Predicting customer behavior using Machine Learning models

1 Break-Even Response Rate & ROI

```r
cost_per_offer <- 1.5
amazon_prime_fee <- 8.99
avg_revenue <-40
COGS <- 0.7
shipping <- 6

profit_per_customer <- avg_revenue*(1-COGS) + amazon_prime_fee - shipping #We multiply the average
#revenue with the percentage that is not COGS, then add revenue generated by the prime membership
#and subtract the shipping cost supported by Amazon.



breakeven_response_rate <- cost_per_offer/profit_per_customer
```

```r
# pls do not modify the codes below
# these are for TAs to check results
print(paste("cost_per_offer is ", cost_per_offer))
```

```
[1] "cost_per_offer is  1.5"
```

```r
print(paste("profit_per_customer is", profit_per_customer))
```

```
[1] "profit_per_customer is 14.99"
```

```r
print(paste("breakeven_response_rate is", breakeven_response_rate))
```

```
[1] "breakeven_response_rate is 0.10006671114076"
```

```r
sub_sum <- sum(data_full$subscribe == "yes", na.rm = TRUE) #Filtered and summed only
#the customers who are subscribers (ignored the null values).
total_costs_of_mailing_blanket <- cost_per_offer * 10000 ##The total cost of mailing
#all 10000 customers. Whether they will subscribe or not, we still incur this cost.

total_profit_blanket <- profit_per_customer * sub_sum #Determined the expected profit.



ROI_blanket <- (total_profit_blanket-total_costs_of_mailing_blanket)/
  total_costs_of_mailing_blanket
```

```r
print(paste("total_costs_of_mailing_blanket is ", total_costs_of_mailing_blanket))
```

```
[1] "total_costs_of_mailing_blanket is  15000"
```

```r
print(paste("total_profit_blanket is ", total_profit_blanket))
```

```
[1] "total_profit_blanket is  12561.62"
```

```r
print(paste("ROI_blanket is ", ROI_blanket))
```

```
[1] "ROI_blanket is  -0.162558666666667"
```

