MSIN0094 Second Assignment Answer Sheet

Anonymous Candidate Number: TBDH9

Self-reported word count: 1478 words

1. Break-Even Response Rate (16pts)

Question 1. Compute the break-even response rate for Tom's targeting campaign based on the cost information given and assign it into a variable named `breakeven_response_rate`. Explain the role of break-even response rate in the focal targeted marketing campaign. (**7 pt**)

The break-even response rate applies the core break-even analysis framework to a targeted marketing context. It identifies the minimum response probability at which the expected incremental profit from a customer equals the mailing cost. If predicted response probability exceeds this threshold, then expected incremental profit is positive, if below, it is negative.

This threshold becomes the campaign's go/no-go rule (Ahmadi 2024). Customers whose predicted probabilities exceed this break-even response rate deliver positive expected value, while those below it generate negative expected value, meaning the campaign destroys value in expectation.

Therefore, the break-even response rate translates profitability logic to the individual customer level. It ensures that targeting decisions remain economically rational.

```r
# complete the code below.
# you can create additional intermediate variables if needed
# Step 1. The variables are taken from the assessment brief
cost_per_offer <- 1.5
COGS <- 0.9
sub_fee <- 6.99
rev_sub <- 40
ship_cost <- 4

# Step 2. Calculating the profit per customer
# (expected CLV for this offer), based on:
# subscription fee (sub_fee) for the first month,
# additional revenue from future subscription months (rev_sub),
# COGS on subscription revenue,
# shipping costs.
# Formula: profit_per_customer = sub_fee + rev_sub * (1 - COGS) - ship_cost.
# The shipping cost is substracted because Amazon covers it
```

```
profit_per_customer <- sub_fee + (rev_sub * (1 - COGS)) - ship_cost

# Step 3. Computing break-even response rate.
# The wanted minimum fraction of mailed customers is that must respond
# such that total profit from responders covers total mailing cost.

breakeven_response_rate <- cost_per_offer/profit_per_customer

# 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"

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

[1] "profit_per_customer is 6.99"

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

[1] "breakeven_response_rate is 0.214592274678112"
```

Question 2. Compute the ROI of marketing for **blanket marketing** by sending offers to all customers in the `data_full`. (**5 pt**)

Blanket marketing is not profitable under the profitability analysis framework because the pilot results show that the realised response rate is far below the break-even response rate. This means the expected contribution margin from an average customer does not cover the mailing cost. As a consequence, the campaign produces negative expected incremental profit, which appears directly in the negative ROI. Under the ROI decision rule, such outcomes are classified as value-destroying.

Since the pilot sample was randomly drawn, it gives an unbiased estimate of the wider customer population. Same logic applies to the remaining customers: scaling a negative expected value simply scales the loss.

Blanket marketing assumes customer homogeneity. Treating all customers as equally likely to convert disregards these drivers and leads to systematically inefficient allocation of marketing spend.

In line with break-even analysis and targeted marketing logic, Tom should not proceed with blanket marketing for the rest of the customer base.

```
# Step 1. Computing the number of customers
# in the full dataset using nrow()
# nrow() returns the number of rows, therefore the
# number of customers
n_customers <- nrow(data_full)
```

```r
# Step2. Counting how many customers actually
# subscribed ("yes") in the dataset. This is the
# observed number of responders to the offer.

responders <- sum(data_full$subscribe == "yes")

# Step 3. Computing a "response rate" measure.
# Here the mean of the scalar 'responders' is taken,
# which simply returns that same value.
# (If divide by n_customers,
# response_rate would be responders / n_customers.)

response_rate <- mean(responders)

# Step 4. Computing the total cost of mailing when Tom
# mails everyone (blanket marketing).
# Total cost = number of customers × cost per mailed offer.

total_costs_of_mailing_blanket <- n_customers * cost_per_offer

# Step 5. Computing the total profit under blanket marketing.
# Only customers who subscribe (responders) generate profit_per_customer,
# so total profit = profit_per_customer × number of responders.
total_profit_blanket <- profit_per_customer * responders

# Step 6. Computing the ROI for the blanket mailing strategy.
# ROI = (total profit − total cost) / total cost.

ROI_blanket <- (total_profit_blanket - total_costs_of_mailing_blanket) /
total_costs_of_mailing_blanket

# do not modify the code below
print(paste("total_costs_of_mailing_blanket is ",
total_costs_of_mailing_blanket))

[1] "total_costs_of_mailing_blanket is  7500"

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

[1] "total_profit_blanket is  5857.62"

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

[1] "ROI_blanket is  -0.218984"
```

2. Descriptive Analytics for Segmentation and Targeting: RFM Analysis (24 pts)

Question 3. Use `dplyr` data wrangling tools to compute the RFM variables `recency`, `frequency`, and `monetary_value` based on existing variables in the data. Also create a binary numeric variable `subscribe_yes` which equals 1 if a user subscribes and 0 otherwise. Report the summary statistics AND correlation table of `recency`, `frequency`, `monetary_value`, and `subscribe_yes`. Discuss any notable findings relevant for targeted marketing from the correlation table (**10 pts**)

The summary statistics show that all RFM variables and subscribe_yes contain no missing values and exhibit substantial dispersion, indicating good data quality for segmentation and strong behavioural heterogeneity in the customer base. Recency (days since last purchase) ranges widely, suggesting large differences in current engagement, while frequency and monetary_value display pronounced spreads that reflect variation in purchasing intensity and spend.

