio of success, in other words the 1 unit increase in X will result in an increase in the odds of $Y = promo$ user against $Y = non - promo$ user.
Looking at the results about we can conclude that *Avg_spent*, *Pre_trips*, *Drop_of_PreBAH*, *Pick_of_PreBAH* will increase the odds of the customer of being a non-promo, and vise users for the features with positive coefficients.
Now we'll compute penalized logistic regression, by using *glmnet()* function we can specify different model parameters, to obtain we define:

```
glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL)
```

Where *alpha = 1* represent *LASSO regression*, which will be used in this research. The value of *Lambda* or *(penalty parameter)* is used to adjust the amount of the coefficient shrinkage. The best lambda for our data, can be defined as the lambda that minimize the *cross-validation* prediction error rate. To achieve this we'll use *cv.glmnet*, first we specify our features model matrix by using *model.matrix* from stats package, and *y*.



The plot displays the *cross-validation* error according to the log of *lambda*. The left dashed vertical line indicates that the log of the optimal value of lambda is *approximately -7*, which is the one that minimizes the prediction error. This lambda value will give the most accurate model. The exact value of lambda can be viewed as follow:

```
## [1] 3.928728e-05
```

Now let's compare the penalized model with the default in term of Misclassification rate:

Data	Misclassification_Train	Misclassification_Test
Defult	0.2162157	0.2130291
Penalized	0.2170595	0.2138424

Models Compersion: ROC as Evaluation Metric

In order to study which model could be used to achieve the best customer prediction, we will compare the best model that we achieved from step 1 (Decision trees) vs the one that we conclude with form step 2 (Logistic regression) the compression will be base on ROC curve in which it tell us what model is more capable of distinguishing between classes.And as ROC takes into account all possible threshold levels we'll favor it to other metrics like missclassification error which takes only one threshold level into account.

ROC usually plotted with TPR(Sensitivity) against the FPR (1-Specificity) where TPR is on the y-axis and FPR is on the x-axis. We'll use package pROC() to plot the both ROC lines for logistic regression and pruned tree model as follow:

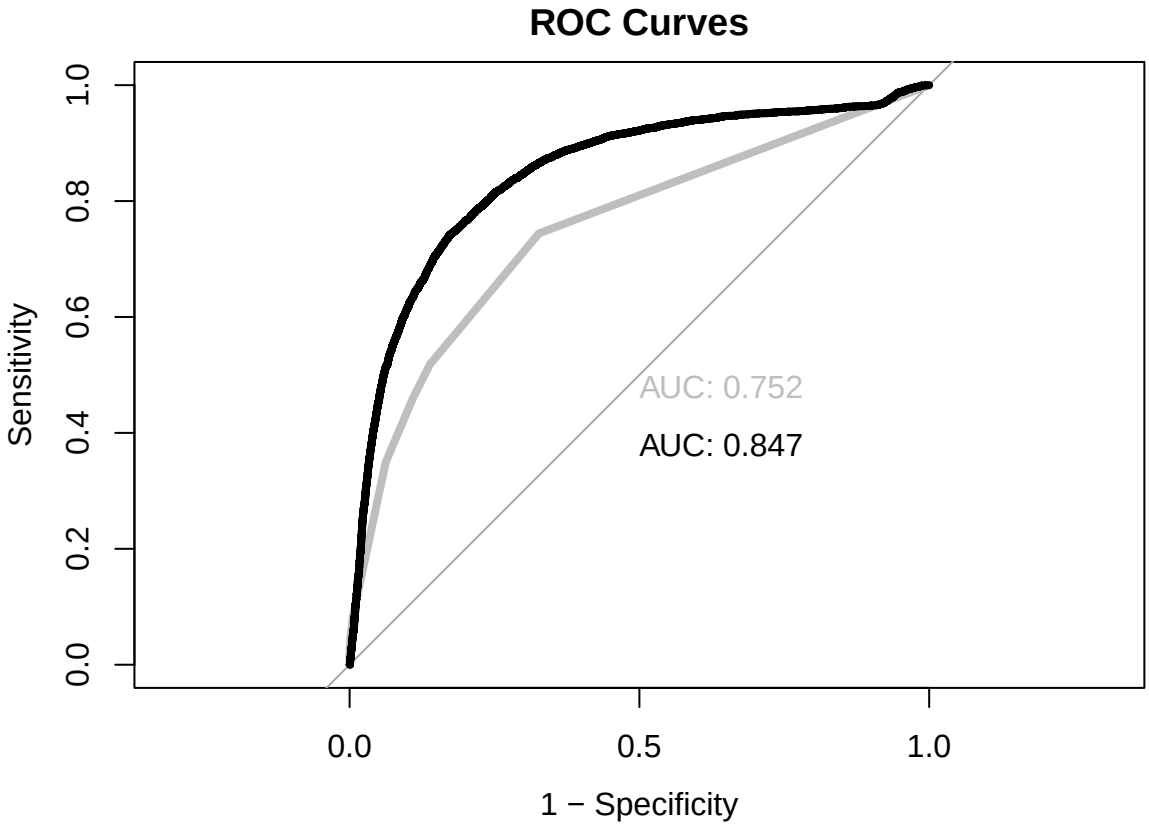
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls > cases
```