2 Unsupervised Learning for Segmentation and Targeting

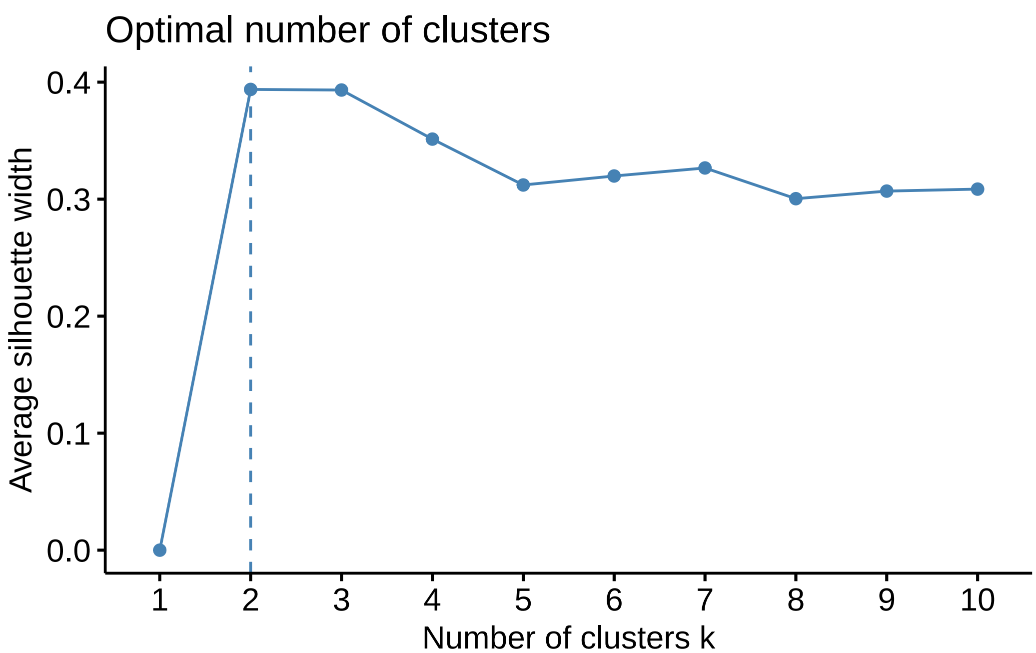
```r
# RFM variables
rfm <- data_full%>%
  mutate(recency = last,  # days since last purchase
         frequency = home + sports + clothes + health + books + digital + toys,
         #total purchases across categories
         monetary_value = electronics+nonelectronics  #total spending

  )

summary(rfm[c("recency", "frequency", "monetary_value")])
```

```
    recency        frequency     monetary_value
 Min.   : 1.00   Min.   : 1.00   Min.   : 15.0
 1st Qu.: 7.00   1st Qu.: 1.00   1st Qu.:128.0
 Median :11.00   Median : 2.00   Median :209.0
 Mean   :12.26   Mean   : 3.85   Mean   :208.2
 3rd Qu.:15.00   3rd Qu.: 6.00   3rd Qu.:284.0
 Max.   :35.00   Max.   :12.00   Max.   :478.0
```

```r
#Scaling data
data_kmeans <- rfm %>%
    mutate(
        recency = scale(recency),
        frequency = scale(frequency),
        monetary_value = scale(monetary_value)
    )
#K-means clustering
data_kmeans <- data_kmeans %>%
    select(recency, frequency, monetary_value) %>%
    mutate(
```

```
        recency = scale(recency),
        frequency = scale(frequency),
        monetary_value = scale(monetary_value)
    )
```

```
# Determine the optimal number of clusters using the Silhouette method below
set.seed(888)
pacman::p_load(factoextra, cluster)
fviz_nbclust(data_kmeans, kmeans, method = "silhouette")
```




```
# implement k-means clustering below

# do not modify seeds
set.seed(888)

result_kmeans <- kmeans(data_kmeans,
    centers = 2,
    nstart = 10
```

```r
)
```

```r
# use broom::tidy() to check the clusters.
pacman::p_load(broom)
tidy(result_kmeans)
```

```
# A tibble: 2 x 6
   recency frequency monetary_value  size withinss cluster
     <dbl>     <dbl>          <dbl> <int>    <dbl> <fct>
1 -0.00843    -0.547         -0.344  7263   13207. 1
2  0.0224      1.45           0.912  2737    5712. 2
```

```r
data_full <- data_full %>%
    mutate(subscribe = ifelse(subscribe == "yes", 1, 0))
data_full <- data_full %>%
    mutate(segment = result_kmeans$cluster)
data_full %>%
    group_by(segment) %>%
    summarise(avg_subscribe_rate = mean(subscribe, na.rm = T)) %>%
    ungroup()
```

```
# A tibble: 2 x 2
  segment avg_subscribe_rate
    <int>              <dbl>
1       1             0.0680
2       2             0.126
```

```r
# ROI for cluster 1 and 2
cluster2 <- sum(result_kmeans$cluster == 2, na.rm = TRUE)
total_costs_of_mailing_kmeans <- cost_per_offer*cluster2
total_profit_kmeans <- profit_per_customer*cluster2* 0.12568506
ROI_kmeans <- (total_profit_kmeans-total_costs_of_mailing_kmeans)/
  total_costs_of_mailing_kmeans
```

```

cluster1 <- sum(result_kmeans$cluster == 1, na.rm = TRUE)
total_costs_of_mailing_kmeans <- cost_per_offer*cluster1
total_profit_kmeans <- profit_per_customer*cluster1* 0.06801597
ROI_kmeans1 <- (total_profit_kmeans-total_costs_of_mailing_kmeans)/
  total_costs_of_mailing_kmeans

ROI_kmeans
```

```
[1] 0.2560127
```

```
ROI_kmeans1
```

```
[1] -0.3202937
```

```
print(paste("ROI_kmeans is ", ROI_kmeans))
```

```
[1] "ROI_kmeans is  0.2560126996"
```

3 Decision Tree Analysis

```
data_tree <- rfm %>%
    select(-user_id) %>%
    select(-gender) %>%
    select(-city)
# set seed
set.seed(1314520)

data_tree <- rfm %>%
    mutate(subscribe = ifelse(subscribe == "yes", 1, 0))
n_rows_data_tree <- nrow(data_tree)
```