Recency emerges as the strongest predictor of subscription behaviour. Its negative and non-trivial correlation with subscribe_yes indicates that more recent customers are significantly more likely to subscribe, aligning with the RFM principle that recency is the primary behavioural signal and should anchor targeting decisions. Frequency and monetary_value increase subscription probability but act mainly as refinement variables within the pool of recent customers rather than as independent predictors.

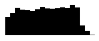
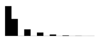
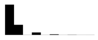
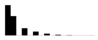
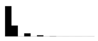
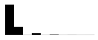
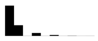
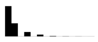
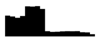
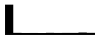
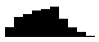
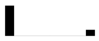
The three RFM dimensions capture complementary behavioural information. Frequency and monetary_value are moderately correlated, reflecting that heavier buyers typically spend more, whereas recency is nearly uncorrelated with either variable. This orthogonality shows that “how recently” someone purchased is largely independent of “how often” or “how much” they buy, validating the use of all three RFM dimensions rather than relying on any single metric.

Overall, the correlation structure implies that targeting should prioritise customers with low recency, and within that group, favour those with higher frequency and monetary_value. This combination maximises predicted response probability relative to the break-even threshold and therefore maximises expected incremental profit and ROI compared with blanket marketing.

```r
# create the required variables below

# Step 1. Loading dplyr explicitly (even though
# pacman::p_load was called before),
# to ensure that dplyr verbs are available.
library(dplyr)

# Step 2. Starting from the full dataset
# and derive RFM and response variables.
```

```r
data_RFM <- data_full %>%
  mutate(
    recency = last,
    frequency = rowSums(across(c(home,
                                 sports, clothes,
                                 health, books, digital,
                                 toys)), na.rm = TRUE),
    monetary_value = rowSums(across(c(electronics,
                                      nonelectronics)),
                             na.rm = TRUE),

    subscribe_yes = ifelse(subscribe == "yes", 1, 0)
  )

# Recency representing how recently the customer purchased
# Frequency is the total count of
# purchases across all listed product categories.
# Monetary value as total spend across electronics
# and non-electronics

# rowSums() adds the values across home,
# sports, clothes, health, books, digital, and toys.
# na.rm = TRUE treats missing entries as zeros.
# ifelse is used to derive who is subscribed and who is not

# report the summary statistics below
# Step 1. Loading modelsummary, which provides
# convenient summary table functions.
pacman::p_load(modelsummary)

# Step 2. Requesting a summary of numeric
# variables in data_RFM.
# datasummary_skim(type = "numeric") returns
# distributional statistics (e.g., mean, sd, min, max)
# for each numeric variable in the dataset.
data_RFM %>%
  datasummary_skim(type = "numeric")
```

	Unique	Missing Pct.	Mean	SD	Min	Median	Max	Histogram
user_id	5000	0	15063.4	2872.5	10001.0	15131.0	20000.0	
first	50	0	25.3	18.4	1.0	19.0	99.0	
last	18	0	11.	8.	1.0	11.	35.	

	Unique	Missing Pct.	Mean	SD	Min	Median	Max	Histogram
			8	0		0	0	
electronics	129	0	47.7	39.0	15.0	28.0	155.0	
nonelectronics	343	0	163.1	88.7	0.0	165.0	344.0	
home	8	0	0.8	1.1	0.0	0.0	7.0	
sports	6	0	0.4	0.7	0.0	0.0	5.0	
clothes	9	0	0.9	1.2	0.0	1.0	8.0	
health	8	0	0.5	0.8	0.0	0.0	7.0	
books	6	0	0.3	0.6	0.0	0.0	5.0	
digital	6	0	0.4	0.7	0.0	0.0	5.0	
toys	8	0	0.6	0.9	0.0	0.0	7.0	
recency	18	0	11.8	8.0	1.0	11.0	35.0	
frequency	12	0	3.9	3.5	1.0	2.0	12.0	
monetary_value	445	0	210.8	102.6	15.0	213.0	477.0	
subscribe_yes	2	0	0.2	0.4	0.0	0.0	1.0	

```r
# report the correlation table below

# Step 1. Subseting data_RFM to keep only the
# variables of interest for correlation:
# recency, frequency, monetary_value, and
# the binary subscribe_yes.
data_corr <- data_RFM %>%
  select(recency, frequency, monetary_value, subscribe_yes)
```

```
# Step 2. Computing the correlation matrix for these variables.
# use = "pairwise.complete.obs" means each correlation is computed
# using all rows where both variables are non-missing.
correlation_table <- cor(data_corr, use = "pairwise.complete.obs")

# Step 3. Displaing the resulting correlation matrix,
# for relationships interpretation between
# RFM variables and subscription behaviour.
correlation_table

                    recency    frequency monetary_value subscribe_yes
recency         1.000000000 -0.001123097    0.004861382    -0.1925138
frequency      -0.001123097  1.000000000    0.524403745     0.1521874
monetary_value  0.004861382  0.524403745    1.000000000     0.1078519
subscribe_yes  -0.192513829  0.152187442    0.107851929     1.0000000
```