```
##
## Call:
## roc.formula(formula = test$Y ~ pred_tree[, 1], plot = TRUE, print.auc = TRUE,     col = "gray", lwd =
##
## Data: pred_tree[, 1] in 17079 controls (test$Y 0) > 6284 cases (test$Y 1).
## Area under the curve: 0.7519
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.formula(formula = test$Y ~ pred_logreg, plot = TRUE, print.auc = TRUE,     col = "black", lwd = 4
##
## Data: pred_logreg in 17079 controls (test$Y 0) < 6284 cases (test$Y 1).
## Area under the curve: 0.8473
```

AUC tells us how much the model is capable of distinguishing between classes. therefore we can conclude that Logistic Regression model perform well on our data-set.

Conclusion

From the logistic regression model, we observed that we have more significant features in the model compared to the classification tree model. This helps to predict that segmenting customer activities with a focus on the significant features in our model will reduce subsidy cost and capital expenditures in SMS cost. With this actionable data from the logistics model, we can suggest segmentation of campaigns.
1- Based on geographic location only to *UMD*.
2- Average spent in the last 3 months.
3- Total number of days since the customer joined the service.
4- Average distance traveled by the customer for the last three months.
5- Average customer spending in the past 3 months.
6- Number of active days for the last three months.

Statement of Contribution

Mohamed and Adesijibomi devised the whole assignment together, the main conceptual ideas and codes outline. Mohamed worked out the first model, Adesijibomi worked with the second model, both results been discussed together and the final out comes as well.

Appendix

Output 1.1

```
summary(Ride_Halling)
```

```
##   customer_id      total_trips        promo_Perc        promo_trips
##  Min.   :    26   Min.   :  1.000   Min.   : 0.00000   Min.   :  0.000
##  1st Qu.: 77213   1st Qu.:  1.000   1st Qu.: 0.00000   1st Qu.:  0.000
##  Median :153085   Median :  3.000   Median : 0.00000   Median :  0.000
##  Mean   :143544   Mean   :  6.823   Mean   : 0.17143   Mean   :  1.194
##  3rd Qu.:212207   3rd Qu.:  7.000   3rd Qu.: 0.09091   3rd Qu.:  1.000
##  Max.   :263542   Max.   :193.000   Max.   :22.00000   Max.   :211.000
##  Y             Tenure        Avg_spent      Avg_Distance       Gender
##  0:56668   Min.   :  0.0   Min.   :  300   Min.   : 0.000   Male  :41913
##  1:21211   1st Qu.: 79.0   1st Qu.: 1261   1st Qu.: 6.050   Female:35966
##            Median :194.0   Median : 1625   Median : 8.500
##            Mean   :219.5   Mean   : 1765   Mean   : 9.292
##            3rd Qu.:336.0   3rd Qu.: 2100   3rd Qu.:11.800
##            Max.   :683.0   Max.   :92810   Max.   :57.000
##    Pre_trips         Post_trips          Rating        Open_trips
##  Min.   :  0.000   Min.   :  0.000   Min.   :0.000   Min.   :  0.0000
##  1st Qu.:  0.000   1st Qu.:  0.000   1st Qu.:1.667   1st Qu.:  0.0000
##  Median :  0.000   Median :  1.000   Median :2.688   Median :  0.0000
##  Mean   :  1.073   Mean   :  2.058   Mean   :2.681   Mean   :  0.2956
##  3rd Qu.:  1.000   3rd Qu.:  2.000   3rd Qu.:3.889   3rd Qu.:  0.0000
##  Max.   :342.000   Max.   :237.000   Max.   :5.000   Max.   :112.0000
##   Closed_trips       Cash_trips       Wallet_trips      Customer engagement
##  Min.   :  0.000   Min.   :  0.000   Min.   :  0.0000   Min.   : 0.00
##  1st Qu.:  1.000   1st Qu.:  1.000   1st Qu.:  0.0000   1st Qu.: 2.00
##  Median :  3.000   Median :  3.000   Median :  0.0000   Median : 3.00
##  Mean   :  6.527   Mean   :  6.268   Mean   :  0.5546   Mean   : 6.44
##  3rd Qu.:  7.000   3rd Qu.:  7.000   3rd Qu.:  0.0000   3rd Qu.: 8.00
##  Max.   :193.000   Max.   :193.000   Max.   :185.0000   Max.   :88.00
##  Drop_of_Pre Pick_of_Pre
##  KH :49839   KH :51232
```