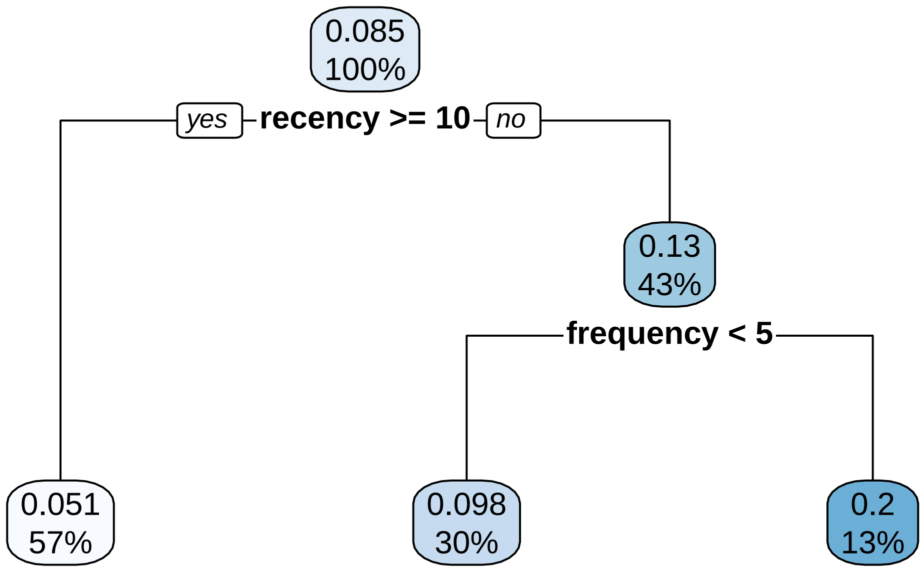
```r
training_set_index <- sample(
  x = 1:n_rows_data_tree,
  size = 0.75*n_rows_data_tree,
  replace = FALSE
)


# create data_training and data_test
data_training <- data_tree %>%
  slice(training_set_index)

data_test <- data_tree %>%
  slice(-training_set_index)
```

```r
# This is to print out first 5 customers
training_set_index[1:5]
```

```
[1] 3620   43 3574 4308 7387
```

```r
pacman::p_load(rpart, rpart.plot)

# train model tree1 below

tree1 <- rpart(
  formula = subscribe ~ recency + frequency + monetary_value, #subscribe is the
  #variable Tom tries to predict based on predictors
  #(recency, frequency, monetary_value).
  data = data_training, #training the model
  method = "anova" #probabilities
)


# visualize tree1 below
```

```

rpart.plot(tree1)
```




```
# ROI for tree1
prediction_from_decision_tree <- predict(tree1, data_test)
data_test <- data_test %>%
    mutate(predicted_prob_decisiontree = prediction_from_decision_tree)
data_test <- data_test %>%
    mutate(is_target_decisiontree = ifelse(predicted_prob_decisiontree >
                                           breakeven_response_rate, 1, 0))
total_costs_of_mailing_decisiontree <- cost_per_offer *
  sum(data_test$is_target_decisiontree)
data_test_targeted_customers <- data_test %>%
    filter(is_target_decisiontree == 1)
total_profit_decisiontree <- sum(data_test_targeted_customers$subscribe) *
  profit_per_customer

# Compute ROI
```

```
ROI_decisiontree <-
  (total_profit_decisiontree - total_costs_of_mailing_decisiontree)/
  total_costs_of_mailing_decisiontree

ROI_decisiontree
```

```
[1] 0.7185987
```

4 Random Forest

```
pacman::p_load(ranger)
set.seed(888)
# train the random forest model below
randomforest <- ranger(
  formula = subscribe ~ recency + frequency + monetary_value, #subscribe is the
  #variable Tom tries to predict based on predictors
  #(recency, frequency, monetary_value)
  data = data_training, # training the model
  probability = TRUE, #TRUE because we are interested in finding the probability
  #of a customer subscribing
  num.trees = 5000
)

prediction_from_randomforest <- predict(randomforest, data_test)

data_test <- data_test %>%
    mutate(predicted_prob_randomforest = prediction_from_randomforest$predictions[, 2])

data_test <- data_test %>%
    mutate(is_target_randomforest = ifelse(predicted_prob_randomforest >
                                            breakeven_response_rate, 1, 0))
```

```r

total_costs_of_mailing_randomforest <- cost_per_offer *
  sum(data_test$is_target_randomforest)



data_responding_targeted_customers <- data_test %>%
    filter(is_target_randomforest == 1) %>%
    filter(subscribe == 1)

# total profits from responding customers
total_profit_randomforest <- nrow(data_responding_targeted_customers) * profit_per_customer

# Compute ROI
ROI_randomforest <-
  (total_profit_randomforest - total_costs_of_mailing_randomforest) /
  total_costs_of_mailing_randomforest

ROI_randomforest
```

```
[1] 0.4295828
```

```r
print(paste("ROI_randomforest is ", ROI_randomforest))
```

```
[1] "ROI_randomforest is  0.429582760201743"
```