Question 4. Use R's `dplyr` package to implement the RFM segmentation algorithm described above, dividing customers into 125 R-F-M segments based on their RFM variables. The resulting dataset should have the same rows as `data_RFM` but include the following new variables: `recency_quintile`, `frequency_quintile`, `monetary_quintile`, and `avg_response_rate` (the average response rate for each RFM segment). Report which RFM segment has the highest average response rate? (**8 pts**)

```
# create the RFM segments below

# Step 1. Starting from data_RFM and creating RFM
# quintile variables and segment-level average response rate.
# Fore quintiles, we create them by groups, so they R, F,
# and M do not exist separately, but R -> F -> M
data_RFM <- data_RFM %>%
  mutate(recency_quintile = ntile(-recency, 5)) %>%
  group_by(recency_quintile) %>%
# Grouping by recency_quintile before computing
# frequency quintiles, so frequency quintiles
# are computed within each recency group if needed.
  mutate(frequency_quintile = ntile(frequency, 5)) %>%
  group_by(recency_quintile, frequency_quintile) %>%
# For each recency_quintile, create a frequency_quintile
# (1-5) using ntile().
# Further group by recency and frequency
# quintiles to create monetary quintiles within each R-F cell.
  group_by(recency_quintile, frequency_quintile) %>%
# For each (recency_quintile, frequency_quintile)
# combination, create monetary_quintile.
  mutate(monetary_quintile = ntile(monetary_value, 5)) %>%
# Removing grouping so we can re-group at the R-F-M segment level.
  ungroup() %>%
  group_by(recency_quintile, frequency_quintile, monetary_quintile) %>%
  mutate(avg_response_rate = mean(subscribe_yes)) %>%
  ungroup()
# Ungroup again so that subsequent analyses
# are not constrained by grouping.

# report the RFM segment with highest average response rate below
# Step 1. Grouping the data by RFM segment
# identifiers (recency_quintile, frequency_quintile,
# monetary_quintile).
highest_avg_response_rate_segment <- data_RFM %>%
  group_by(recency_quintile, frequency_quintile, monetary_quintile) %>%
# Step 2. For each RFM segment, computing the
# mean subscribe_yes as avg_response_rate.
  summarise(
    avg_response_rate = mean(subscribe_yes, na.rm = TRUE),
  .groups = "drop") %>%
# Step 3. Ordering segments descending by
# avg_response_rate so the best segment appears first.
  arrange(desc(avg_response_rate)) %>%
# Step 4. Keep only the very top row (segment
# with the highest observed response rate).
  slice_head()

# do not modify the code below, this is for TA to check the results
highest_avg_response_rate_segment %>%
  dplyr::glimpse()

Rows: 1
Columns: 4
$ recency_quintile   <int> 5
$ frequency_quintile <int> 2
$ monetary_quintile  <int> 5
$ avg_response_rate  <dbl> 0.475
```

Question 5. Among the RFM segments created in the previous step, which segment(s) should Tom target for the marketing campaign (i.e., send marketing offers to all customers in those segments)? Compute the ROI of marketing if Tom conducts targeted marketing to the segment you selected. (**6 pts**)

Tom should target the R5-F2-M5 segment: customers with very recent purchases (recency quintile 5), moderate purchase frequency (frequency quintile 2), and very high monetary value (monetary quintile 5).

The rationale is strictly profitability-driven. This segment records the highest average response rate across all 125 RFM cells, and that response rate sits well above the break-even response rate. As a result, the expected contribution margin per mailed customer more than covers the acquisition cost, producing a clearly positive expected incremental profit. When mailing is restricted to this segment, the resulting ROI becomes positive and meaningfully higher than any alternative, in contrast to blanket marketing where ROI is negative.