```
##  BAH:13848   BAH:14469
##  OMD:14192   OMD:12178
##
##
##
```

Output 2.1

```
summary(fit_logreg)
```

```
##
## Call:
## glm(formula = Y ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.1239  -0.6928  -0.4790   0.6208   8.4904
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.0547371  0.0537669  19.617  < 2e-16 ***
## Tenure               0.0011153  0.0000702  15.889  < 2e-16 ***
## Avg_spent           -0.0066867  0.0001118 -59.809  < 2e-16 ***
## Avg_Distance         0.8579442  0.0158596  54.096  < 2e-16 ***
## GenderFemale         0.0206714  0.0223560   0.925   0.3552
## Pre_trips           -0.0176025  0.0032566  -5.405 6.47e-08 ***
## Post_trips          -0.0038021  0.0044180  -0.861   0.3895
## Rating              -0.0013808  0.0072970  -0.189   0.8499
## Open_trips           1.2627081  6.0388093   0.209   0.8344
## Closed_trips         0.9011872  6.0388054   0.149   0.8814
## Cash_trips          -0.9466193  6.0388320  -0.157   0.8754
## Wallet_trips        -0.9160223  6.0388395  -0.152   0.8794
## Customer_engagement  0.1546296  0.0050553  30.587  < 2e-16 ***
## Drop_of_PreBAH      -0.0294870  0.0329929  -0.894   0.3715
## Drop_of_PreOMD       0.0893925  0.0372736   2.398   0.0165 *
## Pick_of_PreBAH      -0.0176776  0.0325985  -0.542   0.5876
## Pick_of_PreOMD       0.0948622  0.0378066   2.509   0.0121 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 64002  on 54514  degrees of freedom
## Residual deviance: 50091  on 54498  degrees of freedom
## AIC: 50125
##
## Number of Fisher Scoring iterations: 8
```

Codes

```
knitr::opts_chunk$set(echo = TRUE)
library(glmnet)
library(pROC)
library(readr)
```

```r
library(tidyverse)
library(tree)
library(rpart)
library(rattle)
library(rpart.plot)
library(RColorBrewer)
Ride_Halling<- read_csv("Ride_Haling.csv", col_types = cols(Y = col_factor(levels = c("0", "1")),Gender

Ride_Halling<- read_csv("Ride_Haling.csv", col_types = cols(Y = col_factor(levels = c("0", "1")),Gender

data<-Ride_Halling[,-c(1:4)]
colnames(data)[13]<-c("Customer_engagement")
data<- na.omit(data)
class(as.numeric(data$Tenure))

n <- dim(data)[1]
set.seed(12345)
id <- sample(1:n, floor(n*0.7))
train <- data[id,]
id1 <- setdiff(1:n, id)
set.seed(12345)
id2 <- sample(id1, floor(n*0.3))
test <- data[id2,]

fit_def <- rpart(Y~.,data = train,method = "class")
print(fit_def)
fancyRpartPlot(fit_def, caption = NULL)
plotcp(fit_def)
fit_full <- rpart(Y~.,data = train,method = "class",control = list(cp = 0,xval = 10))
plotcp(fit_full)
abline(v = 40, lty = "dashed")
hyper_grid <- expand.grid(
  minsplit = seq(10,20,2),
  maxdepth = seq(8,40,2)
)

models <- list()
for (i in 1:nrow(hyper_grid)) {

  # get minsplit, maxdepth values at row i
  minsplit <- hyper_grid$minsplit[i]
  maxdepth <- hyper_grid$maxdepth[i]

  # train a model and store in the list
  models[[i]] <- rpart(Y~.,data = train,method = "class",
    control = list(minsplit = minsplit, maxdepth = maxdepth,xval = 10)
  )
}

get_cp <- function(x) {
  min    <- which.min(x$cptable[, "xerror"])
  cp <- x$cptable[min, "CP"]
}
```

```
get_min_error <- function(x) {
  min    <- which.min(x$cptable[, "xerror"])
  xerror <- x$cptable[min, "xerror"]
}
knitr::kable(hyper_grid %>%
  mutate(
    cp    = purrr::map_dbl(models, get_cp),
    error = purrr::map_dbl(models, get_min_error)) %>%
  arrange(error) %>%
  top_n(-5, wt = error))
fit_loss<- rpart(Y~.,data = train,method = "class",
                 parms=list(loss=matrix(c(0,1,2,0),byrow=TRUE,ncol=2)))
fit_def <- tree(Y~.,data = train,method = "class")
fit_prune<-rpart(Y~.,data = train,method = "class",
      control = list(minsplit = 18, maxdepth = 24,cp=0.01))
fit_loss<- rpart(Y~.,data = train,method = "class",
                 parms=list(loss=matrix(c(0,1,2,0),byrow=TRUE,ncol=2)))