This targeting choice is fully consistent with RFM logic and customer heterogeneity. These customers pair very strong recent engagement (R) with very high monetary value (M), which jointly increases both their response probability and the profit generated conditional on subscription. Although their frequency score is moderate, this does not materially diminish their expected performance: the combination of high recency and high monetary value more than compensates. Taken together, these characteristics make the R5-F2-M5 group the segment with the highest expected incremental profit per offer and therefore the most attractive and financially defensible target for the campaign.

```r
# Write your codes below to compute the ROI for the targeted marketing based
on RFM segments

# Step 1. Loading dplyr (ensuring pipe and verbs are available).
library(dplyr)

# Step 2: Aggregate customers into RFM
# segments and compute segment size and
# average response rate.
# n_segment = number of customers in this RFM segment.
# avg_response_rate = average subscription rate in this segment.
rfm_segments <- data_RFM %>%
  group_by(recency_quintile, frequency_quintile, monetary_quintile) %>%
  summarise(
    n_segment = n(),
    avg_response_rate = mean(subscribe_yes, na.rm = TRUE),
    .groups = "drop"
  )

# Step 3. Identifying RFM segments whose average
# response rate is above the break-even rate.
# These segments are profitable to target.
rfm_target <- rfm_segments %>%
  dplyr::filter(avg_response_rate > breakeven_response_rate)

# Step 4. Computing total mailing cost for targeting
# only profitable RFM segments. Sum the customer counts
# in profitable segments and multiply by cost_per_offer.

total_costs_of_mailing_RFM <- sum(rfm_target$n_segment) * cost_per_offer

# Step 5. Computing total profit from RFM-based targeting.
# For each targeted segment, expected responders =
# n_segment × avg_response_rate, multiplied by
# profit_per_customer, then summed across all targeted segments.

total_profit_RFM <- sum(rfm_target$n_segment * rfm_target$avg_response_rate)
* profit_per_customer
```

```
# Step 6. Computing the ROI for the RFM-based targeting strategy.
# ROI_RFM = (total profit − total cost) ÷ total cost.
ROI_RFM <- (total_profit_RFM - total_costs_of_mailing_RFM) /
total_costs_of_mailing_RFM

# do not modify the code below
print(paste("total_costs_of_mailing_RFM is ", total_costs_of_mailing_RFM))

[1] "total_costs_of_mailing_RFM is  2820"

print(paste("total_profit_RFM is ", total_profit_RFM))

[1] "total_profit_RFM is  3942.36"

print(paste("ROI_RFM is ", ROI_RFM))

[1] "ROI_RFM is  0.398"
```

3. Unsupervised Learning for Segmentation and Targeting (22 pts)

Question 6. Use the three RFM variables to segment customers using the K-means clustering algorithm. Please use `set.seed(888)` for reproducibility. Specify x and `centers` in the `kmeans()` function, while using the default values for ALL remaining arguments for simplicity. (**8 pts**)

```
# create the RFM variables below
# Step 1. Preparing a reduced dataset with only the
# numeric RFM variables used for k-means.
rfm_for_kmeans <- data_RFM %>%
  dplyr::select(recency, frequency, monetary_value)

# Complete the code below to pre-process the data for k-means clustering

# Step 1. Starting from the RFM dataset for k-means.
data_kmeans <- rfm_for_kmeans %>%
# Dropping any rows with missing values in recency, frequency,
# or monetary_value to ensure k-means sees complete cases only.
  tidyr::drop_na() %>%
# Coercing all columns to numeric (to avoid factors/characters).
  dplyr::mutate(dplyr::across(dplyr::everything(), as.numeric)) %>%
# Converting the dplyr tibble back to a base R data.frame for kmeans().
  as.data.frame()
# Step 2: Standardising the RFM variables.
# scale() transforms each column to mean 0 and standard deviation 1,
# so that variables are on comparable scales for distance-based clustering.
data_kmeans <- as.data.frame(scale(data_kmeans))
# mean 0, sd 1 for all three features
```

```r
# Determine the optimal number of clusters using the Elbow method below
# do not modify the seed below
set.seed(888)

# Step 1. Creating a vector of candidate
# numbers of clusters k = 1 to 10.
ks <- 1:10

# Step 2. For each candidate k, run k-means
# and capture total within-cluster sum of squares.
wss <- sapply(ks, function(k) {
# Run k-means with k clusters on the scaled data_kmeans.
  km <- kmeans(x = data_kmeans, centers = k)  # defaults for all other args
# Return the total within-cluster sum of squares (measure of compactness).
  km$tot.withinss
})

# Step 4. Storing k and their corresponding
# within-cluster sum of squares in a data frame.
elbow_df <- data.frame(k = ks, tot_withinss = wss)

# Step 5. Displaying the table used to plot or inspect the Elbow curve.
elbow_df

    k tot_withinss
1   1    14997.000
2   2     9390.306
3   3     6771.684
4   4     6055.055
5   5     4467.977
6   6     3955.661
7   7     3374.701
8   8     3394.577
9   9     2817.342
10 10     2623.441

# Inspect the "elbow" — diminishing returns after k=4 → choose 4 clusters
(per Tom's conclusion).

# implement k-means clustering below

# do not modify seeds
set.seed(888)

# Runing k-means clustering on the preprocessed RFM data.
result_kmeans <- kmeans(
  x = data_kmeans,    # using the scaled, NA-free matrix
  centers = 4         # K chosen by Elbow; all other arguments at defaults
)
```

```
# do no modify the code below, this is for TA to check the results

# use broom::tidy() to check the clusters.

# Loading broom to convert k-means results into a
# tidy data frame of cluster centres.
pacman::p_load(broom)
result_kmeans %>%
  tidy()

# A tibble: 4 × 6
  recency frequency monetary_value  size withinss cluster
    <dbl>     <dbl>          <dbl> <int>    <dbl> <fct>
1   1.94     -0.234         -0.138   668    1064. 1
2  -0.178     1.57           0.952  1163    1819. 2
3  -0.355    -0.520          0.429  1575    1057. 3
4  -0.332    -0.536         -1.06   1594    1180. 4
```

Question 7 . Among the segments from k-means, which segment should Tom target for the marketing campaign (i.e., send marketing offers to all customers in that segment)? Tip: `result_kmeans$cluster` returns an R vector that tells us which segment each customer is in, in the same order as the rows in `data_kmeans`. (**6 pts**)

```
# Write your codes here to reason which segment to target
# Show numbers from your coding to support your argument.
# Attach cluster labels to rows used in k-means

#Tom should target the k-means cluster with highest subscription rate,
representing recent, frequent, high-spending customers who exceed the break-
even threshold.

# Step 1: Recreate the RFM-only dataset used in k-means.
rfm_for_kmeans <- data_RFM %>% dplyr::select(recency, frequency,
monetary_value)

# Step 2: Identify which rows in rfm_for_kmeans have complete RFM data.
keep_rows <- stats::complete.cases(rfm_for_kmeans)

# Step 3: Subset data_RFM to those complete-case rows and attach a cluster
label to each row.
data_RFM_km <- data_RFM[keep_rows, ] %>%
  dplyr::mutate(cluster_k4 = result_kmeans$cluster)

# Step 4: Summarise each cluster in terms of size and average response rate.
cluster_summary <- data_RFM_km %>%
  dplyr::group_by(cluster_k4) %>%
  dplyr::summarise(
    # Step 4.1: n_cluster = number of customers assigned to this cluster.
    n_cluster = dplyr::n(),
    # Step 4.2: avg_response_rate = mean subscribe_yes in this cluster.
    avg_response_rate = mean(subscribe_yes, na.rm = TRUE),
    .groups = "drop"
  ) %>%
  # Step 4.3: Sort clusters from highest to lowest average response rate
  #           to identify the most attractive segment.
  dplyr::arrange(dplyr::desc(avg_response_rate))

# Step 5: Display the per-cluster summary to support the targeting decision.
cluster_summary
```

```
# A tibble: 4 × 3
  cluster_k4 n_cluster avg_response_rate
       <int>     <int>             <dbl>
1          2      1163            0.262
2          3      1575            0.177
3          4      1594            0.135
4          1       668            0.0584
```

```
# Compute the ROI of marketing if Tom conducts k-means targeted marketing to
the segment you selected

# Step 1. Picking the best-performing cluster (highest avg response)
best_cluster_id <- cluster_summary$cluster_k4[1]

# Step 2. Subsetting rows for that cluster
best_cluster_rows <- data_RFM_km %>%
  dplyr::filter(cluster_k4 == best_cluster_id)

# Step 3. Calculating total cost and total profit for that cluster
total_costs_of_mailing_kmeans <- nrow(best_cluster_rows) * cost_per_offer
total_profit_kmeans <- sum(best_cluster_rows$subscribe_yes == 1) *
profit_per_customer

# Step 4. Computing ROI
ROI_kmeans <- (total_profit_kmeans - total_costs_of_mailing_kmeans) /
total_costs_of_mailing_kmeans

# do not modify the code below

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