# a. Decision Tree with default settings (train)
misclass_train_def <- sum(predict(fit_def, newdata = train, type = "class")
                          != train$Y) / nrow(train)
# a. Decision Tree with default settings (validation)
misclass_val_def <- sum(predict(fit_def, newdata = test, type = "class")
                        != test$Y) / nrow(test)

# b. Decision Tree with prune (train)
misclass_train_prune <- sum(predict(fit_prune, newdata = train, type = "class")
                            != train$Y) / nrow(train)
# b. Decision Tree with prune (validation)
misclass_val_prune <- sum(predict(fit_prune, newdata = test, type = "class")
                          != test$Y) / nrow(test)
# c. Decision Tree with loss (train)
misclass_train_loss <- sum(predict(fit_loss, newdata = train, type = "class")
                           != train$Y) / nrow(train)
# c. Decision Tree with loss (validation)
misclass_val_loss <- sum(predict(fit_loss, newdata = test, type = "class")
                         != test$Y) / nrow(test)
misclass_df <- data.frame(Data = c("Train", "Testing","Tree size"),
                          Default = c(misclass_train_def, misclass_val_def,nrow(fit_def$frame)),
                          Prune_tree = c(misclass_train_prune, misclass_val_prune,nrow(fit_prune$frame)),
                          Loss_Mat_tree = c(misclass_train_loss, misclass_val_loss,nrow(fit_loss$frame)))

knitr::kable(misclass_df)

imp<- data.frame(Variable=fit_prune$variable.importance)
imp$Var<-rownames(imp)
colnames(imp)<-c("Importance","Variable")
fig<-ggplot(imp, aes(x = Importance, y= Variable)) +  geom_point()
fig
fit_logreg <- glm(Y ~ ., data = train, family = "binomial")
glmnet(x, y, family = "binomial", alpha = 1, lambda = NULL)

x<-model.matrix(Y~., train)[,-1]
```

```
y<- train$Y
cv_lambda <- cv.glmnet(x, y, alpha = 1, family = "binomial")
plot(cv_lambda)
cv_lambda$lambda.min
fit_def <- glm(Y ~ ., data = train, family = "binomial")
fit_pena<-glmnet(x, y, alpha = 1,lambda = cv_lambda$lambda.min, family = "binomial")


prob_def_train <- predict(fit_def,newdata =  train)
predc_def_train <- ifelse(prob_def_train > 0.5, 1, 0)
obs_def_train <- train$Y



xtrain <- model.matrix(Y ~., train)[,-1]
prob_pena_train <- predict(fit_pena,newx =  xtrain)
predc_pena_train <- ifelse(prob_pena_train > 0.5, 1, 0)
obs_pena_train <- train$Y


prob_def_test <- predict(fit_def,newdata =  test)
predc_def_test <- ifelse(prob_def_test > 0.5, 1, 0)
obs_def_test <- test$Y


xtest <- model.matrix(Y ~., test)[,-1]
prob_pena_test <- predict(fit_pena,newx =  xtest)
predc_pena_test <- ifelse(prob_pena_test > 0.5, 1, 0)
obs_pena_test <- test$Y


res<-data.frame(Data=c("Defult","Penalized "),
                Misclassification_Train=c(1-mean(predc_def_train == obs_def_train),
                1-mean(predc_pena_train == obs_pena_train)),
                Misclassification_Test=c(1-mean(predc_def_test == obs_def_test),
                1-mean(predc_pena_test == obs_pena_test)))
knitr::kable(res)

fit_logreg<-glm(Y ~., data = train, family = "binomial")
fit_tree<-rpart(Y~.,data = train,method = "class",
                control = list(minsplit = 16, maxdepth = 22,cp=0.01))
pred_tree <- predict(fit_tree, newdata = test)
pred_logreg <- predict(fit_logreg,newdata =  test)
roc(test$Y ~ pred_tree[,1],plot=TRUE,print.auc=TRUE,
                          col="gray",lwd =4,legacy.axes=TRUE,main="ROC Curves")
roc(test$Y ~ pred_logreg,plot=TRUE,print.auc=TRUE,col="black",lwd = 4,
              print.auc.y=0.4,legacy.axes=TRUE,add = TRUE)
summary(Ride_Halling)
summary(fit_logreg)
```


Lindholm, Andreas, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. 2022. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press. https://smlbook.org.

Therneau, Terry M, Elizabeth J Atkinson, and others. 1997. "An Introduction to Recursive Partitioning

Using the RPART Routines." Technical report Mayo Foundation.