```
[1] "ROI_kmeans is  0.222098022355976"
```

Question 8. Discuss the pros and cons of using K-means clustering to conduct segmentation and targeting for this dataset. (**4pts**, 200 words)

K-means clustering offers several strengths for segmentation and targeting in this dataset. It provides an unsupervised learning method capable of uncovering natural structure in customer behaviour without relying on a response variable. Applied to the RFM variables, k-means groups customers based on Euclidean distance in the RFM feature space, allowing the algorithm to identify segments with similar recency, purchase frequency, and monetary_value profiles. K-means is also computationally efficient and scales well to large datasets. Its output is actionable for marketing practice: each customer receives a clear cluster label, enabling cluster-level targeting strategies or differentiated campaign design.

However, the approach carries several limitations. K-means requires the number of clusters to be set in advance; although tools such as the elbow method offer guidance, selecting K remains partly subjective. The algorithm also assumes spherical, equally sized clusters and relies on Euclidean distance, which can misrepresent behavioural patterns when real customer groups are elongated, overlapping, or uneven in size. It is sensitive to variable scaling and initial centroid selection, making standardisation and multiple random starts necessary for reliability. Most importantly, k-means does not incorporate response probability or profitability measures. As a result, behavioural clusters may not align with expected incremental profit or the break-even response rate, meaning that even well-defined clusters still require profitability analysis before informing targeted marketing decisions.

4. Decision Tree Analysis (24 pts)

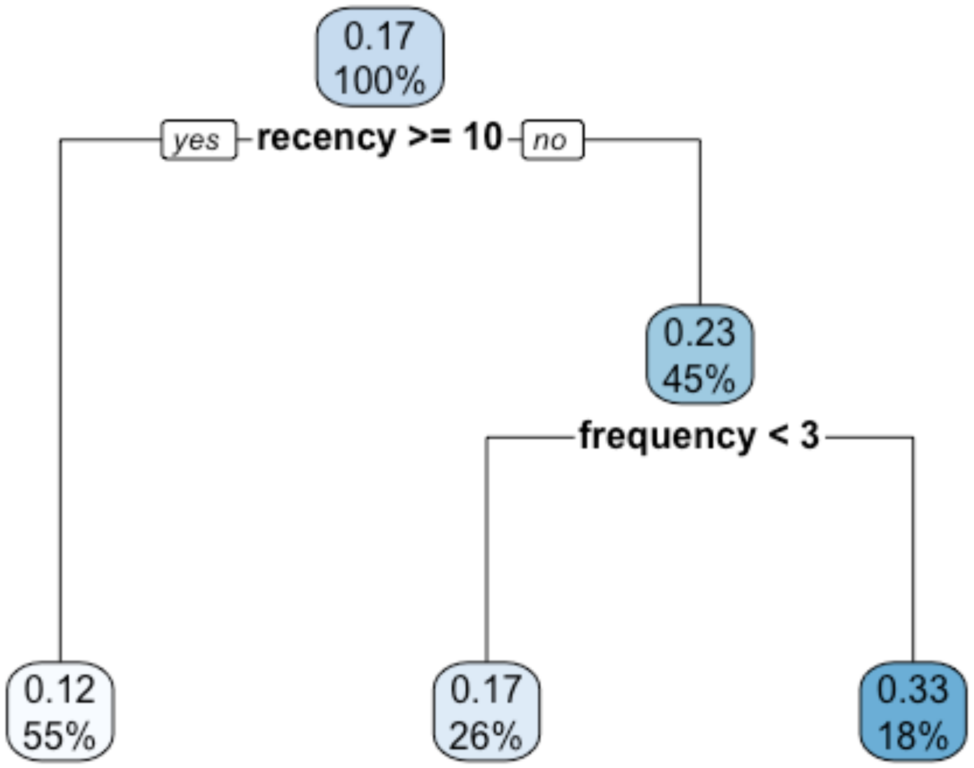
Question 9. Complete the following data preparation tasks before training a decision tree model. (**10 pts**)

```r
# set seed, please do not change the seed
set.seed(1314520)

# Step 1. Starting from data_RFM and create a dataset
# for decision tree modelling.
data_tree <- data_RFM %>%
  mutate(
# Ensuring subscribe_yes is explicitly coded as 1 (yes) or 0 (no).
    subscribe_yes = ifelse(subscribe == "yes", 1, 0)
  )

# Step 2. Randomly sampling indices for the training set.
training_set_index <- sample(
  1:nrow(data_tree),                 # candidate indices = all row indices
  size = floor(0.7 * nrow(data_tree)) # 70% of the data for training
)

# Step 3. Subsetting the data_tree into training
# and test sets using the sampled indices.
data_training <- data_tree[training_set_index, ]

# Step 4. All remaining rows (not in training_set_index)
# form the test set.
data_test <- data_tree[-training_set_index, ]

# Do not modify this code block.
# This is to print out first 5 customers
training_set_index[1:5]

[1] 4439 4646 3620 3803   43
```

Question 10. Complete the following steps to train a decision tree model (**10 pts**)

[placeholder, delete this before writing your answers]

```
# train model tree1 below

# Step 1: Load rpart and rpart.plot packages for decision tree modelling and
visualisation.
pacman::p_load(rpart, rpart.plot)

# Step 2: Fit a decision tree model called tree1 on the training data.
tree1 <-  rpart(
# Model formula — predict subscribe_yes using RFM predictors.
  formula = subscribe_yes ~ recency + frequency + monetary_value,
# Using data_training as the training dataset.
  data = data_training,
# Using method = "anova" for continuous predictions (numeric probability-like
output).
  method = "anova"
)

# Step 3: Plot the fitted decision tree so you can inspect splits and
thresholds visually.
rpart.plot(tree1)
```



Question 11. Compute the ROI from targeted marketing using `tree1`.

```
# Compute the ROI for tree1 below.

# Step 1: Use the fitted tree1 model to predict subscription scores for the
test set.
#         The output is a numeric score for subscribe_yes.
prediction_from_d_tree1 <- predict(tree1, data_test)

# Step 2: Attach the predicted probabilities from the decision tree to
data_test.
data_test <- data_test %>%
  mutate(predict_prob_decisiontree = prediction_from_d_tree1)

# Step 3: Create a binary targeting flag for the decision tree strategy.
#         Customers are targeted if their predicted probability exceeds the
break-even response rate.
data_test <- data_test %>%
  mutate(is_targetdecisiontree1 =
ifelse(predict_prob_decisiontree>breakeven_response_rate, 1, 0))

# Step 4: Compute the total mailing cost under the decision tree targeting
strategy.
```

```
#         Only customers marked as target (is_targetdecisiontree1 == 1) are
mailed.
total_costs_of_mailing_decisiontree1 <- cost_per_offer *
sum(data_test$is_targetdecisiontree1)

# Step 5: Subset the test set to customers who were targeted by the decision
tree.
data_test_targeted_customers <- data_test %>%
  filter(is_targetdecisiontree1 == 1)

# Step 6: Compute total profit under the decision tree strategy.
#         Only targeted customers who actually subscribed (subscribe_yes ==
1) contribute profit_per_customer.
total_profit_decisiontree1 <- profit_per_customer *
sum(data_test_targeted_customers$subscribe_yes)

# Step 7: Compute ROI for the decision tree targeting strategy.
#         ROI_decisiontree1 = (total profit − total cost) ÷ total cost.
ROI_decisiontree1 <- (total_profit_decisiontree1 -
total_costs_of_mailing_decisiontree1)/total_costs_of_mailing_decisiontree1

# do not modify the code below
print(paste("ROI_decisiontree1 is ", ROI_decisiontree1))

[1] "ROI_decisiontree1 is  0.6776"
```

5. Random Forest (20 pts)

Question 12. Train a random forest model with the same set of predictors as `tree1` (**4 pts**)

```
# Train a random forest model with the same predictors as tree1

# Step 1: Load the ranger package, which provides a fast implementation of
random forests.
pacman::p_load(ranger)

# Step 2: Fit a random forest model called randomforest on the training data.
randomforest <- ranger(
  # Step 2.1: Model formula — predict subscribe_yes using RFM predictors.
  formula = subscribe_yes ~ recency + frequency + monetary_value,
  # Step 2.2: Use the same training dataset as the decision tree.
  data = data_training,
  # Step 2.3: num.trees = 2000 → grow 2000 trees, which typically reduces
variance.
  num.trees = 2000,        # number of trees in the forest; more trees →
lower variance
  # Step 2.4: probability = TRUE → ask ranger to return class probabilities
instead of class labels.
  probability = TRUE,      # return class probabilities (required for
targeting)
  # Step 2.5: seed = 888 sets an internal seed for reproducibility of the
random forest.
  seed = 888               # seed set *inside* the model as instructed
)

# Predict probabilities of subscription for the test set
# Step 3: Use the random forest model to predict for data_test.
#         predict(... )$predictions returns a matrix with probabilities for
each class.
rf_predictions <- predict(randomforest, data_test)$predictions[, 2]
# Step 4: Extract the probability of subscribe_yes = 1 from the second column
of the predictions matrix.
# [,2] extracts P(subscribe_yes = 1)

# Targeted marketing based on break-even response rate
# Step 5: Attach the random forest probabilities and targeting flag to the
test set.
data_test <- data_test %>%
  mutate(
    # Step 5.1: Store the predicted probability of subscription from the
random forest.
    predict_prob_randomforest = rf_predictions,
    # Step 5.2: Mark a customer as a target if the predicted probability
exceeds the break-even response rate.
    is_target_randomforest = ifelse(predict_prob_randomforest >
breakeven_response_rate, 1, 0)
  )

# Total mailing cost for targeted customers
# Step 6: Compute the total mailing cost under the random forest targeting
strategy.
#         Only customers with is_target_randomforest == 1 are mailed.
total_costs_randomforest <-
  cost_per_offer * sum(data_test$is_target_randomforest)

# Customers who were targeted AND responded
# Step 7: Subset the test set to customers who were both targeted by random
forest and actually subscribed.
data_test_targeted_rf <- data_test %>%
  filter(is_target_randomforest == 1, subscribe_yes == 1)

# Total profit from targeted campaign
# Step 8: Compute total profit for the random forest strategy.
#         Each targeted-and-responding customer contributes
profit_per_customer.
total_profit_randomforest <-
  nrow(data_test_targeted_rf) * profit_per_customer
```

```
# ROI for the random-forest-based targeting strategy
# Step 9: Compute ROI for the random forest targeting strategy.
#         ROI_randomforest = (total profit − total cost) ÷ total cost.
ROI_randomforest <-
  (total_profit_randomforest - total_costs_randomforest) /
  total_costs_randomforest

# do not modify the code below

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

[1] "ROI_randomforest is  0.486481012658227"
```

Question 13. Based on the results of different models you have obtained in this assignment so far, discuss the fundamental tradeoffs of supervised learning (**8 pts**, 300 words)

Supervised learning faces multiple issues: bias-variance, accuracy-interpretability, and statistical performance vs economic performance.

First, the bias-variance trade-off. A shallow decision tree such as tree1 has relatively high bias but low variance: it approximates the relationship between RFM variables and subscription probability with only a few splits (recency and frequency in our case), which limits overfitting and improves generalisation from the training set to the test set (Ranglani and Gabhane 2024). By contrast, the random forest is a high-variance reduction technique: it aggregates many deep trees built on bootstrap samples and random feature subsets to increase predictive accuracy and reduce overfitting at the individual-tree level (IBM 2023).

Second, there is a clear accuracy-interpretability trade-off. tree1 provides a transparent decision rule (“if recency and frequency satisfy these thresholds, mail the customer”), which is easy to communicate to managers and directly maps into a targeting rule based on the break-even response rate. The random forest, although more flexible, behaves as a black box: it produces response probabilities but its internal structure is not easily interpretable in terms of simple RFM segments. This reduces explainability even if it improves classification metrics such as error rate or AUC.

Third, the results highlight a trade-off between statistical performance and economic performance. Both supervised models outperform blanket marketing in terms of ROI, but the simpler decision tree delivers the highest ROI, whereas the more sophisticated random forest yields a lower, though still positive, ROI. This illustrates that better statistical fit does not automatically translate into higher expected incremental profit: differences in probability calibration around the break-even threshold, and in how customers are ranked by predicted response probability, directly affect which customers are mailed and therefore the final cost-benefit outcome (Pagliaro et al. 2025).

Overall, supervised learning requires balancing model complexity, stability, interpretability, and profitability when designing targeting strategies.

Question 14. Discuss at least two recommendations for Tom to improve the performance of the random forest model based without collecting more data. (**8pts**, 250 words)

One way to improve the random forest's performance without collecting more data is hyperparameter tuning to manage the bias-variance trade-off. The current model uses default settings; Tom can adjust parameters such as the number of variables tried at each split (mtry), the minimum node size, and the maximum tree depth. Reducing maximum depth or increasing minimum node size can reduce overfitting and improve out-of-sample generalisation, while tuning mtry can enhance predictive accuracy by balancing tree diversity and individual-tree strength (Probst, Wrignt, and Boulesteix 2019). These choices should be evaluated using a train-validation split or cross-validation, comparing test-set performance, not just training fit.

A second recommendation is to optimise the targeting rule derived from the forest rather than the model structure alone. The forest outputs predicted response probabilities, but mailing decisions depend on the break-even response rate and marketing ROI, not on classification accuracy. Tom should sort customers by predicted probability, then vary the decision threshold (or proportion of customers mailed) and compute the implied expected incremental profit and ROI for each threshold. Choosing the cut-off that maximises ROI explicitly aligns the model with the cost-benefit decision rule, ensuring that probability estimates around the break-even region are used optimally.

A third improvement is feature engineering from existing variables: for example, adding non-linear transformations (e.g. log monetary_value), interaction terms (recency × frequency), or coarse RFM segment dummies. These re-parameterisations of the same data can help the random forest capture more relevant structure in customer heterogeneity and sharpen the ranking of customers by expected profitability, again without requiring any new observations.

Reference List

Ahmadi, Iman. 2024. "Selecting Audience Segments for Online Advertising." *Telematics and Informatics* 82: 102018. https://www.sciencedirect.com/science/article/pii/S0167811623000502

IBM. 2023. "What Is Random Forest?" *IBM Think Blog*. https://www.ibm.com/think/topics/random-forest

Pagliaro, Antonio, Giuseppe Vizzari, Mona A. Mardini, Marc Estevez, and Enrico Zio. 2025. "Artificial Intelligence vs. Efficient Markets: A Critical Assessment." *Electronics* 14 (9): 1721. https://www.mdpi.com/2079-9292/14/9/1721

Probst, Philipp, Marvin N. Wright, and Anne-Laure Boulesteix. 2019. "Hyperparameters and Tuning Strategies for Random Forest." *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 9 (3): e1301. https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1301

Ranglani, Harsh, and Manish Gabhane. 2024. “Empirical Analysis of the Bias–Variance Tradeoff across Multiple Machine Learning Algorithms.” *Machine Learning and Applications: An International Journal* 11 (4): 1–15. https://aircconline.com/mlaij/V11N4/11424mlaij01.pdf